ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Вебсайт кулінарних рецептів з використанням багатокритеріальних фільтрів та рекомендаційної системи»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-22
Максим МЕЛЬНИК
Керівник: старший викладач кафедри
комп'ютерних наук
Олег ЛУКУТІН

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2026 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри комп'ютерних наук
_____ Сергій МІЧКІВСЬКИЙ
«__» _____ 20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мельник Максим Іванович

Тема роботи	Вебсайт кулінарних рецептів з використанням багатокритеріальних фільтрів та рекомендаційної системи
Номер та дата наказу про затвердження теми	№133-7 від 22 грудня 2025 року
Коротка постановка завдання	Розробка веб-сайту кулінарних рецептів. На сайті реалізувати використання багатокритеріальних фільтрів та рекомендаційну систему підбору рецептів.
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Огляд рекомендаційних систем та їх застосування в вебсервісах. URL: https://developers.google.com/machine-learning/recommendation 2. Офіційна документація JavaScript (MDN Web Docs). URL: https://developer.mozilla.org/uk/docs/Web/JavaScript
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп'ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2025 р.

Керівник Олег ЛУКУТІН

Здобувач освітнього ступеня бакалавра Максим МЕЛЬНИК

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
	Підготовчий етап		
1	Вибір напряму дослідження	01.12.2025 р.	*виконано*
2	Формування теми та призначення керівника	15.12.2025 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	22.12.2025 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	26.12.2025 р.	*виконано*
	Основний етап		
5	Розробка концепції кваліфікаційної роботи	12.01.2026 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	19.01.2026 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	16.03.2026 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	23.03.2026 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	30.03-04.04.2026 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	06.04.2026 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	13.04.2026 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	20.04.2026 р.	*виконано*
	Завершальний етап		
13	Представлення рукопису для перевірки на плагіат	27.04-03.05.2026 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	04.05-10.05.2026 р.	*виконано*
15	Передзахист кваліфікаційної роботи	11.05-15.05.2026 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	18.05-05.06.2026 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	08.06-14.06.2026 р.	
18	Доопрацювання доповіді та презентації для захисту	08.06-14.06.2026 р.	
19	Захист кваліфікаційної роботи	15.06-21.06.2026 р.	

Керівник _____ Олег ЛУКУТІН

Здобувач _____ Максим МЕЛЬНИК

Мельник М. І. Вебсайт кулінарних рецептів з використанням багатокритеріальних фільтрів та рекомендаційної системи.

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2026.

Ключові слова: вебзастосунок, рецепти, кулінарний сервіс, багатокритеріальна фільтрація, рекомендаційна система, користувацькі профілі, база даних, система оцінювання, обране, соціальні функції.

Рис. 15. Бібліограф.: 11 найм.

Melnyk M. I. Website for cooking recipes using multi-criteria filters and a recommendation system.

Project explanatory note by specialty 122 – Computer science. – «KROK» University, Educational and Scientific Institute of information and communication technologies, Department of Computer Science, Kyiv, 2026.

Keywords: web application, recipes, culinary service, multi-criteria filtering, recommendation system, user profiles, database, rating system, favorites, social features.

Fig. 15. Bibliography: 11 Items.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах розвитку інформаційних технологій та цифровізації повсякденного життя все більшої популярності набувають вебзастосунки, що забезпечують зручний доступ до інформації та автоматизацію різних процесів. Однією з таких сфер є кулінарія, яка залишається актуальною для широкого кола користувачів незалежно від віку та професійної діяльності. З розвитком інтернет-технологій значна кількість рецептів стала доступною онлайн, однак процес їх пошуку та вибору часто є ускладненим через надлишок інформації, відсутність ефективної фільтрації та обмежені можливості персоналізації.

Більшість існуючих вебресурсів із рецептами мають недоліки, зокрема незручний інтерфейс, обмежені можливості пошуку за інгредієнтами, часом приготування або дієтичними вподобаннями, а також відсутність адаптивних рекомендацій для користувача. У результаті користувач витрачає значну кількість часу на пошук відповідного рецепту. Це обумовлює необхідність створення сучасних вебзастосунків, які забезпечують багатокритеріальну фільтрацію та рекомендаційну підтримку.

Розробка вебзастосунку каталогу рецептів із багатокрите ріальною фільтрацією та рекомендаційною системою є актуальним напрямом, оскільки дозволяє підвищити ефективність пошуку, покращити користувацький досвід та забезпечити персоналізовану взаємодію з системою.

Мета дослідження полягає у розробці та аналізі вебзастосунку каталогу рецептів, який включає механізми пошуку, багатокритеріальної фільтрації, сортування та рекомендацій для полегшення процесу вибору рецептів користувачами.

Завдання дослідження полягають у наступному:

1) проаналізувати предметну область та існуючі вебресурси з рецептами, визначити їхні переваги та недоліки;

2) сформулювати вимоги до вебзастосунку каталогу рецептів;

3) розробити структуру та архітектуру програмного продукту;

4) реалізувати функціонал пошуку, фільтрації, сортування та рекомендацій;

5) розробити систему користувацьких акаунтів, обраного та оцінювання рецептів;

6) провести тестування розробленого програмного продукту.

Об’єктом дослідження є процес пошуку та вибору кулінарних рецептів у вебсередовищі.

Предметом дослідження є методи та засоби розробки вебзастосунку каталогу рецептів із використанням багатокритеріальної фільтрації та рекомендаційних алгоритмів.

Методи дослідження. У роботі використано методи аналізу та узагальнення інформації, методи проєктування інформаційних систем, алгоритмічні підходи до фільтрації та рекомендацій, а також методи тестування програмного забезпечення.

Практичне значення одержаних результатів полягає у створенні функціонального вебзастосунку, який може бути використаний користувачами для зручного пошуку та вибору рецептів, а також слугувати основою для подальшого розвитку подібних систем.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та переліку джерел посилання. Пояснювальна записка містить 16 рисунків. Загальний обсяг пояснювальної записки становить 46 сторінок,

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ КАТАЛОГУ РЕЦЕПТІВ З БАГАТОКРИТЕРІАЛЬНОЮ ФІЛЬТРАЦІЄЮ ТА РЕКОМЕНДАЦІЙНОЮ СИСТЕМОЮ

1.1 Предметна область

Предметна область даної роботи охоплює сферу вебзастосунків для пошуку, зберігання та обробки кулінарних рецептів. У сучасному цифровому середовищі кулінарні ресурси відіграють важливу роль у повсякденному житті користувачів, оскільки дозволяють швидко отримати доступ до великої кількості рецептів, рекомендацій та порад щодо приготування страв.

З розвитком інформаційних технологій значно зросла кількість онлайн-платформ, які надають доступ до кулінарного контенту. До таких платформ належать спеціалізовані сайти рецептів, мобільні додатки, а також соціальні мережі, де користувачі можуть ділитися власними рецептами. Проте велика кількість інформації ускладнює процес пошуку необхідного рецепту, особливо коли користувач має конкретні вимоги, наприклад, обмеження за інгредієнтами, часом приготування або типом дієти.

Основною проблемою предметної області є недостатня ефективність пошуку та відбору рецептів. Більшість існуючих систем використовують обмежені механізми фільтрації, які не враховують одночасно декілька параметрів. Це призводить до того, що користувач змушений витрачати додатковий час на перегляд великої кількості нерелевантних результатів.

Ще однією важливою проблемою є відсутність персоналізації. Багато платформ не враховують індивідуальні вподобання користувачів, такі як улюблені інгредієнти, типи кухні або попередні дії. У результаті рекомендації носять загальний характер і не відповідають очікуванням користувача.

Також слід відзначити обмежені соціальні можливості більшості кулінарних сервісів. Наявність функцій взаємодії між користувачами, таких як

підписки, оцінювання рецептів, додавання в обране, може значно підвищити залученість користувачів та покращити якість контенту.

У межах даної предметної області можна виділити такі основні сутності:

- користувачі;
- рецепти;
- інгредієнти;
- категорії (типи страв, кухні, дієти);
- оцінки та відгуки;
- обране;
- підписки між користувачами.

Ключовими процесами є:

- пошук рецептів за ключовими словами;
- багатокритеріальна фільтрація;
- сортування результатів;
- формування рекомендацій;
- взаємодія користувачів із контентом (оцінювання, збереження, створення рецептів).

Таким чином, предметна область характеризується великою кількістю даних, складністю їх обробки та необхідністю забезпечення зручного доступу до релевантної інформації. Це обумовлює доцільність розробки вебзастосунку, який поєднує ефективні механізми пошуку, фільтрації та рекомендацій із сучасним інтерфейсом та соціальними функціями.

1.2 Огляд аналогів та обґрунтування очікуваних переваг порівняно з існуючими аналогами

У межах дослідження предметної області було проведено аналіз існуючих вебресурсів, що надають доступ до кулінарних рецептів. Для порівняння обрано такі популярні платформи: Cookpad, Shuba та Tasty, які мають різний підхід до організації контенту та взаємодії з користувачем.

Cookpad (рис 1.1) є міжнародною платформою для обміну кулінарними рецептами, яка орієнтована на взаємодію між користувачами та створення контенту самими користувачами. Основною особливістю сервісу є активна спільнота, що дозволяє користувачам публікувати власні рецепти, ділитися досвідом приготування страв, залишати коментарі та оцінювати рецепти інших користувачів. Завдяки цьому формується велика база різноманітного контенту, яка постійно оновлюється.

Переваги:

- велика кількість рецептів;
- розвинена соціальна складова;
- можливість створення власних рецептів.

Недоліки:

- частина функціоналу доступна лише за підпискою;
- інтерфейс виглядає застарілим;
- відсутність рекомендаційної системи.

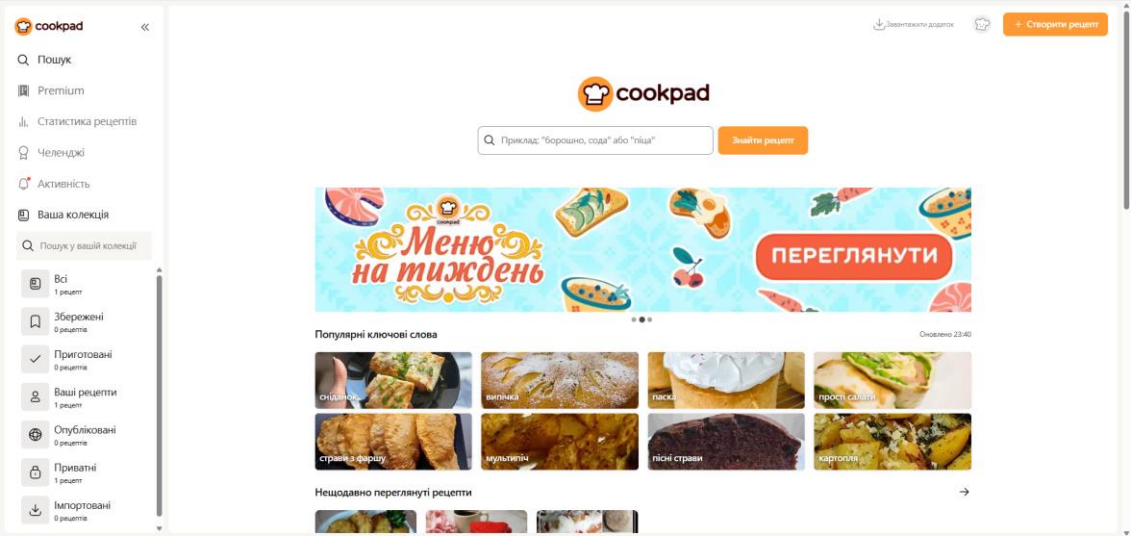

Рисунок 1.1 – Cookpad

Джерело: [1]

Shuba (рис 1.2) є сучасним кулінарним вебресурсом, який орієнтований на зручність використання та візуальну привабливість інтерфейсу. Платформа

пропонує користувачам широкий вибір рецептів, структурованих за категоріями, а також надає можливість створення та публікації власних рецептів. Особлива увага приділяється дизайну, що забезпечує комфортну взаємодію користувача з системою.

Переваги:

- велика кількість рецептів;
- можливість публікації власних рецептів;
- сучасний дизайн.

Недоліки:

- обмежені можливості фільтрації;
- відсутність рекомендаційної системи.

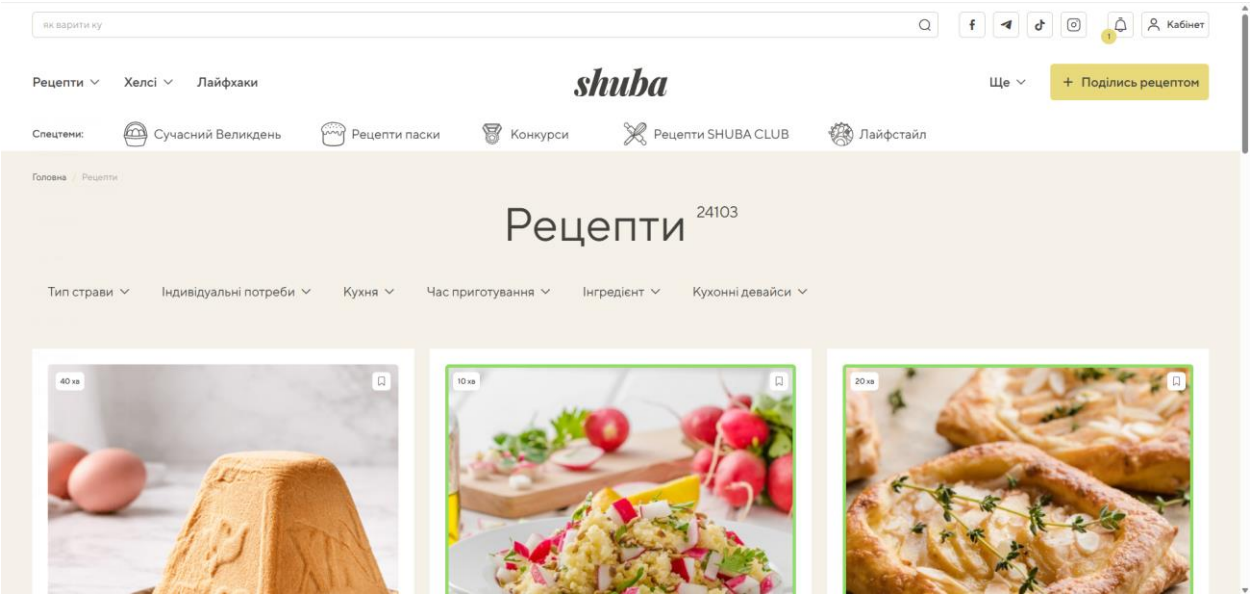

Рисунок 1.2 – Shuba

Джерело: [2]

Tasty (рис 1.3) є сучасним кулінарним вебресурсом, який орієнтований на простоту використання та швидкий доступ до рецептів. Платформа має інтуїтивно зрозумілий інтерфейс і добре структурований контент, що дозволяє користувачам легко знаходити необхідні страви за категоріями або загальним пошуком. Основна увага приділяється зручності перегляду рецептів та їх доступності для широкого кола користувачів.

Переваги:

- сучасний дизайн;
- можливість створення власних рецептів.

Недоліки:

- невелика кількість рецептів;
- відсутність рекомендаційної системи;
- обмежені можливості фільтрації.

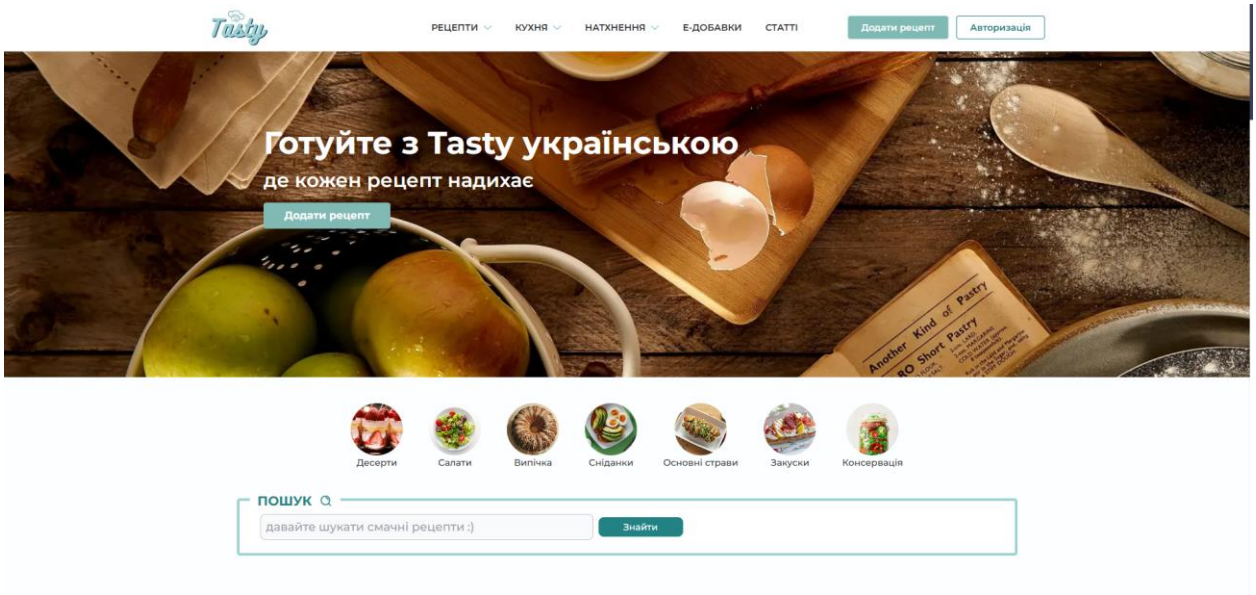

Рисунок 1.3 – Tasty

Джерело: [3]

На основі проведеного аналізу можна виділити основні недоліки існуючих аналогів:

- обмежені можливості багатокритеріальної фільтрації;
- відсутність або недостатній рівень реалізації рекомендацій;
- частковий доступ до функціоналу через платні підписки;
- незбалансованість між зручністю інтерфейсу та функціональністю;
- недостатня інтеграція соціальних можливостей у деяких сервісах.

З урахуванням виявлених недоліків було сформовано основні переваги розроблюваного вебзастосунку CookScope:

- забезпечення безкоштовного доступу до всіх основних функцій;
- реалізація багатокритеріальної фільтрації (за інгредієнтами, кухнею, дієтою, часом приготування, порціями тощо);
- використання рекомендаційної системи на основі схожості рецептів;
- наявність повноцінної системи користувацьких акаунтів (створення, редагування, підписки);
- можливість додавання власних рецептів із мультимедійним контентом;
- реалізація зручного та сучасного інтерфейсу.

Таким чином, аналіз існуючих аналогів показав, що, незважаючи на наявність великої кількості кулінарних платформ, жодна з них не поєднує у повному обсязі всі необхідні функції для ефективного пошуку, персоналізації та взаємодії користувачів. Це обґрунтовує доцільність розробки вебзастосунку CookScope, який враховує виявлені недоліки та пропонує більш комплексний підхід до організації кулінарного контенту.

1.3 Постановка задачі

Основним завданням кваліфікаційної роботи є розробка вебсайту кулінарних рецептів з використанням багатокритеріальних фільтрів та рекомендаційної системи. Даний програмний продукт призначений для забезпечення користувачів зручним, швидким та ефективним доступом до великої бази кулінарних рецептів, що дозволить значно спростити процес пошуку, підбору та використання рецептів у повсякденному житті.

Актуальність створення такої системи обумовлена тим, що більшість існуючих кулінарних ресурсів мають обмежений функціонал фільтрації, перевантажені рекламою або пропонують частину можливостей лише за платною підпискою. Це ускладнює процес пошуку рецептів, особливо коли користувач має конкретні обмеження (наприклад, наявні інгредієнти, час приготування або дієтичні вимоги). Розроблюваний програмний продукт

покликаний усунути ці недоліки шляхом впровадження зручної системи фільтрації та рекомендацій.

Система повинна забезпечувати взаємодію користувача з великою базою рецептів, отриманих із зовнішніх джерел (API або база даних), та надавати можливість швидкого доступу до релевантної інформації. Особлива увага приділяється персоналізації та адаптації результатів пошуку під індивідуальні потреби користувача.

Основні задачі розробки:

– провести аналіз предметної області та визначити ключові функції майбутньої системи;

– розробити структуру вебсайту та логіку взаємодії користувача з системою;

– реалізувати механізм пошуку рецептів за ключовими словами;

– впровадити багатокритеріальну систему фільтрації, яка дозволяє відбирати рецепти за різними параметрами (інгредієнти, час приготування, складність, кухня, калорійність тощо);

– розробити рекомендаційну систему, яка буде пропонувати користувачу релевантні варіанти на основі вибраного рецепта;

– забезпечити можливість перегляду детальної інформації про рецепт (інгредієнти, покроковий опис, зображення);

– реалізувати систему оцінювання або популярності рецептів;

– забезпечити збереження даних користувача та його взаємодії з системою (за наявності авторизації);

– створити інтуїтивно зрозумілий та адаптивний інтерфейс користувача;

– забезпечити інтеграцію з базою даних або стороннім API.

Автоматизовані функції системи:

- обробку пошукових запитів користувача;
- фільтрацію рецептів за заданими критеріями;
- сортування результатів (за популярністю, рейтингом, новизною);

- формування рекомендацій на основі вибраного рецепта;
- відображення списку рецептів у зручному форматі;
- завантаження та відображення детальної інформації про рецепт;
- обробку взаємодії користувача (перегляди, кліки, оцінки);
- оновлення даних у режимі реального часу (за потреби).

Вхідні дані:

1) текстові запити користувача (ключові слова);

2) параметри фільтрації:

а) інгредієнти;

б) час приготування;

в) складність;

г) тип кухні;

д) калорійність або дієтичні обмеження;

3) дані про взаємодію користувача:

а) історія переглядів;

б) обрані рецепти;

в) оцінки (за наявності);

4) дані, отримані із зовнішніх джерел (API або база даних).

Вихідні дані:

1) список рецептів, що відповідають запиту користувача;

2) відфільтровані результати відповідно до заданих критеріїв;

3) рекомендовані рецепти;

4) детальна інформація про рецепт:

а) назва;

б) інгредієнти;

в) спосіб приготування;

г) зображення;

д) додаткова інформація;

5) список популярних або рейтингових рецептів.

Функціональні вимоги:

– можливість пошуку рецептів за ключовими словами;

– підтримка багатокритеріальної фільтрації;

– формування рекомендацій на основі вибраного рецепта;

– перегляд детальної інформації про рецепт;

– відображення списку рецептів у вигляді зручного каталогу;

– сортування результатів пошуку;

– можливість оцінювання або взаємодії з контентом;

– інтеграція з базою даних або API;

Нефункціональні вимоги:

– продуктивність – швидка обробка запитів та відображення результатів;

– зручність використання – інтуїтивно зрозумілий інтерфейс;

– надійність – стабільна робота без збоїв;

– масштабованість – можливість роботи з великою кількістю рецептів;

– адаптивність – коректне відображення на різних пристроях;

– безпека – захист даних користувачів;

– сумісність – робота у сучасних браузерах;

– розширюваність – можливість додавання нових функцій у майбутньому.

Висновки до розділу 1

У першому розділі було розглянуто предметну область вебзастосунків для роботи з кулінарними рецептами та визначено основні проблеми, пов’язані з пошуком і відбором інформації. Встановлено, що існуючі рішення мають обмежені можливості фільтрації, недостатній рівень персоналізації та не завжди забезпечують зручну взаємодію користувача із системою.

Проведений аналіз аналогів, зокрема Cookpad, Shuba та Tasty, дозволив виявити їхні переваги та недоліки, а також підтвердити відсутність

комплексного підходу до реалізації функціоналу пошуку, фільтрації та рекомендацій.

На основі отриманих результатів було сформульовано постановку задачі, визначено основні функціональні можливості, вхідні та вихідні дані, а також вимоги до програмного продукту.

Таким чином, результати розділу створюють основу для подальшого проєктування вебзастосунку у наступному розділі.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ КАТАЛОГУ РЕЦЕПТІВ З БАГАТОКРИТЕРІАЛЬНОЮ ФІЛЬТРАЦІЄЮ ТА РЕКОМЕНДАЦІЙНОЮ СИСТЕМОЮ

2.1 Моделювання поведінки продукту

На етапі проєктування програмного продукту одним із ключових завдань є визначення поведінки системи, що дозволяє описати процеси взаємодії користувача з вебзастосунком, а також внутрішню логіку обробки запитів. Моделювання поведінки продукту забезпечує формалізацію функціональних можливостей системи та створює основу для її подальшої реалізації [4].

Розроблюваний вебзастосунок CookScope призначений для організації процесу пошуку, відбору та перегляду кулінарних рецептів із використанням багатокритеріальної фільтрації та рекомендаційної системи. У рамках даної системи передбачається взаємодія користувача з вебінтерфейсом, серверною частиною та базою даних, що забезпечує зберігання та обробку інформації.

Моделювання поведінки продукту передбачає визначення основних сценаріїв використання, які описують типові дії користувача:

- пошук рецептів за ключовими словами;
- застосування фільтрів для уточнення результатів;
- перегляд списку рецептів;
- перегляд детальної інформації про рецепт;
- отримання рекомендацій;
- додавання рецепту до обраного;
- оцінювання рецепту;
- створення та редагування власних рецептів;
- перегляд профілів користувачів та підписка на них.

Кожен із зазначених сценаріїв відображає окремий функціональний аспект системи та є складовою загальної логіки її роботи.

Для формалізації взаємодії користувачів із системою використовується діаграма варіантів використання (рис 2.1) яка є складовою мови моделювання UML. Візуалізація діаграм у роботі виконана за допомогою інструменту PlantUML [5] Вона дозволяє визначити функціональні можливості системи та відобразити зв’язки між акторами і варіантами використання.

Основними варіантами використання є:

- пошук рецепту;
- фільтрація рецептів;
- перегляд рецепту;
- авторизація користувача;
- створення рецепту;
- редагування рецепту;
- додавання до обраного;
- оцінювання рецепту;
- отримання рекомендацій.

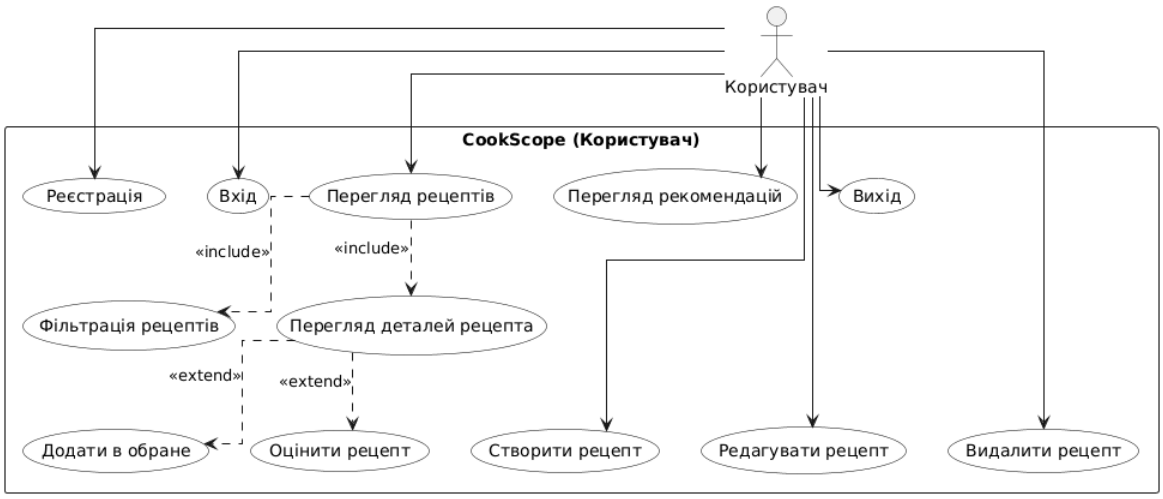

Рисунок 2.1 – діаграма варіантів використання

Джерело: розроблено автором

Взаємозв’язки між варіантами використання можуть включати відношення типу «include» (наприклад, пошук включає фільтрацію) та

«extend» (наприклад, перегляд рецепту може доповнюватися рекомендаціями).

Діаграма діяльності (рис 2.2) використовується для опису послідовності дій під час виконання основних операцій у системі. Одним із ключових процесів є пошук рецепту, який включає наступні етапи:

1) введення користувачем пошукового запиту або вибір параметрів фільтрації;
2) передача запиту на сервер;
3) обробка запиту та формування SQL-запиту до бази даних;
4) отримання результатів;
5) застосування сортування;
6) формування списку рецептів;
7) відображення результатів користувачу.

У випадку відсутності результатів система повинна повідомити користувача та запропонувати змінити параметри пошуку.

Важливою особливістю даного процесу є те, що система повинна забезпечувати не лише коректне виконання пошуку, а й зручну взаємодію з користувачем на кожному етапі. Зокрема, після отримання результатів пошуку користувач повинен мати можливість швидко перейти до перегляду детальної інформації про обраний рецепт, ознайомитися зі списком інгредієнтів, етапами приготування, часом приготування та іншими характеристиками. Крім того, на цьому етапі система може формувати список рекомендованих рецептів, що є схожими за складом інгредієнтів, типом страви або іншими параметрами. Такий підхід дозволяє не лише спростити пошук необхідної страви, а й підвищити зручність користування вебзастосунком у цілому.

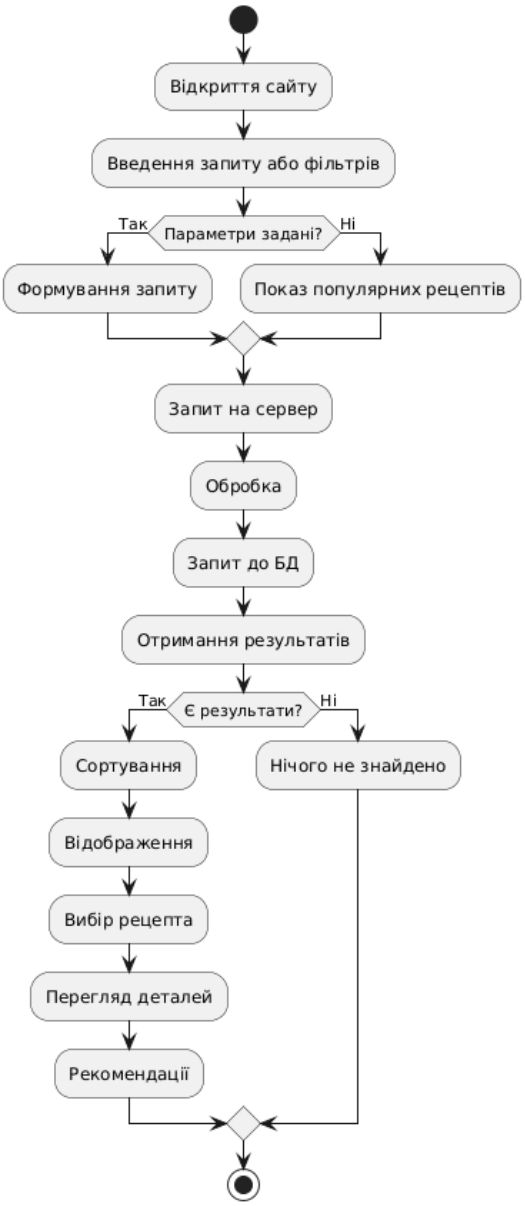

Рисунок 2.2 – діаграма діяльності

Джерело: розроблено автором

Для детального опису взаємодії між компонентами системи використовується діаграма послідовності (рис 2.3). Вона демонструє порядок обміну повідомленнями між клієнтською частиною (frontend), сервером (backend) та базою даних.

Типовий сценарій взаємодії включає наступні кроки:

– користувач надсилає запит через інтерфейс;

– клієнтська частина формує HTTP-запит;

– сервер приймає запит та передає його до відповідного модуля обробки;

– виконується звернення до бази даних;

– отримані дані повертаються на сервер;

– сервер формує відповідь та надсилає її клієнту;

– результати відображаються у вебінтерфейсі.

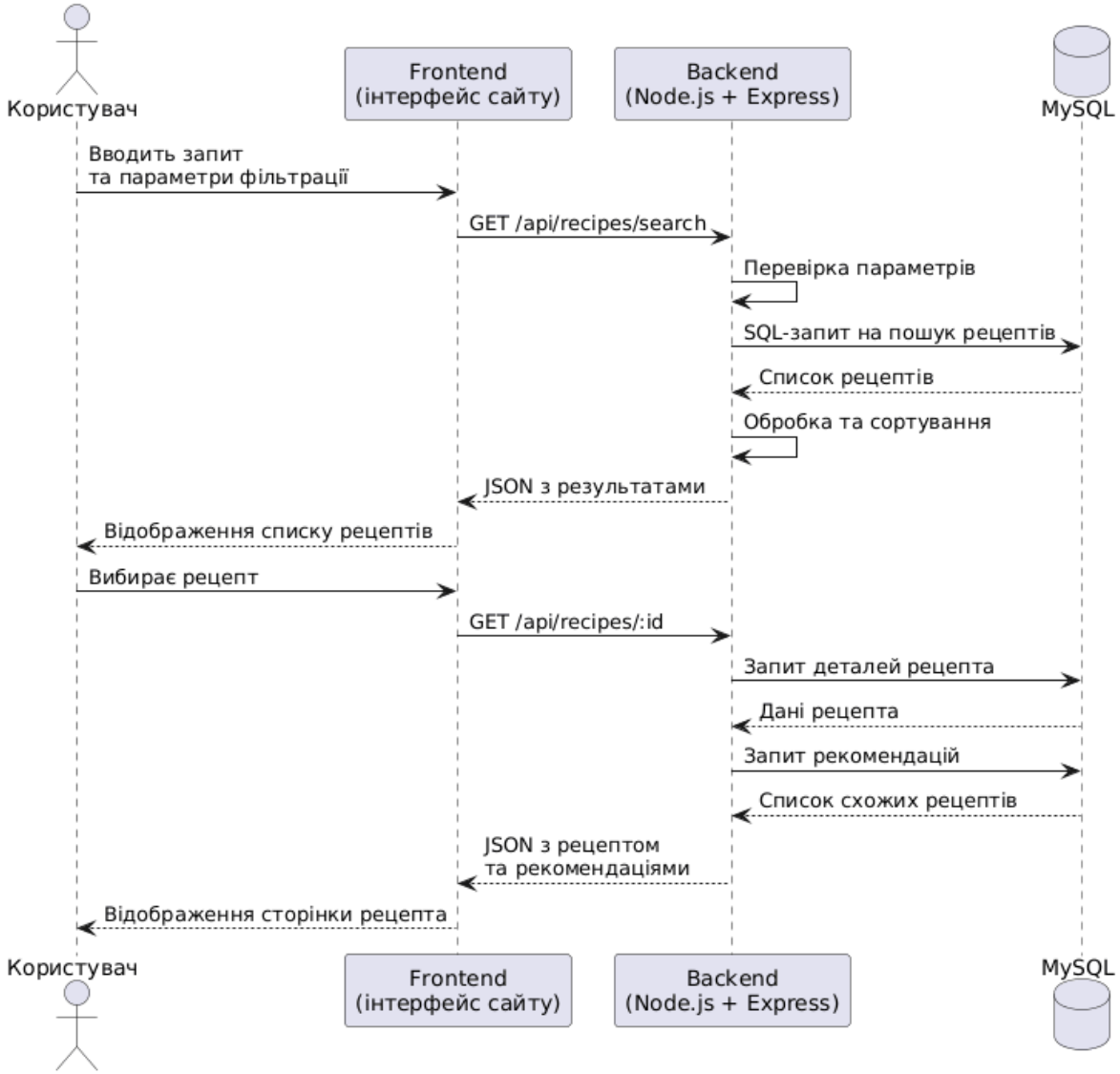

Рисунок 2.3 – діаграма послідовності

Джерело: розроблено автором

Важливим аспектом моделювання поведінки є опис логіки обробки даних. У системі реалізується багатокритеріальний підхід до фільтрації, що передбачає обробку декількох параметрів одночасно. Кожен параметр впливає на формування кінцевого результату, що дозволяє отримати більш точний список рецептів.

Рекомендаційна система функціонує на основі аналізу характеристик рецептів, таких як інгредієнти, тип страви та інші параметри[6]. Це дозволяє визначити схожі рецепти та запропонувати їх користувачу.

2.2 Моделювання архітектури та структури продукту

На етапі проєктування програмного продукту важливим є визначення структури системи, яка описує склад її компонентів на рівні даних та взаємозв’язки між ними. Моделювання структури продукту дозволяє сформувати цілісне уявлення про організацію зберігання інформації, забезпечити цілісність даних та створити основу для подальшої реалізації програмного забезпечення.

У межах розроблюваного вебзастосунку CookScope особлива увага приділяється проєктуванню бази даних, оскільки саме вона забезпечує зберігання всієї інформації про користувачів, рецепти та їх взаємодію. Для цього використовується реляційна модель даних, яка дозволяє чітко визначити структуру таблиць та зв’язки між ними.

Для моделювання структури програмного продукту використовується діаграма класів (рис. 2.4), яка дозволяє відобразити основні об’єкти системи, їх атрибути та взаємозв’язки між ними. Така діаграма є важливою частиною проєктування, оскільки показує логічну структуру вебзастосунку та допомагає зрозуміти, які компоненти беруть участь реалізації основного функціоналу.

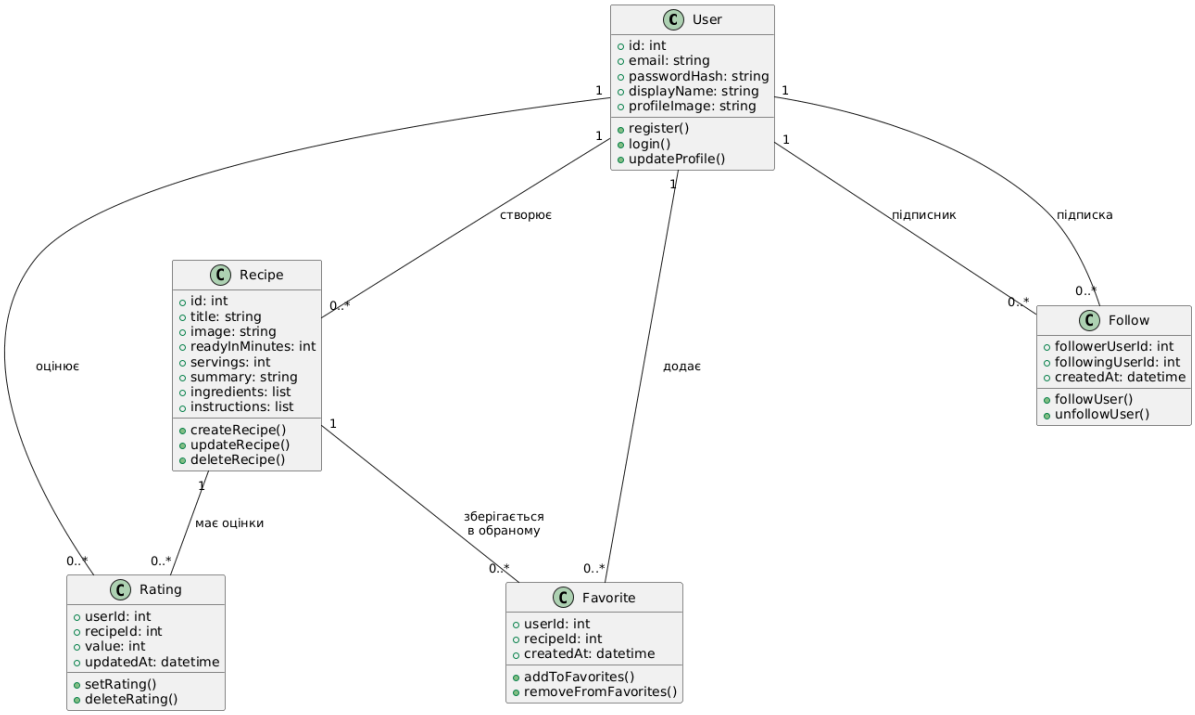

Рисунок 2.4 – Діаграма класів

Джерело: розроблено автором

У межах вебзастосунку CookScope основними класами є User, Recipe, Rating, Favorite та Follow. Клас User описує користувача системи, клас Recipe відповідає за інформацію про рецепт, клас Rating використовується для оцінювання рецептів, клас Favorite забезпечує додавання рецептів до обраного, а клас Follow реалізує підписки між користувачами.

Зв’язки між класами демонструють взаємодію основних об’єктів системи. Один користувач може створювати декілька рецептів, оцінювати рецепти, додавати їх до обраного та підписуватись на інших користувачів. Рецепт, у свою чергу, може мати багато оцінок і бути доданим до обраного різними користувачами.

На етапі проєктування програмного продукту важливим є визначення архітектури системи, яка описує загальну організацію програмного забезпечення, взаємодію його компонентів та принципи обробки даних.

Архітектура програмного продукту визначає спосіб побудови системи, розподіл функцій між її складовими та механізми взаємодії між ними.

Розроблюваний вебзастосунок CookScope побудований за принципом клієнт-серверної архітектури, яка є однією з найбільш поширених моделей у сучасних вебсистемах. Такий підхід передбачає розподіл системи на декілька рівнів, кожен з яких виконує власні функції та взаємодіє з іншими через визначені інтерфейси.

Архітектура вебзастосунку складається з трьох основних рівнів:

- клієнтський рівень (frontend);
- серверний рівень (backend);
- рівень зберігання даних (database).

Кожен із цих рівнів виконує окрему роль у системі, що дозволяє забезпечити розподіл відповідальності та підвищити ефективність роботи програмного продукту.

Клієнтська частина реалізована з використанням HTML, CSS та JavaScript і відповідає за взаємодію з користувачем. Вона забезпечує відображення інтерфейсу, обробку дій користувача, формування запитів до серверної частини та відображення отриманих результатів.

Основними функціями клієнтської частини є:

- відображення каталогу рецептів;
- реалізація пошуку та фільтрації;
- відображення детальної інформації про рецепти;
- взаємодія з функціями обраного та оцінювання;
- робота з обліковим записом користувача.

Клієнтська частина не містить складної бізнес-логіки, оскільки основні обчислення виконуються на сервері.

Серверна частина реалізована на платформі Node.js із використанням фреймворку Express та виконує функції обробки запитів, реалізації бізнес-логіки та взаємодії з базою даних.

Сервер приймає HTTP-запити від клієнта, обробляє їх, виконує необхідні операції та повертає результат у форматі JSON.

Основні функції серверної частини:

- обробка запитів користувача;
- реалізація логіки фільтрації та пошуку рецептів;
- обробка операцій створення, редагування та видалення рецептів;
- управління користувачами та їх авторизацією;
- формування рекомендацій;
- взаємодія з базою даних.

Серверна частина побудована за модульним принципом, що дозволяє розділити функціональність на окремі компоненти (маршрути, сервіси тощо).

Рівень зберігання даних представлений реляційною базою даних MySQL, яка забезпечує зберігання інформації про користувачів, рецепти, оцінки, обране та підписки.

База даних використовується для:

- збереження інформації;
- обробки запитів;
- забезпечення цілісності даних;
- підтримки зв’язків між сутностями.

Взаємодія серверної частини з базою даних здійснюється через SQL-запити.

Взаємодія між клієнтською та серверною частинами здійснюється через REST API, що наочно представлено на рисунку 2.5. Клієнт формує HTTP-запит, який передається на сервер. Сервер обробляє запит, звертається до бази даних, отримує необхідні дані та повертає відповідь клієнту.

Основні типи запитів:

- GET – для отримання даних;
- POST – для створення записів;
- PATCH – для оновлення даних;

– DELETE – для видалення записів.

Такий підхід дозволяє забезпечити стандартизовану взаємодію між компонентами системи та спрощує її масштабування.

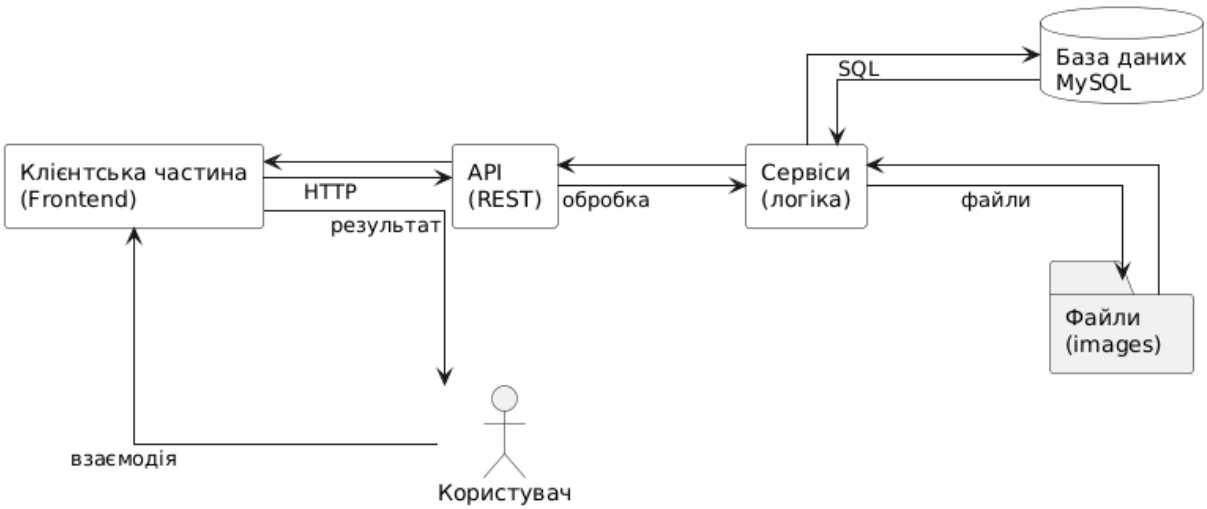

Рисунок 2.5 – Архітектурна діаграма

Джерело: розроблено автором

Як показано на рисунку 2.5, клієнтська частина взаємодіє із сервером через HTTP-запити, а сервер, у свою чергу, здійснює обробку запитів та взаємодію з базою даних.

Обрана архітектура має ряд переваг:

– чіткий розподіл функцій між компонентами системи;
– можливість масштабування;
– зручність у розробці та підтримці;
– незалежність клієнтської та серверної частин;
– можливість розширення функціональності.

2.3 Опис алгоритмів та математичних моделей

У розробленому вебзастосунку CookScope реалізовано ряд алгоритмів, що забезпечують виконання основних функціональних можливостей системи, зокрема пошук рецептів, їх фільтрацію, сортування та формування

рекомендацій. Застосовані алгоритми спрямовані на ефективну обробку даних та забезпечення швидкої відповіді користувачу.

Одним із ключових алгоритмів системи є алгоритм пошуку рецептів. Його робота полягає у порівнянні введеного користувачем запиту з назвами рецептів та їх описами. Для цього використовується перевірка на входження підрядка, що дозволяє знаходити рецепти навіть при частковому співпадінні введеного тексту.

Алгоритм пошуку можна представити у вигляді наступних кроків:

1) отримання пошукового запиту від користувача;
2) нормалізація тексту (переведення до одного регістру);
3) перебір списку рецептів;
4) перевірка входження запиту в назву або опис рецепта;
5) формування списку результатів;
6) передача результатів на відображення.

Важливим компонентом системи є алгоритм багатокритеріальної фільтрації, який дозволяє відбирати рецепти відповідно до заданих параметрів[7]. Фільтрація виконується за такими критеріями, як інгредієнти, тип страви, кухня, дієта та час приготування.

Алгоритм фільтрації передбачає послідовну перевірку кожного рецепта на відповідність заданим умовам. Якщо хоча б один із критеріїв не виконується, рецепт виключається зі списку результатів. Таким чином формується підмножина рецептів, що задовольняє всі обрані параметри.

Формально процес фільтрації можна описати як послідовне застосування логічних умов до множини рецептів:

$$R_{filtered} = \{r \in R | \, f_1(r) \wedge f_2(r) \wedge \cdots \wedge f_n(r)\}$$

де:

- $R_{filtered}$ – відфільтрована множина рецептів;
- R – множина всіх рецептів;

– $f_i(r)$ – умова відповідності рецепта одному з критеріїв фільтрації.

Для сортування результатів використовується алгоритм впорядкування за рейтингом. У системі застосовується формула IMDb[8], яка дозволяє враховувати як середню оцінку рецепта, так і кількість голосів, що підвищує об’єктивність ранжування.

Формула рейтингу має вигляд:

$$W = \frac{v}{v+m} \cdot R + \frac{m}{v+m} \cdot C$$

де:

– W – фінальний рейтинг рецепта;

– R – середня оцінка;

– v – кількість голосів;

– m – мінімальна кількість голосів, необхідна для врахування рейтингу;

– C – середній рейтинг по системі.

Дана формула дозволяє уникнути ситуацій, коли рецепти з малою кількістю оцінок отримують завищений рейтинг, та забезпечує більш справедливе ранжування результатів.

У системі реалізовано алгоритм рекомендацій, який базується на принципі схожості рецептів. Такий підхід дозволяє формувати список рекомендованих рецептів на основі характеристик поточного рецепта без урахування персональних даних користувача.

Рекомендації формуються шляхом порівняння рецептів за набором атрибутів, зокрема:

- тип страви;
- кухня;
- інгредієнти;
- дієтичні характеристики.

Алгоритм передбачає визначення ступеня схожості між поточним рецептом та іншими рецептами у системі. Чим більше спільних характеристик мають рецепти, тим вищою є їхня схожість.

З математичної точки зору, схожість між двома рецептами може бути представлена як:

$$S(r_1, r_2) = \sum_{i=1}^{n} w_i \cdot match_i$$

де:

- $S(r_1, r_2)$ – міра схожості між двома рецептами;
- w_i – ваговий коефіцієнт ознаки;
- $match_i$ – показник збігу за і-ю характеристикою (1 — збіг, 0 — відсутність збігу).

На основі отриманого значення схожості формується список найбільш релевантних рецептів, які пропонуються користувачу як рекомендації.

Такий підхід є простим у реалізації, не потребує збереження історії дій користувача та забезпечує достатньо релевантні результати при роботі з кулінарними даними.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи було виконано проєктування вебзастосунку каталогу рецептів CookScope. На даному етапі визначено поведінку системи, її структуру, архітектуру, а також описано основні алгоритми обробки даних.

У процесі моделювання поведінки продукту було сформовано ключові сценарії взаємодії користувача із системою, що включають пошук рецептів, застосування багатокритеріальної фільтрації, перегляд детальної інформації, а також взаємодію з функціями обраного, оцінювання та рекомендацій. Для візуалізації поведінки використано діаграми UML, що дозволило

формалізувати логіку роботи системи та визначити послідовність виконання основних операцій.

У межах моделювання структури продукту було побудовано діаграму класів, яка відображає основні сутності системи та зв’язки між ними. Визначено ключові класи, такі як користувач, рецепт, оцінка, обране та підписки, що формують основу інформаційної моделі системи. Це дозволило забезпечити цілісність даних та зрозумілу організацію взаємодії між об’єктами.

Також у розділі обґрунтовано вибір клієнт-серверної архітектури, яка передбачає розподіл системи на клієнтський, серверний рівні та рівень зберігання даних. Визначено функції кожного рівня, а також принципи їх взаємодії через REST API, що забезпечує гнучкість, масштабованість та зручність розробки програмного продукту.

Окрему увагу приділено опису алгоритмів, що реалізують основний функціонал системи. Розглянуто алгоритми пошуку, багатокритеріальної фільтрації та сортування результатів, зокрема із використанням формули IMDb для визначення рейтингу рецептів. Також описано алгоритм рекомендацій, який базується на принципі схожості рецептів та дозволяє формувати релевантні пропозиції без використання персоналізованих даних користувача.

Таким чином, у другому розділі сформовано концептуальну модель вебзастосунку, яка охоплює його поведінку, структуру, архітектуру та алгоритмічну основу. Отримані результати створюють необхідну базу для подальшої реалізації програмного продукту, що розглядається у наступному розділі.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ КАТАЛОГУ РЕЦЕПТІВ З БАГАТОКРИТЕРІАЛЬНОЮ ФІЛЬТРАЦІЄЮ ТА РЕКОМЕНДАЦІЙНОЮ СИСТЕМОЮ

3.1 Реалізація та конструювання програмного продукту

Клієнтська частина вебзастосунку CookScope реалізована з використанням технологій HTML, CSS та JavaScript і забезпечує взаємодію користувача із системою. Основним призначенням клієнтської частини є відображення інтерфейсу, обробка дій користувача, формування запитів до серверної частини та візуалізація отриманих результатів.

Розробка клієнтської частини здійснювалась із урахуванням принципів зручності використання, доступності та адаптивності. Особлива увага приділялась логічній структурі сторінок, мінімізації кількості дій для досягнення результату та швидкості взаємодії з інтерфейсом.

Клієнтська частина організована у вигляді набору HTML-сторінок, кожна з яких відповідає за окремий функціональний модуль системи. Такий підхід дозволяє розділити функціональність та спростити підтримку проєкту.

Основні сторінки вебзастосунку:

- головна сторінка (index.html), яка відображає популярні рецепти та основну інформацію;
- сторінка каталогу рецептів (catalog.html), що забезпечує пошук і фільтрацію;
- сторінка рецепта (recipe.html), де відображається детальна інформація;
- сторінка обраного (favorites.html);
- сторінка профілю користувача (account.html);
- сторінки користувачів та підписок.

Кожна сторінка взаємодіє з відповідними JavaScript-файлами, що реалізують клієнтську логіку.

Клієнтська логіка реалізована за допомогою JavaScript без використання додаткових фреймворків[9]. Це дозволило зменшити складність проєкту та забезпечити повний контроль над реалізацією функціональності.

Основні задачі клієнтської логіки:

- обробка подій користувача (натискання кнопок, введення даних);
- формування HTTP-запитів до серверної частини;
- обробка відповідей у форматі JSON;
- динамічне оновлення DOM-структури сторінки;
- управління станом інтерфейсу.

Взаємодія клієнтської частини з сервером реалізована за допомогою HTTP-запитів із використанням функції fetch. Приклад реалізації такого запиту наведено на рисунку 3.1.

```
const Api = (() => {
  const request = async (url, params = {}, options = {}) => {
    const method = options.method || 'GET';
    const query = new URLSearchParams();

    Object.entries(params).forEach(([key, value]) => {
      if (value !== undefined && value !== null && value !== '') {
        query.append(key, value);
      }
    });

    const targetUrl = method === 'GET' ? `${url}?${query.toString()}`.replace(/\?$/, '') : url;
    const response = await fetch(targetUrl, {
      method,
      headers: {
        'Content-Type': 'application/json',
        ...(options.headers || {})
      },
      body: options.body ? JSON.stringify(options.body) : undefined
    });
```

Рисунок 3.1 – Приклад реалізації HTTP-запиту до серверної частини з використанням JavaScript

Джерело: розроблено автором

Взаємодія з серверною частиною здійснюється за допомогою REST API. Для цього використовуються стандартні методи HTTP, зокрема GET, POST,

PATCH та DELETE. Дані передаються у форматі JSON, що забезпечує зручність їх обробки.

Інтерфейс користувача реалізовано за допомогою HTML та CSS. HTML використовується для створення структури сторінок, тоді як CSS забезпечує стилізацію та візуальне оформлення. Приклад реалізації інтерфейсу головної сторінки наведено на рисунку 3.2.

Особливості інтерфейсу:

– логічне групування елементів;
– використання форм для введення даних;
– адаптивне відображення елементів;
– використання списків для відображення рецептів;
– наявність інтерактивних елементів.

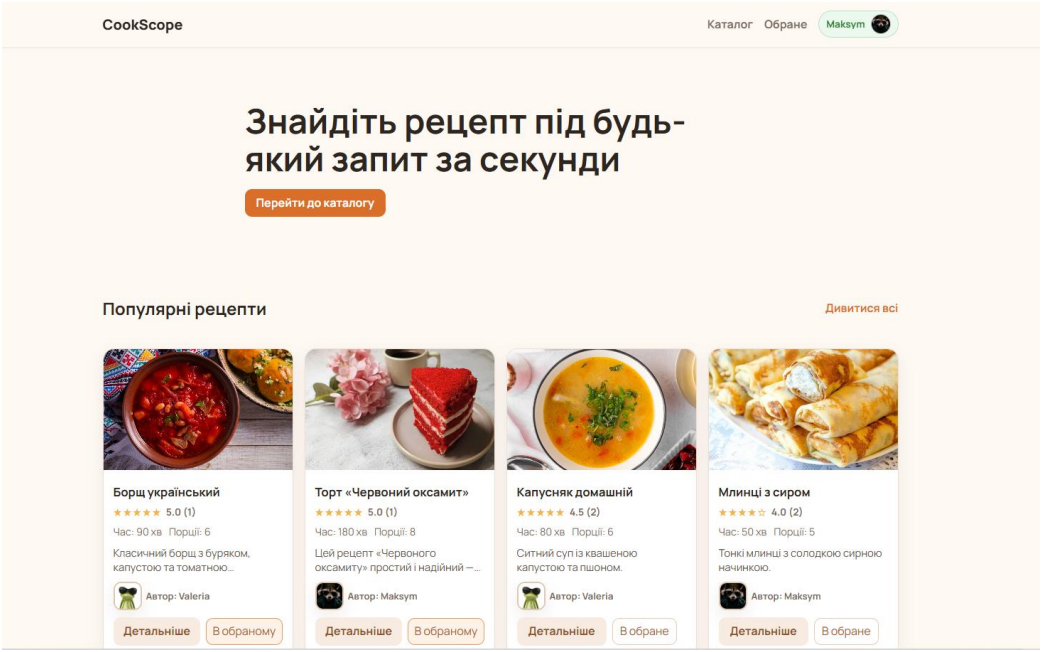

Рисунок 3.2 – Головна сторінка сайту

Джерело: розроблено автором

Функціональність пошуку та фільтрації є ключовою для вебзастосунку. Користувач може вводити пошуковий запит та задавати додаткові параметри, що дозволяє звузити результати.

Алгоритм роботи:

1) Користувач вводить дані у форму;
2) JavaScript формує запит;
3) запит надсилається на сервер;
4) сервер повертає результат;
5) клієнт відображає отримані дані.

Основна логіка багатокритеріальної фільтрації реалізована у функції (рис. 3.3), яка обробляє параметри запиту користувача та виконує послідовну перевірку рецептів за заданими критеріями.

```
342  const filterRecipes = (recipes, queryParams) => {
343    const {
344      title,
345      queryText,
346      includeIngredients,
347      excludeIngredients,
348      type,
349      cuisine,
350      diet,
351      maxReadyTime,
352      minServings,
353      maxServings
354    } = queryParams;
355
356    const titleValue = title || queryText;
357    const includeList = parseQueryList(includeIngredients);
358    const excludeList = parseQueryList(excludeIngredients);
359    const types = parseQueryList(type);
360    const cuisines = parseQueryList(cuisine);
361    const diets = parseQueryList(diet);
362
363    return recipes.filter((recipe) => {
364      if (titleValue && !includesToken(recipe.title, titleValue)) return false;
365      if (types.length && !types.some((item) => listIncludesToken(recipe.dishTypes, item))) return false;
366      if (cuisines.length && !cuisines.some((item) => listIncludesToken(recipe.cuisines, item))) return false;
367      if (diets.length && !diets.some((item) => listIncludesToken(recipe.diets, item))) return false;
368
369      if (maxReadyTime && recipe.readyInMinutes !== null && recipe.readyInMinutes > Number(maxReadyTime)) {
370        return false;
371      }
```

Рисунок 3.3 – Фрагмент коду реалізації фільтрації

Джерело: розроблено автором

Для покращення користувацького досвіду використовується динамічне оновлення сторінок без повного перезавантаження. Це досягається шляхом маніпуляції DOM через JavaScript. приклад відповідного фрагменту коду наведено на рисунку 3.4.

Основні переваги такого підходу:

– зменшення часу відгуку;

– підвищення зручності користування;

– зменшення навантаження на сервер.

```
265  const loadRecipes = async () => {
266    loader.classList.remove('hidden');
267    empty.classList.add('hidden');
268
269    try {
270      const params = {
271        ...state.filters,
272        number: state.limit,
273        offset: (state.page - 1) * state.limit
274      };
275
276      const data = await Api.searchRecipes(params);
277      state.totalResults = data.totalResults || 0;
278
279      if (!data.results?.length) {
280        grid.innerHTML = '';
281        empty.classList.remove('hidden');
282      } else {
283        UI.renderCards(grid, data.results);
284      }
285
286      syncPagination();
287    } catch (error) {
288      grid.innerHTML = '';
289      empty.textContent = error.message;
290      empty.classList.remove('hidden');
291    } finally {
292      loader.classList.add('hidden');
293    }
294  };
```

Рисунок 3.4 – Фрагмент коду динамічного оновлення інтерфейсу

Джерело: розроблено автором

Клієнтська частина реалізує функціональність, пов’язану з обліковим записом користувача:

– реєстрація;

– авторизація;

– редагування профілю;

– управління рецептами;

– додавання до обраного.

Як показано на рисунку 3.5, система виконує перевірку введених користувачем даних, після чого здійснює запит до серверної частини для реєстрації або входу в систему та зберігає отримані дані сесії.

Для збереження стану користувача використовується токен, який передається разом із запитами до серверної частини.

```
1681  const handleSubmit = async (event) => {
1682    event.preventDefault();
1683
1684    const displayName = authDisplayName?.value?.trim() || '';
1685    const email = authEmail?.value?.trim();
1686    const password = authPassword?.value || '';
1687    const confirm = authPasswordConfirm?.value || '';
1688
1689    if (!email || !password) return;
1690    if (state.mode === MODE.REGISTER && !displayName) return;
1691
1692    if (state.mode === MODE.REGISTER && password !== confirm) {
1693      showFeedback('Паролі не співпадають.', true);
1694      return;
1695    }
1696
1697    showFeedback(state.mode === MODE.REGISTER ? 'Реєстрація...' : 'Вхід...');
1698
1699    try {
1700      const payload =
1701        state.mode === MODE.REGISTER
1702          ? await Api.register(email, password, displayName)
1703          : await Api.login(email, password);
1704
1705      saveSession(payload);
1706      updateUi();
```

Рисунок 3.5 – Фрагмент коду реалізації авторизації користувача

Джерело: розроблено автором

Серверна частина вебзастосунку CookScope реалізована з використанням платформи Node.js та фреймворку Express і виконує ключову роль у забезпеченні функціонування системи. Вона відповідає за обробку запитів від клієнтської частини, реалізацію бізнес-логіки, взаємодію з базою даних та формування відповідей у форматі JSON.

Архітектура серверної частини побудована за модульним принципом, що передбачає розподіл функціональності між окремими компонентами. Такий підхід дозволяє підвищити гнучкість системи, спростити її

масштабування та забезпечити зручність супроводу програмного коду. Основними складовими серверної частини є модулі обробки маршрутів, сервіси бізнес-логіки та компоненти взаємодії з базою даних.

Ініціалізація серверної частини здійснюється у головному файлі, де виконується налаштування середовища виконання, підключення необхідних бібліотек, а також реєстрація маршрутів (рис 3.6). Сервер обробляє HTTP-запити, що надходять від клієнта, та передає їх до відповідних обробників залежно від типу запиту та адреси.

Взаємодія між клієнтською та серверною частинами реалізована за допомогою REST API, що забезпечує стандартизований спосіб обміну даними. Сервер приймає запити, обробляє їх та повертає результат у структурованому вигляді, що дозволяє клієнтській частині ефективно відображати отриману інформацію.

Особливістю реалізації є чітке розділення відповідальності між компонентами. Маршрути відповідають за прийом та перенаправлення запитів, тоді як основна логіка обробки даних реалізована у сервісах. Це дозволяє уникнути дублювання коду та забезпечує централізовану обробку функціональності.

```
require('dotenv').config();

const express = require('express');
const cors = require('cors');
const path = require('path');
const recipeRoutes = require('../routes/recipesRoutes');
const authRoutes = require('../routes/authRoutes');
const favoritesRoutes = require('../routes/favoritesRoutes');
const usersRoutes = require('../routes/usersRoutes');
const { initializeDatabase } = require('../services/dbInit');
const { closePool } = require('../services/db');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(express.json({ limit: '12mb' }));

app.use('/api/recipes', recipeRoutes);
app.use('/api/auth', authRoutes);
app.use('/api/favorites', favoritesRoutes);
app.use('/api/users', usersRoutes);

app.use('/assets', express.static(path.join(__dirname, '..', 'frontend')));
app.use(express.static(path.join(__dirname, '..', 'public')));
```

Рисунок 3.6 – Організація серверної частини вебзастосунку

Джерело: розроблено автором

Окрему роль у роботі серверної частини відіграє взаємодія з базою даних. Для зберігання інформації використовується реляційна база даних MySQL, яка забезпечує структуроване зберігання даних та підтримку зв’язків між сутностями. Сервер формує SQL-запити для отримання, оновлення та видалення інформації, що дозволяє реалізувати повний цикл роботи з даними. Приклад взаємодії серверної частини з базою даних під час авторизації користувача наведено на рисунку 3.7.

```
121 const loginUser = async (email, password) => {
122   const normalizedEmail = normalizeEmail(email);
123   const rows = await query('SELECT * FROM users WHERE email = ? LIMIT 1', [normalizedEmail]);
124   const user = mapUserRow(rows[0]);
125
126   if (!user || !verifyPassword(password, user.passwordHash)) {
127     const error = new Error('Невірний email або пароль');
128     error.statusCode = 401;
129     throw error;
130   }
131
132   const token = createToken();
133   await execute('UPDATE users SET token = ? WHERE id = ?', [token, user.id]);
134
135   return {
136     user: toPublicUser({ ...user, token }),
137     token
138   };
139 };
```

Рисунок 3.7 – Приклад взаємодії серверної частини з базою даних

Джерело: розроблено автором

Для забезпечення надійності роботи системи передбачено обробку помилок, яка дозволяє коректно реагувати на некоректні запити або збої під час виконання операцій. У разі виникнення помилки сервер формує відповідне повідомлення та повертає його клієнтській частині, що забезпечує зворотний зв’язок із користувачем та підвищує стабільність роботи системи.

Таким чином, серверна частина вебзастосунку забезпечує повний цикл обробки запитів, починаючи від їх отримання та закінчуючи формуванням відповіді. Використання сучасних технологій та модульного підходу дозволяє створити ефективну, гнучку та масштабовану систему, яка здатна обробляти значну кількість запитів та забезпечувати стабільну роботу вебзастосунку.

3.2 Тестування програмного продукту

Тестування є важливим етапом розробки програмного продукту, оскільки дозволяє перевірити коректність роботи основних функцій системи, виявити помилки та оцінити відповідність реалізованого функціоналу поставленим вимогам [10]. У межах роботи було проведено функціональне

тестування вебзастосунку CookScope, основною метою якого є перевірка працездатності ключових можливостей системи.

Під час тестування перевірялись основні сценарії взаємодії користувача з вебзастосунком: реєстрація, авторизація, пошук рецептів, застосування фільтрів, перегляд детальної інформації про рецепт, додавання рецепта до обраного, оцінювання рецепта, створення власного рецепта та редагування профілю користувача.

Тестування виконувалось вручну шляхом послідовного виконання тестових сценаріїв у вебінтерфейсі системи. Для кожного сценарію визначались вхідні дані, очікуваний результат та фактичний результат виконання.

Таблиця 3.1 – Тест-кейси для перевірки вебзастосунку

№	*Назва тесту*	*Дії користувача*	*Очікуваний результат*	*Статус*
1	Реєстрація користувача	Ввести ім’я, email, пароль та підтвердження пароля	Створюється новий обліковий запис, користувач входить у систему	Пройдено
2	Авторизація користувача	Ввести правильний email та пароль	Користувач успішно входить в систему	Пройдено
3	Авторизація з неправильним паролем	Ввести email і неправильний пароль	Система виводить повідомлення про помилку	Пройдено
4	Пошук рецепта за назвою	Ввести назву рецепта в поле пошуку	Відображаються рецепти що відповідають запиту	Пройдено
5	Фільтрація рецептів	Обрати параметри фільтрації	Відображаються лише рецепти що відповідають заданим критеріям	Пройдено
6	Перегляд рецептів	Натиснути на картку рецепта	Відкривається сторінка з детальною інформацією про рецепт	Пройдено
7	Додавання до обраного	Натиснути на кнопку додавання до обраного	Рецепт додається до списку обраних	Пройдено
8	Видалення з обраного	Натиснути кнопку видалення з обраного	Рецепт видаляється зі списку обраних	Пройдено

За результатами тестування встановлено, що основні функції вебзастосунку працюють коректно та відповідають поставленим вимогам. Особливу увагу було приділено перевірці пошуку та багатокритеріальної фільтрації, оскільки ці функції є ключовими для розроблюваної системи.

Під час тестування також перевірялась реакція системи на некоректні дії користувача, зокрема введення неправильних даних під час авторизації або пошук рецептів за запитом, який не має результатів. У таких випадках система коректно повідомляє користувача про помилку або відсутність знайдених рецептів.

Таким чином, проведене тестування підтвердило працездатність основних функціональних можливостей вебзастосунку CookScope та його готовність до подальшого вдосконалення.

3.3 Можливості використання програмного продукту

Розроблений вебзастосунок CookScope призначений для автоматизації процесу пошуку, перегляду та управління кулінарними рецептами і може бути використаний широким колом користувачів. Завдяки реалізації сучасного інтерфейсу та функціональних можливостей система забезпечує зручну взаємодію з користувачем та ефективне виконання основних операцій.

Основним призначенням програмного продукту є надання користувачеві інструментів для швидкого підбору рецептів відповідно до заданих критеріїв. Користувач має можливість здійснювати пошук за назвою рецепта, а також використовувати багатокритеріальну фільтрацію, яка враховує такі параметри, як інгредієнти, тип страви, кухня, дієта та час приготування. Це дозволяє значно скоротити час на пошук необхідної інформації та підвищує ефективність використання системи.

Вебзастосунок може бути використаний як у повсякденному житті для підбору страв, так і для систематизації власних кулінарних рецептів. Реалізована функціональність створення та редагування рецептів дозволяє

користувачам формувати власну базу даних, що робить систему персоналізованою та зручною для довготривалого використання.

Важливою можливістю програмного продукту є збереження обраних рецептів. Користувач може додавати рецепти до списку обраного, що забезпечує швидкий доступ до найбільш актуальних або часто використовуваних страв. Крім того, система підтримує оцінювання рецептів, що дозволяє формувати рейтинг та підвищує якість рекомендацій.

Функціональність рекомендацій відіграє важливу роль у роботі системи. На основі взаємодії користувача із вебзастосунком формується список рекомендованих рецептів, що дозволяє покращити користувацький досвід та зробити процес пошуку більш інтуїтивним.

Завдяки реалізації механізму авторизації забезпечується можливість персоналізації роботи з системою. Дані користувача, його вподобання та взаємодії зберігаються, що дозволяє формувати індивідуальний досвід використання програмного продукту.

Програмний продукт може використовуватись на різних типах пристроїв, зокрема персональних комп’ютерах, ноутбуках та мобільних пристроях. Адаптивний дизайн забезпечує коректне відображення інтерфейсу незалежно від розміру екрана, що підвищує доступність системи.

Як показано на рисунку 3.8, користувач має можливість переглядати список рецептів, використовувати фільтри та переходити до детальної інформації про обраний рецепт. Інтерфейс побудований таким чином, щоб забезпечити швидкий доступ до основних функцій системи та мінімізувати кількість дій для досягнення результату.

Крім використання у повсякденному житті, вебзастосунок може бути застосований у навчальних цілях для демонстрації принципів розробки сучасних вебсистем, а також у сфері малого бізнесу, наприклад для створення онлайн-каталогів рецептів або кулінарних блогів.

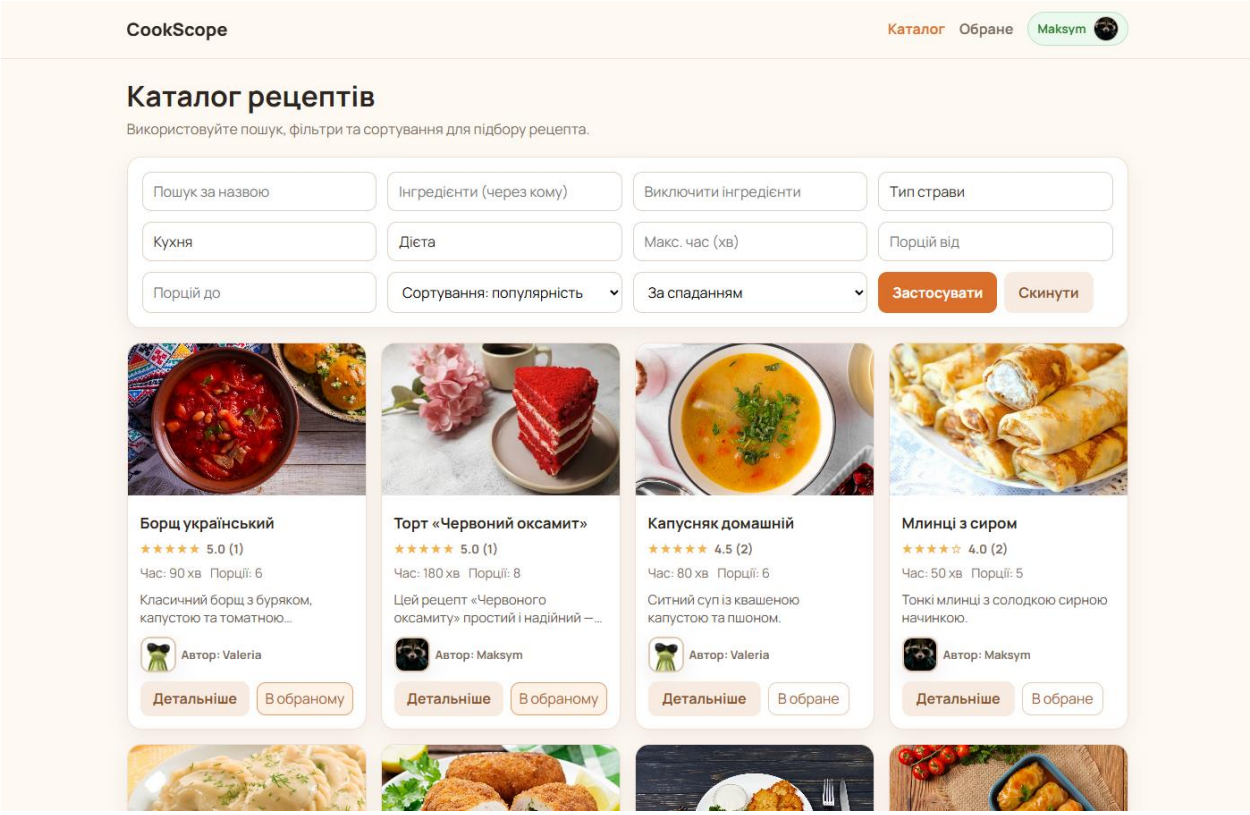

Рисунок 3.8 – Сторінка каталогу рецептів

Джерело: розроблено автором

Перспективи розвитку програмного продукту передбачають розширення функціональності, зокрема інтеграцію із зовнішніми сервісами, покращення алгоритмів рекомендацій, додавання соціальних можливостей (коментарі, підписки), а також оптимізацію продуктивності системи.

Таким чином, розроблений вебзастосунок CookScope є універсальним інструментом для роботи з кулінарними рецептами, який поєднує зручність використання, гнучкість налаштувань та можливість подальшого розвитку.

Висновки до розділу 3

У третьому розділі було розглянуто процес реалізації вебзастосунку CookScope, зокрема клієнтської та серверної частин, а також проведено тестування програмного продукту та проаналізовано можливості його практичного використання.

У ході реалізації клієнтської частини було створено інтерфейс користувача з використанням технологій HTML, CSS та JavaScript, що

забезпечує зручну взаємодію з системою, виконання пошуку, фільтрації та перегляду рецептів. Серверна частина реалізована на базі платформи Node.js із використанням фреймворку Express і забезпечує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних.

Проведене тестування підтвердило коректність роботи основних функціональних можливостей вебзастосунку, зокрема авторизації користувачів, пошуку та фільтрації рецептів, роботи з обраним і створення власних рецептів. У процесі тестування було перевірено основні сценарії взаємодії користувача із системою, що дозволило оцінити її стабільність та надійність.

Також було визначено можливості практичного використання програмного продукту, який може застосовуватись як для повсякденного підбору рецептів, так і для створення та зберігання власної кулінарної бази. Реалізований функціонал забезпечує зручність використання та можливість подальшого розвитку системи.

Отже, у результаті виконання третього розділу було реалізовано повноцінний вебзастосунок, що відповідає поставленим вимогам, демонструє стабільну роботу та є придатним до практичного використання.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено вебзастосунок CookScope, призначений для пошуку, перегляду та управління кулінарними рецептами. У першому розділі проведено аналіз предметної області та існуючих аналогів, що дозволило обґрунтувати актуальність теми, визначити недоліки наявних рішень та сформулювати основні вимоги до створюваної системи. У другому розділі виконано проєктування вебзастосунку, зокрема змодельовано поведінку користувача та системи, розроблено структуру бази даних і визначено зв’язки між основними сутностями, а також описано архітектуру програмного продукту та принципи взаємодії його компонентів. У третьому розділі здійснено реалізацію клієнтської та серверної частин із використанням сучасних вебтехнологій, проведено тестування програмного продукту та визначено можливості його практичного застосування.

Розроблений вебзастосунок забезпечує реалізацію основних функціональних можливостей, зокрема пошук і багатокритеріальну фільтрацію рецептів, перегляд детальної інформації, створення та редагування власних рецептів, роботу з обраним, оцінювання та взаємодію між користувачами. Це дозволяє ефективно використовувати систему як у повсякденній діяльності, так і як основу для подальшого розвитку подібних інформаційних рішень. Проведене тестування підтвердило коректність роботи системи, її стабільність та відповідність поставленим вимогам.

Таким чином, поставлена мета кваліфікаційної роботи досягнута, усі визначені завдання виконані у повному обсязі, а розроблений програмний продукт є функціонально завершеним, надійним у роботі та придатним до практичного використання з можливістю подальшого вдосконалення.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ

1) Cookpad. URL: https://cookpad.com/ua

2) Shuba. URL: https://shuba.life/recipes

3) Tasty. URL: https://tasty.com.ua/

4) Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – Addison-Wesley, 2004.

5) PlantUML Documentation. URL: https://plantuml.com/ru/

6) Огляд рекомендаційних систем та їх застосування в вебсервісах. URL: https://developers.google.com/machine-learning/recommendation

7) Array.prototype.filter(). URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

8) IMDb top rated movies formula. URL: https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#

9) Офіційна документація JavaScript (MDN Web Docs). URL: https://developer.mozilla.org/uk/docs/Web/JavaScript

10) ISTQB Foundation Level Syllabus. – URL: https://istqb.org/