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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка вебсайту для обміну рецептами приготування їжі

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Верцімага В.О.
(підпис) (прізвище та ініціали)

Керівник _____ Голотенко О. С
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

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

Рецензент _____ Чихіра І.В.
(підпис) (прізвище та ініціали)

Тернопіль
2025

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«27» __січня__ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

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

Студенту Верцімага Володимир Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебсайту для обміну рецептами приготування їжі

Керівник роботи Голотенко Олександр Сергійович, к.т.н., доц. кафедри КТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*7*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 20 червня 2025р.

3. Вихідні дані до роботи Інтернет джерела про технології розробки сайту Flavor Alchemy засобами HTML5, CSS3, PHP, JavaScript та MySQL.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Теоретичні засади та формування вимог до розробки вебсайту.
Розділ 2. Концептуальне та технічне проєктування вебсайту.
Розділ 3. Комплексна валідація та багаторівневе тестування якості вебсайту.
Розділ 4. Безпека життєдіяльності, основи охорони праці.
Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Титульний слайд. 2. Актуальність роботи. 3. Мета і завдання роботи. 4. Практичне значення одержаних результатів. 5. Вебсайт. 6. Порівняльний аналіз існуючих рішень. 7. Вимоги до вебсайту. 8. Архітектура вебсайту. 9. Розробка бази даних вебсайту. 10. Концептуальна модель БД вебсайту. 11. Діаграма станів публікації рецепту на вебсайті. 12. Діаграма класів вебсайту. 13. Сторінки: головна та всі рецепти. 14. Сторінки: додавання та керування рецетпами. 15. Висновки. 16. Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В. С., к.т.н., доцент		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	*Виконано*
2.	Підбір джерел про розробку сайту	08.05.2025-10.05.2025	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	11.05.2025-12.05.2025	*Виконано*
4.	Виконання дослідження щодо розробки сайту		*Виконано*
	засобами HTML5, CSS3, PHP, JavaScript та	13.05.2025-15.05.2025	
	та MySQL		
5.	Розроблення сайту	16.05.2025-22.05.2025	*Виконано*
6.	Оформлення розділу «Теоретичні засади та	23.05.2025-01.06.2025	*Виконано*
	формування вимог до розробки вебсайту»		
7.	Оформлення розділу «Концептуальне та технічне	23.05.2025-01.06.2025	*Виконано*
	проєктування вебсайту»		
8.	Оформлення розділу «Комплексна валідація та	23.05.2025-01.06.2025	*Виконано*
	багаторівневе тестування якості вебсайту»		
9.	Виконання завдання до підрозділу «Безпека життєдіяльності»	06.06.2025-07.06.2025	*Виконано*
10.	Виконання завдання до підрозділу «Основи охорони	08.06.2025-09.06.2025	*Виконано*
	праці»		
11.	Оформлення кваліфікаційної роботи	06.06.2025-10.06.2025	*Виконано*
12.	Нормоконтроль	11.06.2025-13.06.2025	*Виконано*
13.	Перевірка на плагіат	16.06.2025	*Виконано*
14.	Попередній захист кваліфікаційної роботи	18.06.2025	*Виконано*
15.	Захист кваліфікаційної роботи	23.06.2025	

Студент _____ Верцімага В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Голотенко О.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка вебсайту для обміну рецептами приготування їжі // Кваліфікаційна робота освітнього рівня “Бакалавр” // Верцімага Володимир Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних наук, група СН-41 // Тернопіль, 2025 // С. – 102, рис. – 32, табл. – 8, слайди – 16, додат. – 8, бібліогр. – 53.

Ключові слова: веб-портал, обмін рецептами, кулінарія, React, Node.js, PostgreSQL, кросплатформність, адаптивний дизайн, CI/CD-деплоймент, тестування

Кваліфікаційна робота присвячена повному циклу створення та впровадження тематичного веб-порталу для обміну кулінарними рецептами. Ідея полягає в тому, щоб запропонувати користувачам дружню до мобільних пристроїв платформу, де можна не лише ділитися власними стравами, а й коментувати, оцінювати й зберігати вподобані рецепти до персональної колекції. Платформа покликана об’єднати різнорідну аудиторію – від початківців, що роблять перші кроки на кухні, до досвідчених шефів, які шукають натхнення для нових гастрономічних експериментів.

Перший розділ присвячено ґрунтовному аналізу предметної області: розглянуто сучасні тренди фуд-блогінгу, класифіковано основні категорії рецептів і проаналізовано конкурентні сервіси. На основі цього сформульовано детальний перелік функціональних і нефункціональних вимог до майбутнього ресурсу, визначено типові сценарії взаємодії та детально описано ключових учасників системи – авторів рецептів, читачів, модераторів і гостей. Також наведено аргументацію вибору стеку технологій (React, Node.js, PostgreSQL) і пояснено, чому саме ці інструменти найкраще задовольняють потреби проєкту з огляду на масштабованість, безпеку й швидкість розробки.

Другий розділ зосереджується на етапі проєктування. У ньому докладно описано багато-рівневу архітектуру (клієнт–сервер), наведено логічну та фізичну схему бази даних, спроєктовано ER-діаграми таблиць «Користувач», «Рецепт», «Інгредієнт», «Коментар» і «Оцінка». Відображено діаграми класів, послідовностей і розміщення, що демонструють зв’язки між компонентами фронтенду та бекенду. Окремо описано структуру каталогів репозиторію, принципи REST-маршрутизації та механізми керування станом на клієнті за допомогою Redux Toolkit.

У третьому розділі наведено комплексну перевірку працездатності веб-порталу. Проведено автоматизовану валідацію HTML і CSS-коду, юніт-тестування API-ендпоінтів і модулів React, а також ручне тестування на різних браузерах і екранах, щоб гарантувати кросплатформність і адаптивність. Після порівняльного аналізу обрано оптимальний хостинг-провайдер і описано процедуру деплойменту через CI/CD-конвеєр GitHub Actions. Додатково подано детальні інструкції щодо експлуатації ключових сторінок – від реєстрації користувача до створення та публікації нового рецепта.

Четвертий розділ присвячено питанням охорони праці та безпеки життєдіяльності. Описано алгоритм надання першої допомоги у разі ураження електричним струмом, наведено рекомендації щодо ергономічного розташування робочого місця за ПК, раціонального режиму праці та відпочинку, а також основні правила пожежної безпеки й захисту від шкідливого випромінювання монітора. Матеріал доповнено нормативними посиланнями на чинні ДСТУ та державні санітарні правила, що регламентують роботу з комп’ютерним обладнанням.

ANNOTATION

Development of a Recipe-Sharing Website // Qualification work of the educational, level “Bachelor” // Vertsimaha Volodymyr Oleksandrovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2025 // P. – 102, fig. – 32, tabl. – 8, slides – 16, annexes. – 8, references – 53.

Keywords: web portal, recipe sharing, cooking, React, Node.js, PostgreSQL, cross-platform, adaptive design, CI/CD deployment, testing

The qualification work is dedicated to the full cycle of creating and implementing a thematic web portal for sharing culinary recipes. The idea is to offer users a mobile-friendly platform where they can not only share their own dishes, but also comment, rate and save their favorite recipes to a personal collection. The platform is designed to unite a diverse audience - from beginners taking their first steps in the kitchen to experienced chefs looking for inspiration for new gastronomic experiments.

The first section is devoted to a thorough analysis of the subject area: modern trends in food blogging are considered, the main categories of recipes are classified and competitive services are analyzed. Based on this, a detailed list of functional and non-functional requirements for the future resource is formulated, typical interaction scenarios are identified, and key participants in the system are described in detail - recipe authors, readers, moderators, and guests. The argumentation for choosing a technology stack (React, Node.js, PostgreSQL) is also provided and it is explained why these tools best meet the needs of the project in terms of scalability, security, and development speed. The second section focuses on the design stage. It describes in detail the multi-tier architecture (client-server), provides a logical and physical database schema, and designs ER diagrams for the User, Recipe, Ingredient, Comment, and Rating tables. Class, sequence, and placement diagrams are displayed, demonstrating the relationships between frontend and backend components. The

structure of the repository directories, REST routing principles, and client state management mechanisms using the Redux Toolkit are separately described.

The third section provides a comprehensive check of the web portal's performance. Automated validation of HTML and CSS code, unit testing of API endpoints and React modules, as well as manual testing on different browsers and screens were carried out to ensure cross-platform and adaptability. After comparative analysis, the optimal hosting provider was selected and the deployment procedure via the GitHub Actions CI/CD pipeline was described. Additionally, detailed instructions for operating key pages are provided - from user registration to creating and publishing a new recipe.

The fourth section is devoted to issues of occupational health and safety. The algorithm for providing first aid in case of electric shock is described, recommendations are given for the ergonomic arrangement of the workplace at the PC, a rational work and rest regime, as well as the basic rules of fire safety and protection from harmful monitor radiation. The material is supplemented with regulatory references to the current DSTU and state sanitary rules regulating work with computer equipment.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

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

HTML (англ. HyperText Markup Language) – мова розмітки гіпертекстових документів.

ID (англ. Identity Document) – унікальний номер, який використовується для ідентифікації.

JS (англ. JavaScript) – мова програмування для створення інтерактивності та веб-сайтах.

MySQL (англ. My Structured Query Language) – вільна система керування реляційними базами даних.

PHP (англ. Hypertext Preprocessor, Personal Hone Page) – гіпертекстовий препроцесор, скриптова мова програмування для генерування веб-сторінок на стороні веб-сервера.

W3C (англ. World Wide Web Consortium) – організація, яка створила стандарти веб-розробки й надає онлайн валідатор для перевірки правильності коду.

БД – база даних.

ПК – персональний комп’ютер.

ЗМІСТ

ВСТУП

Актуальність теми. У контексті стрімкої цифровізації щоденних побутових практик кулінарія перетворилася з приватної сфери діяльності на глобальний комунікаційний простір. Онлайн-платформи, що забезпечують обмін гастрономічним досвідом, відіграють дедалі помітнішу роль у формуванні харчових звичок, популяризації здорового способу життя та підтримці локальних кулінарних традицій. Водночас більшість наявних сервісів страждає від обмеженої персоналізації, слабкої соціальної взаємодії або застарілих технічних рішень, що ускладнює масштабування й адаптацію під мобільні пристрої. Тому створення сучасного вебсайту, здатного об’єднати активну кулінарну спільноту на принципах взаємного обміну знаннями, кросплатформності та високої продуктивності, є актуальним завданням як з практичної, так і з науково-технічної точки зору.

Мета і задачі дослідження. Метою кваліфікаційної роботи є розроблення, впровадження та експериментальна перевірка вебсайту, що забезпечує зручний обмін рецептами, підтримує соціальні функції (коментування, оцінювання, збереження у вибране) і відповідає сучасним вимогам щодо безпеки, масштабованості та ергономіки користувацького інтерфейсу.

Для досягнення поставленої мети було сформовано такі завдання:

- Провести аналіз предметної області: дослідити тенденції фуд-блогінгу, оцінити функціонал конкурентних сервісів і сформулювати перелік цільових аудиторій з їхніми інформаційними потребами.
- Сформувати вимоги до системи: визначити функціональні й нефункціональні характеристики, що забезпечать високу продуктивність, безпечне зберігання даних і дружній до мобільних пристроїв дизайн.
- Вибрати й обґрунтувати технологічний стек (React, Node.js, PostgreSQL) та середовище розроблення, виходячи з критеріїв масштабованості, доступності бібліотек і швидкості розгортання.

- Спроєктувати архітектуру вебсайту: розробити багаторівневу клієнт–серверну модель, побудувати логічну та фізичну схеми бази даних, підготувати діаграми класів, послідовностей і розміщення.
- Реалізувати програмний прототип веб-платформи з підтримкою реєстрації користувачів, публікації рецептів, пошуку, фільтрації та соціальної взаємодії.
- Провести валідацію та багаторівневе тестування: автоматизовану перевірку коду, юніт- і інтеграційні тести, а також ручне тестування кросбраузерності та адаптивності інтерфейсу.

Виконання цих завдань дозволить комплексно вирішити проблему створення та впровадження сучасного вебсайту для обміну кулінарними рецептами й підтвердити ефективність обраних технологічних і організаційних рішень.

Практичне значення одержаних результатів. Практична цінність результатів кваліфікаційної роботи полягає у створенні повністю функціонального вебсайту, який може бути негайно розгорнутий у виробничому середовищі й одразу приносити користь різним категоріям користувачів. По-перше, платформа надає домашнім кулінарам і професійним кухарям інтуїтивний інструмент для публікації, каталогізації та поширення авторських рецептів, що сприяє популяризації здорового харчування та збереженню локальних гастрономічних традицій. По-друге, завдяки вбудованим механізмам оцінювання, коментування та персональних добірок система стимулює активну взаємодію всередині спільноти, полегшує пошук релевантного контенту й скорочує час на планування щоденного меню.

З технічного погляду розроблений продукт демонструє репрезентативний зразок застосування сучасного JavaScript-стеку (React + Node.js + PostgreSQL) у реальному проєкті. Оптимізована архітектура клієнт–сервер, реалізація CI/CD-конвеєра та модульний підхід до тестування роблять платформу придатною як навчальний кейс для курсів із веб-розробки, DevOps та баз даних. Крім того, відкритий і добре задокументований вихідний код дозволяє швидко адаптувати рішення під різні бізнес-моделі: від нішевих кулінарних блогів і корпоративних

порталів харчових брендів до маркетплейсів локальних фуд-стартапів. Таким чином, отримані результати мають потенціал комерційної реалізації, можуть бути інтегровані у системи планування раціону або сервісні мобільні додатки й слугувати базою для подальших досліджень у галузі персоналізованих рекомендацій та соціальних мереж харчових уподобань.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ТА ФОРМУВАННЯ ВИМОГ ДО РОЗРОБКИ ВЕБСАЙТУ

1.1 Аналіз предметної області

Кулінарні спільноти в інтернеті вже кілька десятиліть залишаються одним із найдинамічніших сегментів мережі. Попит на якісний контент про приготування їжі стрімко зростає паралельно з поширенням здорового способу життя, популярністю локальних продуктів і прагненням користувачів урізноманітнити свій раціон. Численні платформи, що спеціалізуються на рецептах, набули рис соціальних мереж: вони дають змогу ставити оцінки, коментувати, зберігати вподобані страви та обмінюватися власними кулінарними знахідками. Утім, більшість існуючих сервісів орієнтовані передусім на англомовну аудиторію або на певну кухню, тоді як універсальних українськомовних рішень, де контент легко структурований і швидко адаптується під мобільні пристрої, доволі мало.

Типовими учасниками системи є автори рецептів, читачі-початківці, досвідчені гастрономи й модератори. Автори зацікавлені у простому інтерфейсі додавання контенту, можливості завантажити фото чи відео й автоматично формувати друковану версію рецепта. Читачі насамперед шукають швидку навігацію, фільтри за інгредієнтами, часом приготування та калорійністю, а також достовірні відгуки інших користувачів. Просунуті кулінари звертають увагу на систему тегів для складніших технік, наприклад су-від чи ферментацію, і на можливість будувати власні збірки меню. Нарешті, модераторам потрібні інструменти перевірки правопису, виявлення дубльованих матеріалів і контролю за дотриманням авторських прав.

Технологічні вимоги до сучасного вебсайту для обміну рецептами приготування їжі формуються під впливом кількох трендів. По-перше, мобільний трафік перевищує десктопний, тож адаптивний дизайн стає обов’язковою передумовою успіху. По-друге, популярність коротких відео підштовхує до підтримки медіафайлів великого обсягу та інтеграції з хмарними

CDN-сховищами. По-третє, персоналізовані рекомендації поступово витісняють класичне сортування за датою: алгоритми машинного навчання аналізують історію переглядів, щоб пропонувати релевантні страви. Однак подібні функції вимагають ретельного налаштування безпеки доступу до персональних даних, особливо у світлі вимог GDPR та українського законодавства про захист інформації.

Під час аналізу конкурентних продуктів виявлено дві ключові проблеми. Перша – недосконала локалізація: українська версія часто обмежується перекладом інтерфейсу без адаптації кулінарних мірок і назв інгредієнтів. Друга – відсутність глибокої соціальної взаємодії: коментарі або вбудовані чати переважно реалізовані поверхнево й не стимулюють накопичення колективного досвіду. Отже, новий вебсайт має запропонувати гнучку модель контент-менеджменту, де рецепти підтримують кілька мов вихідних даних, а комунікаційні засоби заохочують користувачів ділитися нюансами технологічних процесів, оцінювати складність та уточнювати помилки автора.

Підсумовуючи, ринок вебсервісів для поширення рецептів залишається відкритим для рішень, що поєднують адаптивний дизайн, мультимедійну подачу, розширені соціальні механізми та якісну українську локалізацію. Впровадження такого вебсайту здатне задовольнити потреби широкої аудиторії – від студентів і молодих сімей до професійних кухарів – і водночас створити базу для подальших досліджень у галузі персоналізованих гастрономічних рекомендацій..

1.2 Аналіз наявних рішень

Світові кулінарні спільноти давно перетворилися на повноцінні соціальні медіа-майданчики. Найстаршим прикладом є Allrecipes, де понад 7 мільйонів оцінок і відгуків формують «колективний смак», що впливає на ранжування страв і підказує авторам, які рецепти доробити чи переосмислити. Сервіс заохочує користувачів ділитися власними фото готових страв і пропонує добірки

«топ-продуктів року», орієнтуючись на сезонні тренди харчування, зокрема healthy-food і plant-based меню.

Універсальний підхід демонструє також Food52, що поєднує рецептурний контент із власною крамницею посуду та домашнього декору: платформа має щомісячні конкурси, де переможець отримує промоцію рецепта й знижки у фірмовому магазині, чим зміцнює лояльність спільноти та допомагає монетизувати трафік.

Мобільно-орієнтовану нішу очолює Tasty від BuzzFeed: застосунок пропонує понад 10 тисяч відеорецептів, режим покрокового приготування з «несплячим» екраном та функцію Recipe Remixes, що дозволяє змінювати калорійність чи інгредієнти «на льоту». Останні оновлення додали AI-асистента Botatouille, який радить страви за часом доби і сезонністю, а також інтеграцію з он-лайн кошиком Walmart для миттєвої купівлі продуктів.

На персоналізацію робить ставку і Yummly: алгоритми машинного навчання аналізують уподобання, фільтрують алергени та складають список покупок за обраними стравами, що підвищує утримання аудиторії та сприяє партнерству з брендами продуктів.

Український сегмент представлений переважно медійними проектами. Сайт Smachno.ua пропонує тематичні добірки до свят і щотижневі «меню вихідного дня», проте навігація побудована радше за редакційною логікою, ніж за запитами користувача.

Платформа Klopotenko.com робить ставку на особистий бренд шеф-кухаря й додає нестандартні рубрики (кулінарні гороскопи, поєднання страв і музики), але поки що не має розвиненої системи взаємних рецензій та оцифрованих кулінарних мір.

Cookpad, хоч і глобальна, підтримує українську локалізацію, дозволяє користувачеві публікувати власні рецепти з фото та вести кулінарний блог, однак пошук за складниками працює не завжди коректно через неоднозначність назв інгредієнтів та відсутність єдиного словника міри.

Порівняльний огляд показує спільні недоліки чинних рішень: обмежена багатомовність, недостатня підтримка інтерактивних форматів (покрокове відео

українською, AR-підказки нарізки продуктів), а також слабка інтеграція соціальних механік гейміфікації, що могла б стимулювати «рецептурні батли» чи обмін лайф-хаками в реальному часі. Крім того, лише окремі гравці впровадили прозору систему авторських прав на зображення й відео. Таким чином, ринок залишається відкритим для вебсайту обміну рецептами приготування їжі, який поєднає гнучку українську локалізацію, алгоритмічні рекомендації, глибоку соціальну взаємодію та безпечне зберігання персональних даних, надаючи користувачам водночас інструменти навчання й самопрезентації.

1.2.1 Онлайн-платформа «Пательня» для розміщення кулінарних рецептів

«Пательня» вирізняється привабливою палітрою кольорів і добре продуманою, на перший погляд, навігаційною сіткою. Домашня сторінка відкривається добіркою популярних страв, а під кожною карткою містяться лічильники вподобайок і переглядів, що мимоволі заохочує користувача дослідити контент глибше. Розробники подбали про пошук за ключовими словами й базові фільтри – за типом кухні, складністю та тривалістю приготування. Крім того, авторизовані відвідувачі можуть додавати власні рецепти: форму заповнити просто, є поля для детального опису інгредієнтів, кроків і порад.

Проте враження від зручності зменшується, коли увага падає на дрібні, але відчутні огріхи. На сторінках деяких страв немає світлин (див. рис. 1.1) – і це суттєвий мінус, адже більшість користувачів обирає страву насамперед очима [1]. Ще одне слабке місце – подвійне головне меню. Його частину розміщено у шапці, а частину – у футері, тому власникам смартфонів доводиться прокручувати сторінку, щоб дістатися потрібного розділу. Таке рішення уповільнює навігацію та збиває з пантелику при першому знайомстві з ресурсом, особливо в мобільній версії [1].

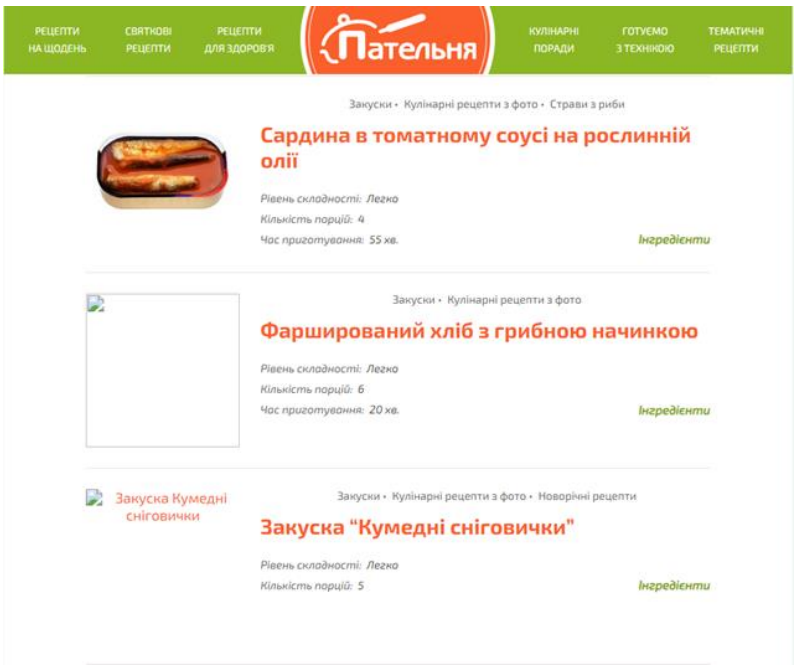

Рисунок 1.1 – Приклад рецепту, наведений без супровідної світлини

До недоліків можна також віднести надто дрібний шрифт у списках інгредієнтів, що погіршує читаність на екранах із високою щільністю пікселів. Крім того, не всі медіафайли мають альтернативний текст, отже сайт втрачає бали доступності. З технічного боку помітно, що сторінки без зображень завантажуються значно швидше, тому оптимізована система обов'язкових ілюстрацій могла б водночас покращити UX і зберегти швидкість. У підсумку «Пательня» справляє позитивне перше враження завдяки естетиці та широкому каталогу рецептів, однак потребує доопрацювань – передусім у презентації контенту та мобільній навігації.

1.2.2 Онлайн-платформа «Кукорама» для розміщення кулінарних рецептів

«Кукорама» справляє приємне перше враження завдяки легкій кольоровій гамі, логічно розміщеним елементам навігації й можливості швидко переключатися між основними розділами – «Рецепти», «Блоги», «Обговорення». Після авторизації користувач отримує повний набір інструментів для створення власних записів: форму з полями для інгредієнтів, кроків приготування і фотографій, а також опцію додати страву до персональної добірки, щоби не

втратити її серед сотень інших. Сам факт наявності «улюблених» – вагомий плюс, адже відвідувачеві не потрібно щоразу повторно шукати вподобані рецепти.

Водночас широкі функціональні можливості мають зворотний бік. Кожна сторінка перенасичена додатковими блоками – рекламою, списками популярних дописів, зовнішніми віджетами, що не стосуються безпосередньо конкретної страви. Через це вміст довго завантажується, особливо на мобільних мережах; читач може втратити терпіння ще до того, як дочекається повного рендеру картинок і скриптів. На рисунку 1.2 видно, що поряд із рецептурною частиною розміщено великий фрагмент вторинної інформації, яка відволікає від основного контенту [2].

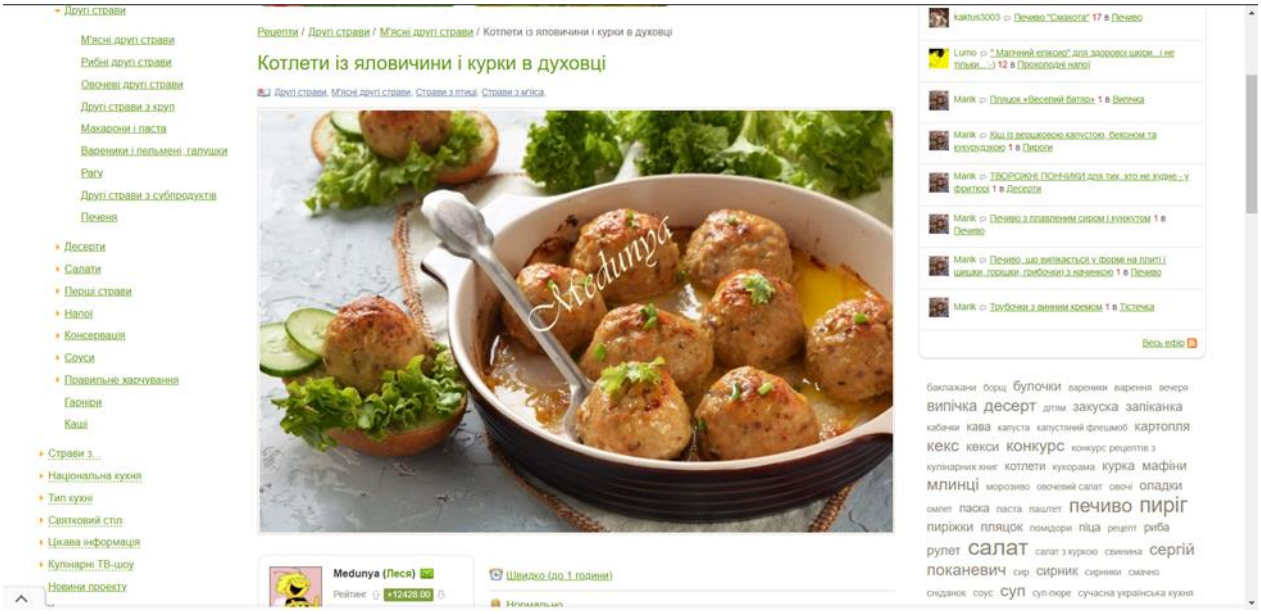

Рисунок 1.2 – Перегляд рецепта з надмірними інформаційними блоками

Помітна й низка дрібніших недоліків, що накопичуються у відчутну проблему з юзабіліті. Шрифти різного розміру і блідо-сірих відтінків зустрічаються на одному екрані, тож людям із порушеннями зору доводиться збільшувати масштаб або напружувати очі, щоб розібрати текст [2]. У комірках таблиць інгредієнтів іноді використано курсив маленького кегля, що додатково погіршує читабельність. Поліпшити ситуацію допомогла б єдина типографічна

сітка та контрастніші кольори, але наразі ці деталі знижують комфорт тривалого перегляду матеріалів ресурсу.

1.2.3 Порівняння наявних рішень

Компаративний огляд двох популярних українських кулінарних майданчиків дає змогу чіткіше окреслити ніші, які залишаються незаповненими, та з’ясувати, якими саме властивостями повинен володіти сучасний вебсайт для обміну рецептами приготування їжі. «Пательня» і «Кукорама» обрали різні підходи до побудови взаємодії з аудиторією, однак обидві платформи зіштовхнулися з подібними викликами, пов’язаними передусім з балансом між візуальною привабливістю, швидкодією й доступністю контенту.

«Пательня» робить ставку на мінімалістичний інтерфейс і лаконічну палітру, що одразу створює відчуття впорядкованості. Користувач бачить компактні картки страв без надлишку тексту, тому головний акцент спрямовано на зображення. Коли ж у картки немає фотографії, – а така ситуація трапляється досить часто, – логіка порталу ламалася: сторінка видається недоповненою, а сам рецепт позбавляється важливого емоційного тригера. На мобільних пристроях цей недолік підсилюється подвійним меню, розкиданим між верхньою панеллю й футером; людині доводиться прокручувати сторінку навіть для простих переходів, що збільшує час до першої інтеракції й негативно впливає на показник утримання.

«Кукорама» – повна стилістична протилежність. Автори сервісу намагаються дати читачеві якомога більше інформації на одній сторінці: поряд із рецептом з’являються смуги популярних публікацій, рекламні банери, блоки коментарів і тематичні оголошення. Таке насичення контентом збагачує сайт з погляду функціональності, проте істотно навантажує канал зв’язку та браузер. На смартфонах зі слабким процесором або обмеженим трафіком повне завантаження рецептурної сторінки іноді триває довше, ніж приготування наведеного десерту. У поєднанні з нечіткою типографікою – тонким сіруватим

шрифтом, який губиться на світлому тлі – це формує помітний бар’єр для людей з ослабленим зором і знижує загальну зручність читання.

Якщо зіставити сильні сторони платформ, «Пательня» перемагає завдяки швидкому пошуку та чистому, не перевантаженому інтерфейсу, що спонукає новачка одразу зосередитися на кулінарному процесі. «Кукорама» ж приваблює широтою соціального функціоналу: можливість вести кулінарний блог, обговорювати рецепти у коментарях і збирати улюблені страви у власні добірки мотивує повертатися й будувати довкола сервісу активну спільноту. Водночас слабкі місця кожного ресурсу віддзеркалюють переваги конкурента: де «Пательня» страждає від браку медіавмісту й розірваної навігації, «Кукорама» грішить перенасиченістю сторінок і поганою читабельністю.

Отже, узагальнений портрет ідеального рішення містить легку, адаптивну верстку без дублювання навігаційних елементів, обов’язкову наявність якісних фото- й відеоілюстрацій для кожного рецепта, оптимізоване кешування, а також зрозумілу типографічну систему, що відповідає рекомендаціям Web Content Accessibility Guidelines. Додатково слід передбачити гнучкі механізми соціальної взаємодії, але виводити їх дозовано, аби не перевантажити головний контент. Поєднання найкращих практик обох сервісів дало б змогу створити вебсайт, який не лише забезпечує швидкий доступ до рецептів, а й формує комфортний простір для постійної кулінарної взаємодії.

1.3 Визначення функціональних і нефункціональних вимог до вебсайту

Формування вимог до вебсайту, призначеного для обміну рецептами приготування їжі, спирається на попередній аналіз наявних рішень і очікування цільової аудиторії. Насамперед окреслюються функціональні потреби, що безпосередньо забезпечують основний сценарій: автор має швидко публікувати рецепт із фотографіями, покроковим описом і списком інгредієнтів, а читач – так само легко знаходити, фільтрувати й зберігати цікаві страви. Звідси випливає обов’язкова підтримка пошуку за ключовими словами, тегами, кухнями світу,

нутрітивною цінністю та тривалістю приготування. Додатково потрібні механізми оцінювання, коментування та формування особистих колекцій, щоб стимулювати повторні відвідування та зростання спільноти.

Нефункціональний пласт вимог визначає, яким саме має бути користувацький досвід. Інтерфейс повинен коректно відображатися на екранах від 320 px до 4K, забезпечуючи повністю адаптивний дизайн і час першого візуального відгуку не довше трьох секунд при швидкості з'єднання 4 G. Типографіка мусить відповідати принаймні рівню AA рекомендацій WCAG 2.1; усі медіафайли мають супроводжуватися альтернативним текстом, а контраст кольорів – залишатися в межах, зручних для людей з ослабленим зором. З огляду на швидке зростання мобільного трафіку, пріоритетом є оптимізація зображень через WebP та lazy load-завантаження, а відео – через HLS-потоки з автоматичним вибором якості.

Архітектурні вимоги охоплюють розділення клієнтської та серверної частини, використання REST або GraphQL-API для взаємодії, а також горизонтальне масштабування бекенду. Сховище даних варто реалізувати на реляційній СКБД (наприклад, PostgreSQL), доповненій об'єктним CDN для великих медіафайлів. Умовою подальшого розвитку стає модульна структура репозиторію, що спрощує впровадження CI/CD-конвеєра і автоматичне розгортання оновлень без простою сервісу.

Безпека й конфіденційність посідають окреме місце серед вимог. Аутентифікація користувачів повинна підтримувати як традиційні методи (e-mail + пароль), так і соціальний логін. Усі запити мають передаватися через HTTPS; критичні дані – шифруватися у сховищі. На рівні бізнес-логіки знадобляться засоби попередження XSS, CSRF і SQL-ін'єкцій, а також система резервного копіювання з відновленням не пізніше шести годин з моменту збою. Окремо слід забезпечити можливість видалення особистих даних за запитом, аби відповідати вимогам GDPR та українського законодавства щодо захисту персональної інформації.

Ще одна група вимог стосується локалізації та контент-менеджменту. Вебсайт має підтримувати багатомовний інтерфейс із гнучким перемиканням

мови, переклад усіх кулінарних мір і можливість дублювати рецепти різними мовами. Для модераторів необхідні інструменти автоматичної перевірки правопису, виявлення дублікатів, опрацювання скарг та напівавтоматична система підтвердження фотографій. Щоб заохочувати авторів, доречно реалізувати систему бейджів і рейтингу, а також можливість експортувати рецепти у PDF чи друковану версію стандартного формату.

Нарешті, важливим залишається показник надійності та доступності. Планова готовність сервісу має складати не менше 99,5 % на рік, а середній час виправлення критичних помилок – не перевищувати 24 годин. Моніторинг продуктивності, журналювання подій та аналітика поведінки користувачів інтегруються з перших етапів розроблення, що дозволяє оперативно реагувати на пікові навантаження й вдосконалювати персоналізовані рекомендації, спираючись на реальні дані експлуатації. Сукупність перелічених вимог формує комплексне технічне завдання, покликане забезпечити не лише стабільну роботу вебсайту, а й створення комфортного, безпечного та інклюзивного простору для кулінарної творчості.

1.4 Визначення ключових ролей користувачів і сценаріїв їхньої взаємодії в системі обміну кулінарними рецептами

Щоби коректно сформувати вимоги до майбутньої системи, насамперед потрібно зрозуміти, які саме учасники взаємодіятимуть із вебсайтом, – іншими словами, виконати пошук актантів. Така процедура дає змогу не лише окреслити спектр функцій кожного користувача, а й заздалегідь передбачити конфлікти доступу, типові сценарії переходів між ролями та потенційні точки зростання функціоналу. У контексті ресурсу, що спеціалізується на публікації та обміні рецептами, виділено чотири базові групи акторів, кожна з яких виконує власний набір дій і впливає на поведінку інших.

Незареєстрований відвідувач. Це людина, яка вперше переходить на сторінку й поки що не має облікового запису. Її інтерес зосереджений на огляді популярних страв, тестовому пошуку та швидкому знайомстві з дизайном.

Незареєстрований гість читає рецепти в режимі «тільки перегляд», але не може залишати коментарі, оцінки чи зберігати вподобані матеріали у добірки.

Зареєстрований користувач. Щойно гість проходить процедуру створення акаунта, він отримує розширений набір можливостей: персональний профіль, налаштування алергенних фільтрів, можливість керувати «улюбленими» рецептами та залишати відгуки. У разі бажання та відповідності базовим правилам спільноти такий користувач може перейти до наступного рівня.

Автор контенту. Це підмножина зареєстрованих користувачів, які опублікували хоча б один власний рецепт. Для них особливо важливі інструменти швидкого завантаження фото, автозаповнення списку інгредієнтів, перевірка унікальності тексту й статистика переглядів. Автори взаємодіють між собою через коментарі, відповіді та тематичні флешмоби, що стимулює постійне зростання бази рецептів.

Адміністратор. Найвища роль у системі. Адміністратор керує структурою категорій, розподіляє модераторські права, слідкує за дотриманням авторських прав і запускає рекламні кампанії. Він також відповідає за резервне копіювання даних і відновлення працездатності у разі масштабних збоїв.

На діаграмі ролей (див. рис. 1.3) показано, як відбувається еволюція прав користувача: незареєстрований відвідувач після успішної реєстрації стає зареєстрованим, із часом – автором, а в особливих випадках – модератором чи адміністратором. Наведена схема демонструє прямі залежності між рівнем довіри спільноти та обсягом доступних можливостей.

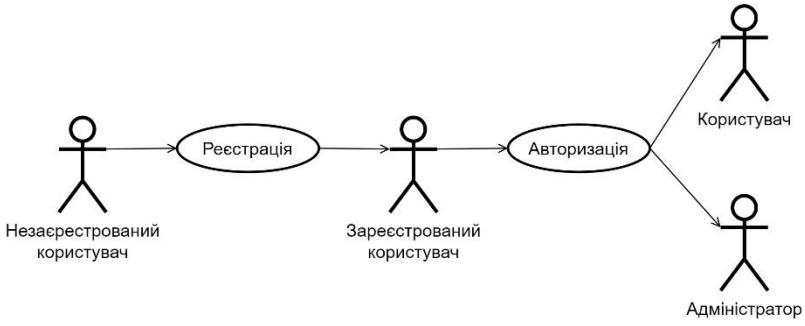

Рисунок 1.1 – Діаграма ролей взаємодії акторів вебсайту

Другий графічний матеріал (див. рис. 1.4) деталізує, які саме сторінки відкриті кожній групі. Відвідувач без акаунта бачить лише головну, каталог рецептів, інформаційні рубрики й форми входу або реєстрації. Зареєстрованому користувачеві додаються особистий кабінет і функції коментування, тоді як автор отримує ще й конструктор публікацій та статистику власних матеріалів. Адміністратор, у свою чергу, має універсальний доступ до всіх модулів: від керування користувачами до редагування глобальних налаштувань платформи.

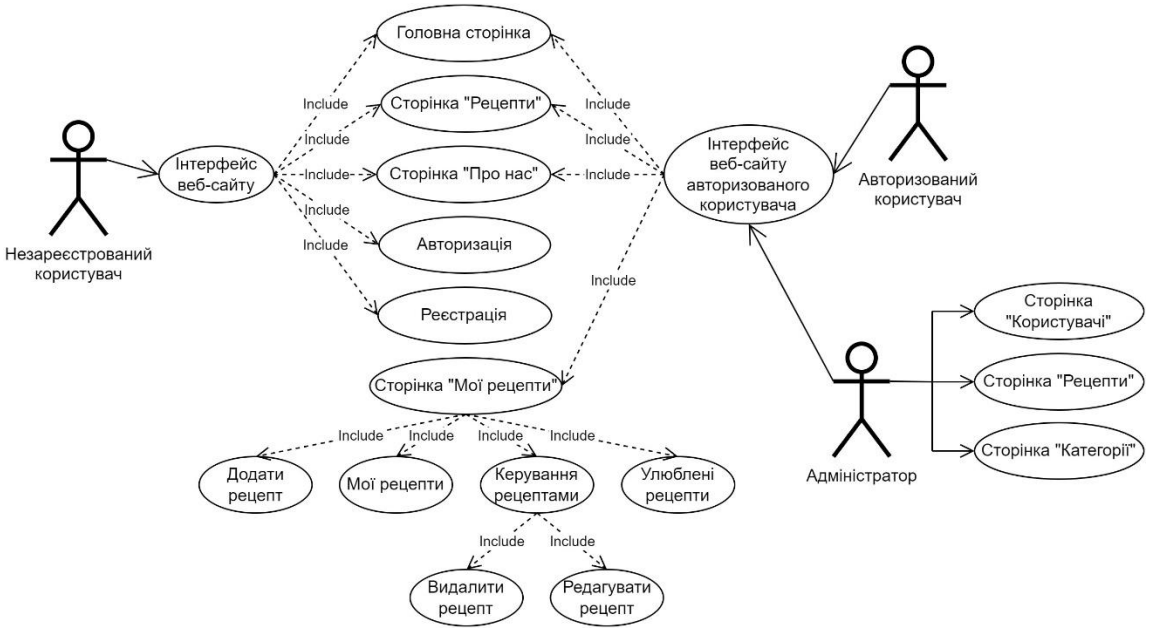

Рисунок 1.4 – Діаграма варіантів використання вебсайту

Таким чином, чотири описані ролі утворюють мінімально достатній каркас взаємодії для кулінарного вебсайту. Чітке розмежування повноважень спрощує подальше проєктування системи прав доступу, дозволяє забезпечити безпеку даних і водночас гарантує, що користувач поступово отримуватиме нові можливості в міру зростання своєї активності та вкладеного внеску у розвиток спільноти.

1.5 Сценарії використання платформи обміну кулінарними рецептами

Щоб зрозуміти, як саме відвідувачі – і новачки, і досвідчені кулінари – взаємодіятимуть із платформою, доцільно розглянути найбільш показові сценарії роботи вебсайту. Кожен сценарій описує окрему життєву історію: що спонукає людину відкрити ресурс, які дії вона виконує крок за кроком, як сайт відповідає на запити й яких результатів зрештою досягає користувач.

Уявімо гостя, який випадково натрапив на головну сторінку, шукаючи «пасту з креветками без глютену». Він не має облікового запису, тому система показує спрощений інтерфейс із великим рядком пошуку та плиткою популярних категорій. Вводячи ключові слова, відвідувач одразу бачить підказки з варіантами рецептів, а після натискання кнопки «Знайти» дістає перелік страв, відфільтрованих за часом приготування та дієтичними позначками. Пройшовши ланцюжком «Головна → Пошук → Перегляд рецепта», користувач або закриє вкладку, або вирішить зареєструватися, щоб зберегти знайдений рецепт на потім.

Наступна ситуація розгортається вже в акаунті автора. Власник профілю натискає «Додати рецепт», заповнює назву, короткий опис і список інгредієнтів, користуючись авто-доповненням, яке підтягує усталені міри та одиниці. Коли всі кроки приготування описано, автор завантажує світлини та активує попередній перегляд. Система автоматично генерує калорійність, пропонує хештеги та перевіряє, чи не дублює текст існуючих матеріалів. Після натискання «Опублікувати» рецепт потрапляє у чергу модератора, а у профілі автора з’являється статус «очікує на перевірку».

Вже готовий рецепт бачить зареєстрований користувач, який переглядає стрічку рекомендацій. Він ставить оцінку, пише коментар із порадою щодо альтернативного інгредієнта й додає страву у власну колекцію «Суботні вечері». Через тиждень, повернувшись до добірки, користувач переглядає список куплених продуктів і бере смартфон на кухню: сайт переходить у режим покрокового приготування з великими кнопками «далі» й «назад» для зручності на сенсорному екрані. Після успішного тесту він завантажує фото власного

результату, що автоматично доповнює галерею рецепта та підвищує його популярність у видачі.

У бекграунді працює адміністратор платформи. Отримавши сповіщення про нову публікацію, він переглядає текст на відповідність правилам, підтверджує оригінальність фото та додає рецепт до тематичної колекції «Осінні морепродукти». Так само адміністратор втручається, якщо система машинного виявлення спаму фіксує підозріло часті посилання в коментарях: він може приховати дописи, заблокувати акаунт або призначити перевірку правомірності дій.

На наступному етапі сайт пропонує користувачеві розширений функціонал: створення тижневого меню. Коли людина додає кілька улюблених страв до плану, портальна логіка генерує список покупок, групуючи однакові інгредієнти й підраховуючи сумарну вагу. Якщо профіль містить інформацію про алергії, система автоматично виключає небажані продукти або радить безпечні замінники. Результат можна експортувати у PDF чи надіслати на електронну пошту.

Розгорнутий перелік сценаріїв взаємодії із системою подано у таблиці-реєстрі варіантів використання (див. додаток А). У цьому підсумковому документі зведено в єдину структуру всіх ключових акторів платформи, їхні повноваження, можливі дії та очікувані результати, що дозволяє швидко оцінити повноту функціоналу й узгодженість прав доступу. Кожен запис містить назву сценарію, короткий опис кроків, перелік задіяних ролей, а також позначку про бізнес-призначення – від базового перегляду рецептів до адміністративного керування контентом. Такий формат сприяє прозорому управлінню вимогами, спрощує пріоритезацію задач у беклозі й дає можливість оперативно вносити зміни, коли з’являються нові функції чи змінюються потреби користувачів [5].

1.6 Комплексна оцінка архітектурних і технологічних підходів до розроблення вебсайтів

Розглядаючи способи створення вебсайту для обміну кулінарними рецептами, насамперед варто оцінити, наскільки кожен із популярних підходів відповідає вимогам швидкого запуску, масштабованості й безпеки даних. Готові CMS на зразок WordPress із плагінами для «recipe-card» дають змогу підняти мінімально життєздатний продукт буквально за кілька днів і користуватися величезним вибором тем та розширень. Разом із тим ядро подібної системи містить багато надлишкового коду, який складно оптимізувати під високу швидкодію та нестандартні поля, скажімо, для калорійності чи алергенів, а без ретельного захисту CMS лишається вразливою до XSS- і SQL-ін’єкцій.

Low-code і no-code платформи на кшталт Bubble, Webflow або Wix іще більше знижують бар’єр входу: засновники без технічної підготовки можуть «перетягнути» компоненти мишею, отримати хмарний хостинг у комплекті й одразу протестувати ринковий попит. Недоліком стає повна прив’язаність до провайдера: перенести бекенд без переписування практично неможливо, а при спробі додати реактивну стрічку чи алгоритмічні рекомендації виникають апаратні й тарифні обмеження.

Третю стратегію пропонують класичні MVC-фреймворки на кшталт Django, Ruby on Rails або Laravel. Вони забезпечують чисту структуру директорій, ORM для швидкого CRUD і готові модулі автентифікації, але важче масштабуються горизонтально, адже вся логіка змішана в одному процесі. Крім того, серверна візуалізація сторінок без SPA-механізмів означає перезавантаження після кожної дії, що знижує інтерактивність.

JAMstack-підхід із генерацією статичних сторінок у Gatsby чи Next.js і винесенням динаміки в serverless-функції дозволяє миттєво віддавати контент через CDN і не турбуватися про адміністрування серверів. Проблема з’являється, коли обсяг рецептів вимірюється десятками тисяч: час проходження build-процесу різко зростає, доводиться жонглювати кешами ISR і вебхуками, щоб підтримувати актуальність даних.

SPA, побудоване на React, Vue чи Svelte та пов’язане з REST- або GraphQL-API, забезпечує багатий клієнтський інтерфейс із миттєвими реакціями на оцінки чи коментарі, зручною мобільною версією у форматі PWA й потенціалом для подальшої інтеграції з React Native. Натомість початкова розробка потребує двох окремих команд – для фронтенду й бекенду – а питання SEO доводиться розв’язувати через серверний рендер або пререндеринг.

Нарешті, мікросервісна архітектура з контейнерами Docker і оркестрацією Kubernetes дає майже необмежене горизонтальне масштабування, можливість писати кожен сервіс тією мовою, що найкраще пасує задачі, та високу відмовостійкість. Разом із перевагами зростає складність мережевих викликів, виникає потреба у власному DevOps-відділі, а без кешування збільшується латентність.

З огляду на те, що контент на кулінарному порталі створюють самі користувачі й роблять це щодня, а інтерфейс має реагувати без перезавантаження, найбільш збалансованим видається підхід «SPA плюс окремий API» з можливістю поступово виокремлювати важкі підсистеми у мікросервіси. На етапі мінімально життєздатного продукту достатньо React-фронтенду з серверним рендером через Next.js, бекенду на Node.js і реляційної бази PostgreSQL, загорнутих у монолітний Docker-контейнер і розгорнутих через конвеєр GitHub Actions на орендованому VPS. Коли аудиторія зросте, службу автентифікації та рекомендативний модуль можна винести в окремі образи, додати Redis для кешування та налаштувати автоматичне масштабування. Подальший перехід до повноцінного Kubernetes, зберігання медіафайлів у S3-сумісному сховищі та розподіленої обробки потоків подій у Kafka й Spark забезпечить стабільність сервісу навіть за сотень тисяч одночасних користувачів, не жертвуючи швидкодією та гнучкістю розробки [6].

1.6.1 Обґрунтування та вибір найраціональнішого підходу до створення веб-ресурсу

Оптимальним шляхом реалізації платформи обміну кулінарними рецептами є повна розробка з нуля, коли і клієнтська, і серверна частини створюються спеціально під вимоги проєкту. Така стратегія усуває обмеження готових CMS і конструкторів, дозволяє одразу закласти односторінкову архітектуру з реактивним інтерфейсом, адаптивною версткою та чіткою схемою розмежування ролей користувачів. Розробник отримує свободу добирати інструменти, що найкраще відповідають очікуваному навантаженню: сучасний фреймворк React із серверним рендером через Next.js на фронтенді та легковажний, але масштабований бекенд на Node.js з GraphQL-шаром для гнучких запитів. Завдяки цьому можна безболісно інтегрувати механізми машинного навчання, покрокові відео, push-нотифікації чи офлайн-режим PWA, коли такі функції стануть актуальними.

Повна кастомізація бази даних у PostgreSQL дає змогу нормально моделювати складники рецептів, харчову цінність та тегування без сторонніх плагінів і ризику непередбачених конфліктів під час оновлень. Запуск проєкту у власних Docker-контейнерах із CI/CD-конвеєром забезпечує контроль версій, швидке розгортання на будь-якому хмарному середовищі та автоматизоване масштабування під час сезонних піків трафіку. Водночас закладається чітка стратегія безпеки: трафік передається виключно через HTTPS, критичні дані шифруються, а модульна структура коду спрощує аудит і подальші оновлення залежностей. У сумі це створює основу, яка легко еволюціонує разом з аудиторією і бізнес-метою, не потрапляючи до пастки технічного боргу, притаманного універсальним, але важко кастомізованим рішенням [7].

1.6.2 Послідовні етапи повного життєвого циклу проєктування, розгортання та довгострокового супроводу веб-ресурсу

На рисунку 1.5 схематично подано повний життєвий цикл вебсайту, який охоплює всі ключові кроки – від первинного формулювання вимог до тривалого супроводу вже функціонального ресурсу.

Рисунок 1.5 – Покрокова схема повного життєвого циклу створення, впровадження та подальшого супроводу вебсайту

Спершу команда визначає мету, аудиторію та перелік обов’язкових функцій майбутньої платформи, після чого переходить до детального проєктування логіки й інтерфейсів. Наступним етапом є безпосередня розробка, де програмісти перетворюють концепцію на працюючий код. Потім проводяться багаторівневі перевірки, що виявляють і виправляють помилки, аби забезпечити стабільність і безпеку сервісу. Після успішного тестування сайт розгортається у виробничому середовищі, стає доступним користувачам і проходить етап впровадження, що включає налаштування домену, хостингу та аналітики. Завершальна фаза – експлуатація та підтримка, коли розробники регулярно

оновлюють функціонал, встановлюють патчі безпеки й реагують на запити спільноти, гарантуючи довготривалу життєздатність проєкту [8].

1.7 Аргументація вибору технологічного стека та інструментів розробки

У розробленні вебсайту для обміну кулінарними рецептами доцільно спиратися на перевірений стек із відкритими стандартами, що гарантують сумісність на широкому спектрі пристроїв та хостинг-платформ. Базовими складовими такого стека є HTML5, CSS3, JavaScript (ES6+) і реляційна СУБД MySQL.

HTML5 обрано як ядро розмітки, оскільки він надає семантичні теги <article>, <section>, <nav> та <figure>, що дозволяють описувати рецепт, блок інгредієнтів і навігаційні елементи зрозуміло як для браузерів, так і для пошукових систем. Семантика безпосередньо поліпшує SEO й доступність: скринрідери легко інтерпретують структуру сторінки, а пошукові боти коректно індексують вміст. Крім того, HTML5 підтримує вбудовані медіатеги <video> та <audio>, які знадобляться для покрокових відео-інструкцій без потреби у сторонніх плагінах.

CSS3 відповідає за візуальне оформлення та адаптивне відображення. Завдяки медіазапитам (@media) можна забезпечити коректне масштабування верстки від смартфонів 320 px до настільних моніторів 4K, використовуючи Mobile First-підхід. Flexbox і CSS Grid дають змогу створити динамічну сітку карток рецептів, а властивості анімації (@keyframes, transition) дозволяють додати плавні ефекти без JavaScript-навантаження на процесор. У поєднанні з CSS-змінними легко реалізувати тему «світлий/темний» інтерфейс, що підвищує зручність для користувачів.

JavaScript (ES6+) забезпечує клієнтську інтерактивність. Сучасний синтаксис із модулями (import/export), async/await та Fetch API спрощує обробку форм публікації, динамічний фільтр рецептів і лайв-пошук без перезавантаження сторінки. Легка бібліотека або власний компонентний мікрофреймворк

дозволять поступово нарощувати функціонал, не втягуючись у складні екосистеми, що критично на етапі MVP. Підтримка Service Worker відкриває шлях до Progressive Web App: сайт зможе працювати офлайн, кешуючи популярні рецепти й медіа-ресурси. Для реалізації серверної логіки було обрано PHP, оскільки ця мова успішно поєднує зрілість екосистеми, низький поріг входу та величезну кількість перевірених на практиці бібліотек. По-перше, PHP залишається одним із найбільш поширених рішень на ринку веб-хостингу: майже кожен провайдер пропонує оптимізовані конфігурації LAMP/LEMP-стека, що спрощує розгортання і суттєво знижує витрати на інфраструктуру. По-друге, сучасні версії (від 7.4 і вище) забезпечують значне зростання продуктивності завдяки JIT-компіляції та поліпшеному управлінню пам’яттю, тому навіть під час пікових навантажень ресурс здатен працювати бездоганно.

Важливим чинником стала й наявність розвинених фреймворків – передусім Laravel, Symfony та CodeIgniter. Вони пропонують інструменти для захисту від поширених векторів атак (CSRF, SQL-ін’єкції, XSS), мають вбудовані ORM-шари й підтримують REST чи GraphQL-ендпоїнти без додаткового кодування «з нуля». Це дає змогу зосередитися на бізнес-логіці платформи: обробці рецептів, керуванні користувацькими ролями та генерації персональних стрічок. Розвинена спільнота забезпечує швидкий пошук відповідей на нетипові запитання, а регулярні патчі безпеки мінімізують ризик вразливостей.

З погляду інтеграції з фронтендом HTML5 / CSS3 / JavaScript PHP легко взаємодіє через JSON-API, а у разі потреби підтримує серверний рендер Blade-шаблонів для сторінок, критичних до SEO. Крім того, мова добре «дружить» з MySQL: використання розширених можливостей InnoDB та індексів дозволяє організувати швидкий пошук рецептів, підрахунок лайків і сортування за популярністю.

У підсумку вибір PHP виправданий комплексом аргументів: доступність хостингів, висока продуктивність сучасних версій, багатий набір фреймворків, що пришвидшує розробку, та перевірена роками стійкість до навантажень у проектах подібного масштабу. Це створює надійну основу для подальшого

розвитку платформи, забезпечуючи баланс між швидким запуском MVP і можливістю еволюціонувати до високонавантаженого сервісу. [10].

MySQL обрано для зберігання структурованих даних. Реляційна модель природно підходить для сутностей «Користувач», «Рецепт», «Інгредієнт», «Коментар» та «Оцінка», де чітко визначені зв’язки «один-до-багатьох» і «багато-до-багатьох». Інструменти реплікації та механізми індексування (зокрема повнотекстові індекси InnoDB) дозволяють масштабувати читання під час пікових навантажень, наприклад перед святами. Популярність MySQL гарантує велику кількість готових хостингових рішень, резервних стратегій і фреймворків для міграцій, що знижує витрати на супровід. До того ж, open-source-ліцензія спрощує правові аспекти і дає змогу тонко налаштовувати конфігурацію під конкретні потреби проєкту.

У сумі вибраний набір технологій забезпечує легке освоєння командою, гнучкість у поступовому зростанні функціоналу та високу сумісність із недорогими серверами. Стек HTML5 + CSS3 + JavaScript формує швидкий, доступний і SEO-дружній фронтенд, а MySQL гарантує надійне зберігання даних і можливість горизонтального масштабування бази в міру розширення кулінарної спільноти. [11].

1.8 Обґрунтований вибір інструментального середовища для розробки вебсайту

У таблиці 1.1 подано зведене порівняння чотирьох популярних середовищ розробки, що найчастіше застосовуються для роботи зі стеком HTML5 / CSS3 / JavaScript / PHP / MySQL.

Таблиця 1.1 – Порівняльні характеристики інтегрованих середовищ веб-розробки

Характеристика	Visual Studio Code	WebStorm	Atom	Sublime Text
Ціна	Безкоштовний	Платний	Безкоштовний	Платний
Мови програмування	Багатомовна	HTML, CSS, JS	Багатомовна	Багатомовна
Розширення / плагіни	Наявні	Наявні	Наявні	Наявні
Налагодження	Вбудоване	Вбудоване	Використання плагінів	Використання плагінів
Системні вимоги	Невисокі	Невисокі	Середні	Невисокі
Спільнота	Велика	Менша	Велика	Менша

Під час вибору інтегрованого середовища розроблення (IDE) для створення й підтримки кулінарного вебсайту доцільно спиратися на три групи критеріїв: функціональність «з-коробки», доступність екосистеми розширень і повна вартість володіння (TCO) у довгостроковій перспективі.

Visual Studio Code посідає перше місце у щорічних опитуваннях Stack Overflow (понад 70 % професійних розробників використовують його щодня) і пропонує понад 60 тисяч безкоштовних розширень для мов, фреймворків, відлагоджувачів та DevOps-інструментів.

Розширена підтримка Git, GitHub Copilot і вбудований термінал роблять VS Code універсальним вибором для команд, що працюють з JavaScript, PHP і SQL. Ліцензія MIT із відкритим вихідним кодом знімає обмеження на кількість інсталяцій у межах проєкту.

WebStorm вирізняється найглибшою інтеграцією з екосистемою JavaScript: інтелектуальні підказки для React і Vue, рефакторинги, профілювання продуктивності й вбудований інспектор безпеки пакетів npm. З жовтня 2024 р. IDE стала безкоштовною для некомерційного використання, тоді як комерційна підписка зберігається на умовах JetBrains Toolbox.

Для стартапу чи навчального проєкту це зменшує бар’єр входу, але в корпоративному середовищі потрібно закладати щорічні витрати на ліцензії.

Atom більше не розвивається: GitHub офіційно «засунсетив» редактор 15 грудня 2022 р., архівувавши репозиторій і припинивши постачання оновлень безпеки

Відсутність підтримки робить його неприпустимим вибором для сучасного веб-проєкту, особливо якщо планується публічний деплой і активна експлуатація.

Sublime Text залишається швидким і мінімалістичним, підтримує десятки мов, багатовіконний режим та «розумне» мультивиділення, але повнофункціональна версія коштує 99 USD за персональну ліцензію; безкоштовна оцінювальна копія періодично показує нав’язливі нагадування про купівлю

Плагінів помітно менше, ніж у VS Code, а відсутність інтегрованих засобів відлагодження для фронтенду змушує встановлювати сторонні пакети.

Для проєкту, що стартує сьогодні й розраховує на швидке розширення команди, найоптимальнішим є Visual Studio Code. Він безкоштовний, має величезний маркетплейс, регулярно оновлюється й не накладає ліцензійних обмежень. Якщо потрібна глибока «коробкова» підтримка складних JavaScript-фреймворків і команда готова оплачувати передплату, WebStorm може стати професійним доповненням до VS Code, наприклад, для старших розробників або рев’ю коду. Sublime Text варто розглядати як легку «запасну» IDE для коротких правок, а Atom – доцільно виключити з технічного стеку через припинення супроводу. [12].

1.9 Висновок до першого розділу

У першому розділі здійснено всебічний огляд кулінарного сегмента мережі, що виявив сталий попит на україномовну платформу з адаптивним дизайном, розширеною мультимедіа-подачею та глибокою соціальною взаємодією. Аналіз глобальних і вітчизняних сервісів показав, що наявні рішення

або недостатньо локалізовані, або перевантажені другорядним контентом, що створює вікно можливостей для запуску нового ресурсу. Порівняння двох найближчих конкурентів – «Пательні» та «Кукорами» – дало змогу визначити ключові напрями удосконалення: обов'язкові зображення й відео, оптимізована мобільна навігація, збалансований обсяг додаткової інформації, уніфікована типографіка та гейміфіковані механізми спільнотної активності.

На підставі цих висновків сформульовано комплексні функціональні й нефункціональні вимоги до майбутнього сайту, акцентовано на безпеці персональних даних, доступності, багатомовності й можливості горизонтального масштабування. Проведено ідентифікацію основних ролей – від гостьового відвідувача до адміністратора – і описано типові сценарії їхньої взаємодії, що гарантує адекватне розмежування прав доступу.

Оцінка методів розробки показала, що найкращим шляхом є створення SPA-платформи з окремим API, реалізованої на React + Next.js, PHP-бекенді та MySQL-базі, із подальшим потенціалом міграції критичних підсистем у мікросервіси. Вибір вільних і зрілих технологій HTML5, CSS3, JavaScript та PHP забезпечує швидкий старт, високу продуктивність і широкі можливості кастомізації, тоді як MySQL відповідає за надійне зберігання структурованих даних.

Нарешті, порівняння інструментів розробки засвідчило перевагу Visual Studio Code як основного середовища завдяки безкоштовній ліцензії, багатій екосистемі плагінів і потужним інтеграціям із системами контролю версій; WebStorm рекомендовано як професійне доповнення для глибокого статичного аналізу коду.

Сукупність проведених досліджень і прийнятих рішень формує міцну методологічну основу для подальшої розробки Вебсайту – платформи, здатної задовольнити запити широкого кола користувачів і забезпечити стабільний розвиток сервісу в умовах швидко зростаючого кулінарного контенту.

РОЗДІЛ 2. КОНЦЕПТУАЛЬНЕ ТА ТЕХНІЧНЕ ПРОЄКТУВАННЯ ВЕБСАЙТУ

2.1 Логічна та технічна архітектура вебсайту

Для проєкту вебсайту доцільно обрати класичну трирівневу архітектуру, у якій кожен шар має чітко окреслену відповідальність.

Рівень клієнта (Client)

На цьому шарі працює браузер або мобільний PWA-оболонка. Він отримує HTML-розмітку, стилі та скрипти, відтворює інтерфейс і приймає дії користувача. Сюди належать: рендер карток рецептів, робота фільтрів без перезавантаження, надсилання форм публікації, а також локальне кешування статичних ресурсів і офлайн-режим через Service Worker.

Рівень логіки (Application Logic)

Середній шар поєднує бізнес-процеси й технічні сервісні функції. До нього входять контролери API, модулі авторизації, валідатори, шаблонізатор або JSON-серіалізатор і ядро рекомендаційного алгоритму. Саме тут реалізуються правила платформи: перевірка прав доступу, розрахунок рейтингу страв, формування списку покупок, обробка зображень та кешування частих запитів.

Рівень даних (Data)

Нижній шар репрезентує систему зберігання: реляційну базу MySQL / PostgreSQL і, за потреби, об’єктне сховище для великих медіафайлів. Тут розміщені таблиці користувачів, рецептів, інгредієнтів, коментарів і лайків, налаштовані індекси для швидкого пошуку та механізми резервного копіювання. Логічний рівень звертається до бази через ORM або SQL-запити, отримує дані й повертає їх клієнтові у вигляді готової відповіді.

Такий поділ дозволяє незалежно розвивати інтерфейс, бізнес-логіку й підсистему даних, спрощує тестування, підвищує безпеку та полегшує горизонтальне масштабування кожного шару у разі збільшення аудиторії.

Трирівнева архітектура (див. рис. 2.1) поділяє вебсайт на три логічно незалежні шари, кожен із яких виконує власні функції та взаємодіє із сусідніми через чітко визначені інтерфейси.

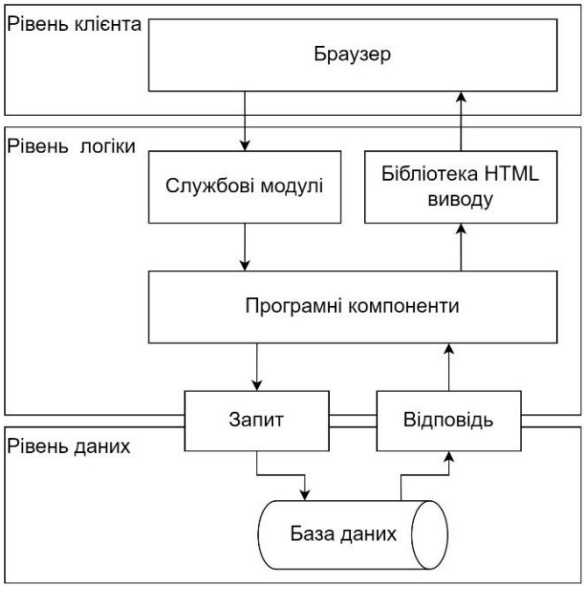

Рисунок 2.1 – Логічна схема трирівневої організації вебсайту

1. Рівень клієнта (presentation layer).

На верхньому рівні працює браузер користувача. Він надсилає HTTP-запити й відображає отриману розмітку, стилі та сценарії. Уся взаємодія з відвідувачем відбувається саме тут: рендеряться HTML-сторінки, виконується JavaScript, ініціюється AJAX- або Fetch-запити для динамічних оновлень. Таким чином, клієнтська частина відповідає за презентацію даних і збір введеної інформації (форми публікації рецептів, фільтри пошуку тощо).

2. Рівень логіки (application layer).

Середній шар реалізує бізнес-процеси. Він складений із програмних компонентів, службових модулів і бібліотеки HTML-виводу. Службові модулі опікуються технічними завданнями — автентифікацією, кешуванням, валідацією даних, захистом від CSRF/XSS. Бібліотека HTML-виводу (темплейт-двигун або API-серіалізатор) перетворює структуровані дані на готову розмітку або JSON-відповідь. Центральні програмні компоненти інкапсулюють бізнес-логіку: обробляють запити створення/редагування рецептів, застосовують

фільтри, підраховують рейтинги й формують рекомендації. Саме тут ухвалюються рішення, які дані необхідно отримати або змінити в базі.

3. Рівень даних (data layer).

Нижній шар складається з СУБД та механізмів доступу до неї. Будь-який запит, що потребує зчитування чи модифікації інформації, проходить через інтерфейс «Запит/Відповідь»: бізнес-компонент формує SQL- або ORM-запит, база повертає результати, після чого логічний рівень перетворює їх у зручний формат для клієнта. База даних зберігає сутності «Користувач», «Рецепт», «Інгредієнт», «Коментар», «Оцінка» та їхні зв’язки, забезпечуючи цілісність і швидкий пошук завдяки індексам.

Коли користувач у браузері надсилає запит (наприклад, відкриває рецепт), клієнтський рівень передає його на застосунковий шар. Там службові модулі перевіряють права доступу, головні компоненти формують запит до бази, отримують потрібні записи й через HTML-бібліотеку або API-серіалізатор відправляють відповідь назад у браузер. Завдяки чіткому розподілу обов’язків зміни на одному рівні (наприклад, заміна СУБД або редизайн інтерфейсу) мінімально впливають на інші, що спрощує масштабування, тестування та підтримку системи. [13].

2.2 Розробка ієрархічної схеми та логічної навігаційної структури вебсайту

Структурна організація вебресурсу відтворює логічне розміщення всіх його сторінок, завдяки чому розробнику легко орієнтуватися, де розташовані ключові розділи й яким чином забезпечуються переходи між ними [14]. Схему цієї ієрархії для весайту подано на рисунку 2.2.

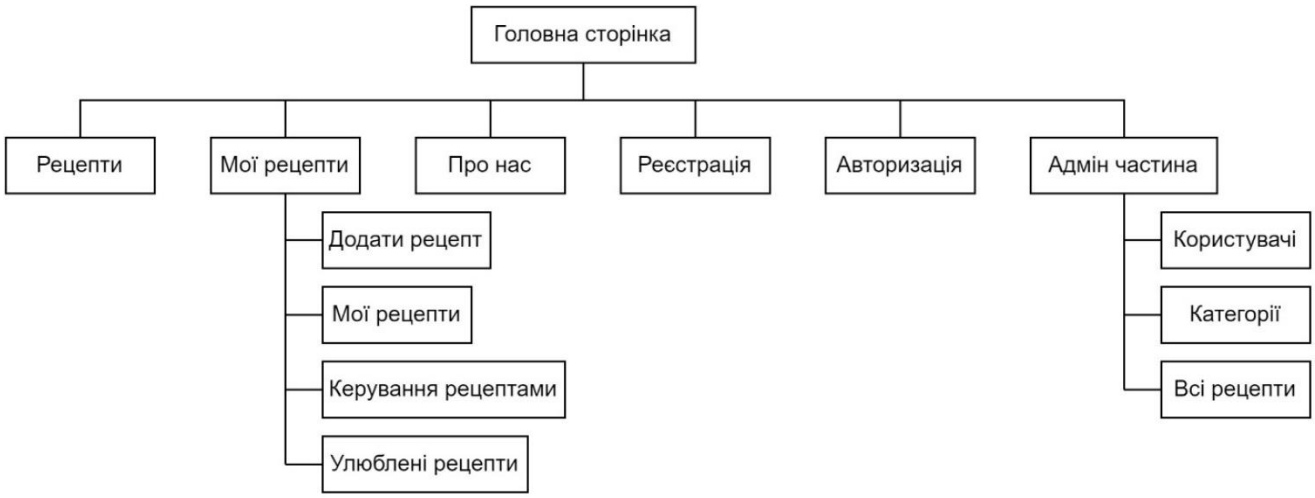

Рисунок 2.2 – Структурна схема вебсайту

Структурна схема вебсайту побудована за ієрархічним принципом, у центрі якого розташовано головну сторінку – точку входу для всіх відвідувачів. Від неї відгалужуються шість основних розділів верхнього рівня.

1. «Рецепти». Це публічний каталог, де без реєстрації можна переглядати, шукати й фільтрувати всі доступні страви.

2. «Мої рецепти». Розділ відображається лише після входу в систему і складається з чотирьох підсторінок:

- «Додати рецепт» – форма створення нового запису з полями для назви, опису, списку інгредієнтів і фото;
- «Мої рецепти» – перелік усіх публікацій поточного користувача з опціями редагування;
- «Керування рецептами» – розширена панель, де автор може масово оновлювати або видаляти власні страви;
- «Улюблені рецепти» – збережені вподобані страви інших авторів.

3. «Про нас». Інформаційна сторінка з описом місії платформи, контактами та політикою конфіденційності.

4. «Реєстрація». Форма створення облікового запису із заповненням особистих даних або швидким входом через соціальні мережі.

5. «Авторизація». Вікно входу для вже зареєстрованих користувачів із можливістю відновлення пароля.

6. «Адмін частина». Секція доступна лише адміністраторам і містить три підрозділи:

- «Користувачі» – керування акаунтами, призначення ролей і блокування порушників;
- «Категорії» – створення, перейменування й сортування кулінарних рубрик;
- «Всі рецепти» – глобальний список публікацій з інструментами модерації та статистикою переглядів.

Такий поділ забезпечує чітке розмежування прав і спрощує навігацію: гості мають доступ лише до загального каталогу й інформаційних сторінок, автори керують власним контентом у спеціальному блоці, а адміністрація отримує окремі інструменти для повного контролю над платформою.

2.3 Проєктування та реалізація логічної й фізичної моделі бази даних для вебсайту

Під час створення Вебсайту першочерговим завданням стало грамотно впорядкувати великий масив взаємопов’язаних відомостей – від даних профілю до детальних карток рецептів. Без чіткої схеми сховища не вдалось би забезпечити ні високу швидкість відгуку, ні зручні фільтри, ні масштабування проєкту в майбутньому. Саме тому на етапі концептуального моделювання було закладено три фундаментальні принципи: нормалізацію структури, мінімізацію дублювання та підтримку очевидних для розробника зв’язків між сутностями.

Однією зі щоденних операцій на платформі є додавання й оновлення інформації в базі. Коли автор наповнює форму нового рецепта або читач зберігає страву до «улюблених», веб-сайт генерує транзакції, що мають бути виконані максимально ефективно. Тож ієрархію таблиць проєктували так, аби кожен запит містив якомога менше приєднань, а найпопулярніші вибірки – список категорій, топ-рецепти, останні коментарі – кешувалися за допомогою індексів.

Провівши тематичний аналіз і зібравши функціональні вимоги, команда виокремила шість ключових об’єктів даних:

- Users (користувачі). Фіксує облікові записи, ролі, обрані налаштування й дату реєстрації.
- Recipes (рецепти). Містить заголовок, опис, список інгредієнтів, кроки приготування, час, калорійність і посилання на зображення.
- Saved_recipes (улюблені). Відстежує, які саме страви конкретний авторизований користувач додав до власної колекції, щоб швидко знаходити їх пізніше.
- Comments (коментарі). Дає змогу обговорювати страви, уточнювати технологічні деталі та оцінювати рецепти за суб'єктивними критеріями.
- Categories (категорії). Зберігає таксономію – від базових «Перші страви» до нішевих «Безглютенові десерти», які створює чи редагує адміністратор.
- Recipe_categories (міст між рецептами й категоріями). Реалізує зв'язок «багато-до-багатьох» і дозволяє одній страві водночас належати, наприклад, до «Італійська кухня» та «Вегетаріанське».

Таке дроблення усуває дублювання тексту та полегшує обслуговування: додавання нової категорії не вимагає змін у таблиці рецептів, а зняття рецепта з публікації не впливає на структуру таксономії. Крім того, воно створює передумови для майбутнього розширення – наприклад, уведення тегів «без цукру» чи «халяль» без болючих міграцій.

Після логічного моделювання перейшли до фізичної схеми. В таблиці users (див. табл. 2.1) первинний ключ id індексується автоматично, що прискорює пошук профілю під час логіна. Поле admin типу TINYINT з двійковим значенням відрізняє читачів від адміністраторів, а за необхідності дозволяє додати проміжні ролі – редактора чи модератора – без переписування коду. Логіни обмежено 20 символами, щоб запобігти зловживанням, а для паролів використано хеш-рядок до 255 символів, що сумісно з алгоритмами bcrypt/argon2.

Таблиця 1.1 – Структура реляційної таблиці «users»

Назва поля	Тип даних	Призначення
id	int (255)	ID користувача
admin	tinyint (1)	Роль користувача
login	varchar (20)	Ім’я користувача
email	varchar (50)	Електрона пошта користувача
password	varchar (255)	Пароль користувача
create_date	datetime	Дата реєстрації користувача

У таблиці 2.2 подано детальну схему справочної таблиці recipes, куди потрапляє увесь масив відомостей про страви, що їх публікують авторизовані користувачі вебсайту. Кожний запис у цій таблиці репрезентує окремий кулінарний витвір і містить не лише базові атрибути, як-от назва та короткий опис, а й допоміжні метадані: склад інгредієнтів, тривалість приготування, калорійність, посилання на зображення й позначку автора. Таким чином «recipes» виконує роль центрального сховища, до якого звертаються модулі пошуку, рекомендаційний алгоритм і система коментарів.

Таблиця 2.2 – Структура реляційної таблиці «recipes»

Назва поля	Тип даних	Призначення
id	int (255)	ID рецепту
user_id	int (255)	ID користувача
title	varchar (255)	Назва рецепту
ingredients	text	Інгредієнти рецепту
preparation	text	Інструкція приготування
image	varchar (255)	Назва та формат зображення
published	tinyint (1)	Статус публікації
create_date	datetime	Дата додавання рецепту

У таблиці 2.3 подано докладний опис службової таблиці saved_recipes, що виконує роль «закладок» для зареєстрованих учасників спільноти. Саме тут система фіксує, які саме страви користувач забажав швидко знаходити надалі, натиснувши кнопку «Додати до улюблених». Запис у цій таблиці з'являється щоразу, коли авторизований відвідувач позначає рецепт як вподобаний; надалі цей запис дає змогу миттєво сформувати персональну добірку й підрахувати загальну популярність страви без обтяжливих вибірок у базовій таблиці recipes.

Структура передбачає мінімальний, але достатній набір полів. Первинний ключ id типу INT однозначно ідентифікує кожну операцію збереження. Колонка user_id (також INT) посилається на унікальний запис у таблиці users, забезпечуючи швидку побудову списку «мої улюблені» для конкретного профілю. Поле recipe_id зберігає посилання на страву з таблиці recipes; завдяки зовнішнім ключам база підтримує цілісність: якщо рецепт видаляють або приховують, відповідні закладки автоматично очищаються. Нарешті, атрибут create_date типу DATETIME фіксує момент додавання рецепта в обране – це дає змогу сортувати колекцію за давністю або будувати аналітику трендів.

Таблиця 2.3 – Структура реляційної таблиці «saved_recipes»

Назва поля	Тип даних	Призначення
id	int (255)	ID збереженого рецепту
user_id	int (255)	ID користувача
recipe_id	int (255)	ID рецепту
create_date	datetime	Дата збереження рецепту

У таблиці 2.4 представлено розгорнуту специфікацію службової таблиці comments, що виконує роль історичного журналу обговорень під кожним рецептом на вебсайті. Саме цей об'єкт даних фіксує всі відгуки, поради й уточнення, залишені авторизованими користувачами, а також пов'язує їх із конкретними стравами й профілями дописувачів. Така централізація дає змогу не тільки швидко виводити розділ «Коментарі» для кожної сторінки рецепта, а й

застосовувати модерування, підрахунок активності та аналітику популярності окремих тем.

Схема передбачає п’ять базових полів. Первинний ключ id (INT) гарантує унікальність кожного допису й забезпечує оперативне адресне звернення до запису. Колонка recipe_id (INT) реалізує зовнішній ключ на таблицю recipes, завдяки чому всі репліки чітко прив’язуються до відповідної страви, а при видаленні рецепта система може автоматично очищати залежні записи. Поле user_id (INT) з’єднує коментар із його автором у таблиці users, що важливо для відстеження активності, відображення аватарів і застосування санкцій у разі порушення правил. Атрибут comment (TEXT) містить безпосередній зміст повідомлення, підтримує ризиковані символи через екранування й може бути індексований для пошуку за ключовими словами під час фільтрації обговорень. Нарешті, create_date (DATETIME) фіксує точний момент публікації, дозволяючи сортувати репліки хронологічно або будувати теплову карту найбільш «гарячих» періодів дискусій.

Таблиця 2.4 – Структура реляційної таблиці «comments»

Назва поля	**Тип даних**	**Призначення**
1	**2**	**3**
id	int (255)	ID коментаря
recipe_id	int (255)	ID рецепту
user_id	int (255)	ID користувача
comment	text	Текст коментаря
create_date	datetime	Дата додавання коментаря

Така структура забезпечує гнучкість вибірок – наприклад, отримання всіх дописів конкретного користувача чи швидке підвантаження останніх десяти відгуків до рецепта – і водночас підтримує цілісність даних завдяки системі зовнішніх ключів і каскадних обмежень.

У таблиці 2.5 описано логічну «сполучну ланку» між стравами й їхнім тематичним класифікатором – службову таблицю recipe_categories. Вона

реалізує зв’язок «багато-до-багатьох», завдяки якому один рецепт може одночасно належати до кількох рубрик («Сніданки», «Веганські», «Італійська кухня»), а кожна категорія, своєю чергою, охоплює безліч різних страв. Інакше кажучи, без цієї допоміжної таблиці було б неможливо організувати точний фільтр меню й побудувати роздільні стрічки на кшталт «ТОП-10 десертів літа» чи «Швидкі обіди до 30 хвилин».

Структура мінімалістична, проте містить усе необхідне для підтримання цілісності й високої продуктивності запитів. Поле id (INT) слугує первинним ключем, що надає кожному зв’язку унікальний ідентифікатор та прискорює звернення додатка під час операцій редагування або видалення. Колонка recipe_id (INT) посилається на таблицю recipes, дозволяючи миттєво знайти всі категорії певної страви. Відповідно, атрибут category_id (INT) зв’язує запис із конкретною рубрикою з довідника categories; це необхідно для виведення переліку «всі рецепти зі значком “Пісне”» чи формування дашборду популярності рубрик. Нарешті, стовпець create_date (TIMESTAMP) автоматично фіксує момент, коли автор (або адміністратор) прив’язав страву до рубрики. Така часовідмітка дає змогу відстежувати динаміку розширення контенту, будувати графік активності кухарів і навіть скасовувати останні зміни, якщо вони виявилися помилковими.

Таблиця 2.5 – Структура реляційної таблиці «recipe_categories»

Назва поля	Тип даних	Призначення
id	int (255)	ID рецепт-категорії
recipe_id	int (255)	ID рецепту
categoru_id	int (255)	ID категорії
create_date	timestamp	Дата додавання категорії до рецепту

Запроваджені зовнішні ключі з каскадними правилами ON DELETE CASCADE гарантують, що під час вилучення рецепта або категорії відповідні перехресні записи автоматично зникнуть і система не накопичуватиме «сирітські» рядки. Додатковий складений індекс за парою (recipe_id, category_id)

робить найчастіші операції вибірки – пошук усіх страв у межах однієї рубрики чи перевірку, чи належить рецепт до певної категорії – практично миттєвими навіть за великого обсягу даних.

У таблиці 2.6 подано розгорнуту специфікацію довідника categories, котрий слугує офіційним переліком тематичних рубрик, що їх створює або редагує адміністратор вебсайту. Завдяки цьому довіднику всі рецепти впорядковуються за зрозумілими мітками –- від базових «Супи» й «Десерти» до нішевих «Кето-страви» чи «Безглютенова випічка». Такий централізований словник забезпечує узгодженість термінів у всіх підрозділах сайту, спрощує фільтрацію контенту й унеможливлює появу дубльованих або хибно написаних рубрик.

Схема побудована максимально лаконічно, аби пришвидшити вибірки й мінімізувати споживання пам’яті. Первинний ключ id типу INT(10) однозначно ідентифікує кожну категорію та використовується як зовнішнє посилання в таблиці recipe_categories. Поле name – це рядок VARCHAR(50), достатній для більшості назв кухонь і стилів приготування; додатковий унікальний індекс на цьому полі запобігає створенню рубрик-двійників, а також прискорює пошук під час автопідказок у формі додавання рецепта.

Таблиця 2.6 – Структура реляційної таблиці «categories»

Назва поля	Тип даних	Призначення
id	int (10)	ID категорії
name	varchar (50)	Назва категорії

Завдяки компактності та чіткій типізаційній політиці довідник легко масштабувати: за потреби можна додати нові стовпці (наприклад, «опис» або «іконка») без ризику порушити наявні зв’язки, а поточна структура лишається оптимальною для швидкої роботи запитів навіть за сотень тисяч прив’язок рецептів до рубрик.

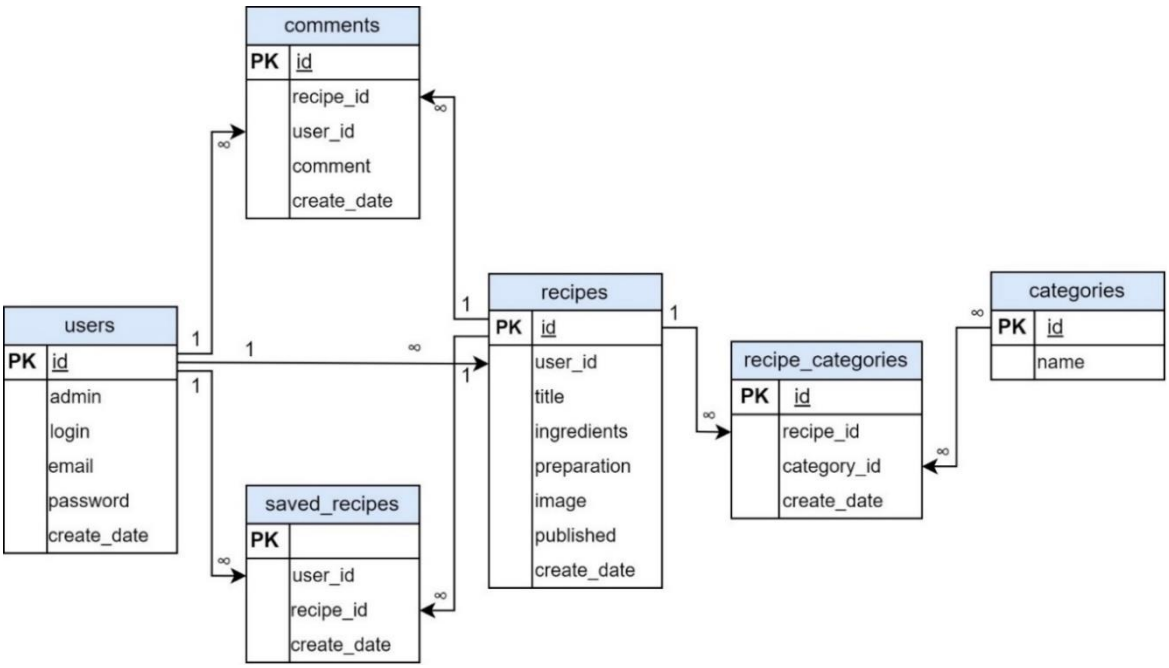

Рисунок 2.3 – Модель бази даних вебсайту

Після того як перелік інформаційних сутностей – користувачі, рецепти, категорії та допоміжні журнальні таблиці – був остаточно затверджений, команда перейшла до побудови повноцінного концептуального рівня бази даних. На цьому етапі завдання полягає не просто у фіксації назв полів: необхідно сформувати цілісну «карту» того, як саме об’єкти співіснують, якими клавішами пов’язані й у який спосіб одна зміна даних відлунює в системі загалом. Інженери створили схематичне подання (див. рис. 2.3), у якому графічно позначені сутності, їхні атрибути, а також кардинальність кожного зв’язку. Така візуалізація відіграє роль дорожньої мапи для програмістів, адміністраторів СУБД та навіть для аналітиків, що перевіряють коректність бізнес-правил.

Усього модель містить шість логічних об’єктів. Таблиця users є «точкою відліку» для майже всіх інших зв’язків: один запис про користувача може бути джерелом багатьох рецептів (recipes), численних коментарів (comments) і довільної кількості «закладок» у таблиці saved_recipes. Отже, тут реалізується відношення «один-до-багатьох»: одиничний профіль → множина дочірніх записів. Завдяки зовнішньому ключу user_id у каскадному режимі видалення гарантовано, що після деактивації облікового запису всі його коментарі та улюблені страви не залишаться «сиротами».

Друга група зв’язків стосується рецептурного контенту. Одна страва, представлена в таблиці recipes, природно породжує безліч відгуків, а інколи – сотні збережень у різних користувацьких добірках. Це знову «один-до-багатьох», але вже з точки зору самої страви. Крім того, кожен рецепт може потрапити одразу до кількох рубрик («Веганське», «Середземноморська кухня», «Менше 30 хвилин»). Щоб не дублювати назви категорій у самій таблиці recipes, застосовано проміжну сутність recipe_categories. Її роль – гнучко реалізувати зв’язок «багато-до-багатьох»: багато рецептів ↔ багато тематичних міток. У структурі ця «перехрестя» позначена двома зовнішніми ключами (recipe_id, category_id) і власним первинним ідентифікатором, що полегшує адресне звернення та статистичні вибірки.

Побудова такої моделі дає одразу декілька переваг. По-перше, дані зберігаються із суворою нормалізацією: повторювана інформація, наприклад назва категорії, знаходиться в єдиному екземплярі в таблиці categories; це зменшує обсяг сховища й мінімізує ризик помилок під час редагування. По-друге, чітко визначені зовнішні ключі забезпечують референційну цілісність – СУБД не дозволить додати коментар, який не посилається на існуючий рецепт, або зберегти в «улюблені» неіснуючу страву. По-третє, стрілки кардинальності («1», «∞») у схемі одразу підказують інженерам оптимальну стратегію індексування: там, де множина записів очікує швидких вибірок (наприклад, коментарі за recipe_id), доречно створювати складені чи часткові B-tree-індекси.

Таким чином, концептуальна модель, яка відображена на рисунку 2.3, є фундаментальною опорою для фізичної реалізації сховища. Вона гарантує, що база даних вебсайту залишатиметься гнучкою для розширення, стійкою до логічних помилок і продуктивною навіть у разі різкого збільшення кількості користувачів або обсягу кулінарного контенту.

2.4 Розробка користувацьких сценаріїв і деталізація модульної організації вебсайту

Поведінковий шар вебресурсу визначає, як система реагує на кожен клік, запит чи подію з боку відвідувача. Іншими словами, це набір правил і маршрутизацій, котрі здійснюють зв’язок між графічним інтерфейсом і сервісною логікою, що працює «під капотом». Для Вебсайту було побудовано багатошарову модульну схему, у якій кожна група файлів виконує строго регламентовану роль, аби спростити супровід і забезпечити масштабованість.

Навігаційний блок

- index.php ініціює фронтовий маршрут «/», формує «герой-секцію» з пошуком та передає перші дев’ять трендових рецептів, підвантажених через Ajax.
- about_us.php містить не тільки опис місії й команди, а й інтерактивний FAQ; відповіді зберігаються у базі та за потреби змінюються модератором без редеплою.
- search.php обробляє GET-параметри, делегує фільтрацію до recipes_controller.php і повертає пагінацію результатів; у разі порожньої вибірки формується блок рекомендацій на основі популярних категорій.

Модуль аутентифікації

- login.php генерує csrf-токен, зберігає його в сесії та пересилає у форму; після надсилання перевіряє токен, шукає користувача за email, верифікує пароль через password_verify().
- register.php крім базових полів пропонує підтвердити правила спільноти; успішна реєстрація викликає сервісну подію UserRegistered, що відправляє welcome-лист.
- logout.php завершує сесію, чистить cookie «remember-me» і робить 302-редирект на головну.

Керування рецептами

- all_recipes.php виводить стрічку з можливістю сортувати за датою, популярністю чи тривалістю приготування; сортування зберігається у LocalStorage для повторних відвідувань.
- add_recipe.php використовує drag-and-drop-зону для zображень, виконує валідацію файлів (тип, розмір), а потім надсилає FormData у recipes_controller.php.
- edit_recipe.php перевіряє приналежність рецепта поточному автору, інакше повертає 403.
- favorite_recipes.php підтягує збережені страви за user_id і виводить їх плиткою; якщо колекція порожня, пропонує додати щось із ТОП-10.
- manage_recipes.php працює як дашборд автора: масові операції «зняти з публікації», «оновити фото», «видалити назавжди».
- my_recipes.php показує статистику переглядів і лайків у реальному часі через websocket-канал.
- single_recipe.php об’єднує контент рецепта, галерею, форму коментарів і блок схожих страв на основі тегів.

Адміністративна зона

- add_category.php / edit_category.php дозволяють керувати таксономією; перед видаленням категорії скрипт шукає рецепти, прив’язані до неї, і пропонує вибрати заміну.
- index_users.php надає таблицю користувачів із live-пошуком, експортом у CSV і кнопкою «заблокувати на 7 днів».
- published_recipes.php та unpublished_recipes.php відображають чергу модерації; адміністратор змінює статус, додає причину відмови, що одразу відправляється автору e-mail-повідомленням.
- edit_user_recipes.php відкриває конкретний рецепт будь-якого автора й дозволяє виправити грубі помилки розмітки Markdown перед повторною публікацією.

Доступ до даних

- connect.php зчитує параметри з .env, ініціює PDO-з'єднання з параметрами ATTR_EMULATE_PREPARES = false.
- db.php містить універсальні методи select, insert, update, delete з обгорткою для обробки винятків; додає логування повільних запитів у файл slow.log.

Шаблонізатори

- header.php та footer.php реалізують DRY-підхід, підключаючи єдиний набір скриптів і стилів; у header підтягується CSRF-токен у meta-тегу для JS-запитів.
- comments.php рендерить відповіді вкладеною структурою, додаючи кнопку «Згорнути ланку» для довгих дискусій.

Контролери логіки

- admin_controller_categories.php перевіряє, чи має користувач прапорець admin=1; у разі спроби доступу без прав повертає 401 JSON-відповідь.
- admin_controller_recipes.php обробляє масові дії, використовуючи транзакцію, аби статуси всіх вибраних рецептів змінилися атомарно.
- admin_controller_users.php підтримує призначення ролей за допомогою ENUM-поля role із варіантами user, moderator, admin.
- comments_controller.php застосовує фільтр нецензурних слів і rate-limit – не більше трьох коментарів за хвилину з однієї IP-адреси.
- recipes_controller.php виконує санітизацію Markdown і генерує підсумковий HTML через Parsedown; зображення проходять ресайз до 1280 px.
- users_controller.php відповідає за реєстрацію, авторизацію й відновлення пароля; відсилання листа відбувається через чергу SMTP, щоби не блокувати відповідь серверу.

Комбінація цих файлів утворює узгоджену поведінкову модель: будь-яка дія користувача – від кліку «зберегти рецепт» до запиту адміністратора «додати категорію» – проходить крізь шар контролерів, потім до бази даних і назад до шаблону, гарантуючи коректну реакцію й цілісність інформації на кожному етапі роботи вебсайту.

На рисунку 2.4 подано розгорнуту логіку обміну даними всередині адміністративного ядра Вебсайту. Центральною точкою старту є файл index.php: після ініціалізації базових залежностей – він під’єднує connect.php для встановлення безпечного з’єднання з СУБД і підтягує універсальну обгортку db.php з методами CRUD – скрипт перевіряє, чи поточний користувач володіє правами адміністратора. У разі позитивної верифікації відбувається маршрутизація до одного з трьох функціональних розділів бек-офісу.

Перша гілка веде до index_users.php, що відображає перелік зареєстрованих профілів. Ця сторінка «живиться» діями контролера admin_controller_users.php, котрий, у свою чергу, оперує тими ж connect.php та db.php. Контролер забезпечує три ключові операції: перегляд реквізитів користувачів, зміна їх ролі (наприклад, підвищення до модератора) та видалення акаунтів із каскадним очищенням залежних коментарів або улюблених рецептів.

Друга гілка переспрямовує адміністратора на categories.php. За логікою діаграми цей інтерфейс обслуговує admin_controller_categories.php, який відповідає за створення нових кулінарних рубрик, редагування назв і їх видалення. Оскільки будь-яка зміна таксономії впливає на пошукову видачу та фільтри користувачів, контролер працює у транзакційному режимі: або застосовує всю серію змін, або відкочує їх у разі помилки, зберігаючи цілісність довідника.

Третя й найрозгалуженіша гілка стосується модерації контенту. Залежно від статусу публікації рецепта index.php спрямовує запит до published_recipes.php чи unpublished_recipes.php. Обидві сторінки використовують admin_controller_recipes.php, який дозволяє змінювати стан видимості страви (опублікувати, приховати, видалити), а також коригувати ключові метадані, якщо автор припустився критичних помилок форматування. І тут, і в попередніх модулях усі запити до бази проходять через універсальну пару connect.php + db.php, що забезпечує єдину точку управління з’єднаннями та мінімізує ризик SQL-ін’єкцій.

Загальна структура схеми чітко відображає принцип інверсії керування: сторінки-представлення (*.php) не містять «важкої» бізнес-логіки, а делегують її

вузькоспеціалізованим контролерам. Завдяки цьому будь-яке оновлення, наприклад розширення механізму модерації або введення нових ролей, локалізується в межах відповідного контролера, не зачіпаючи інші гілки адміністрування. [17].

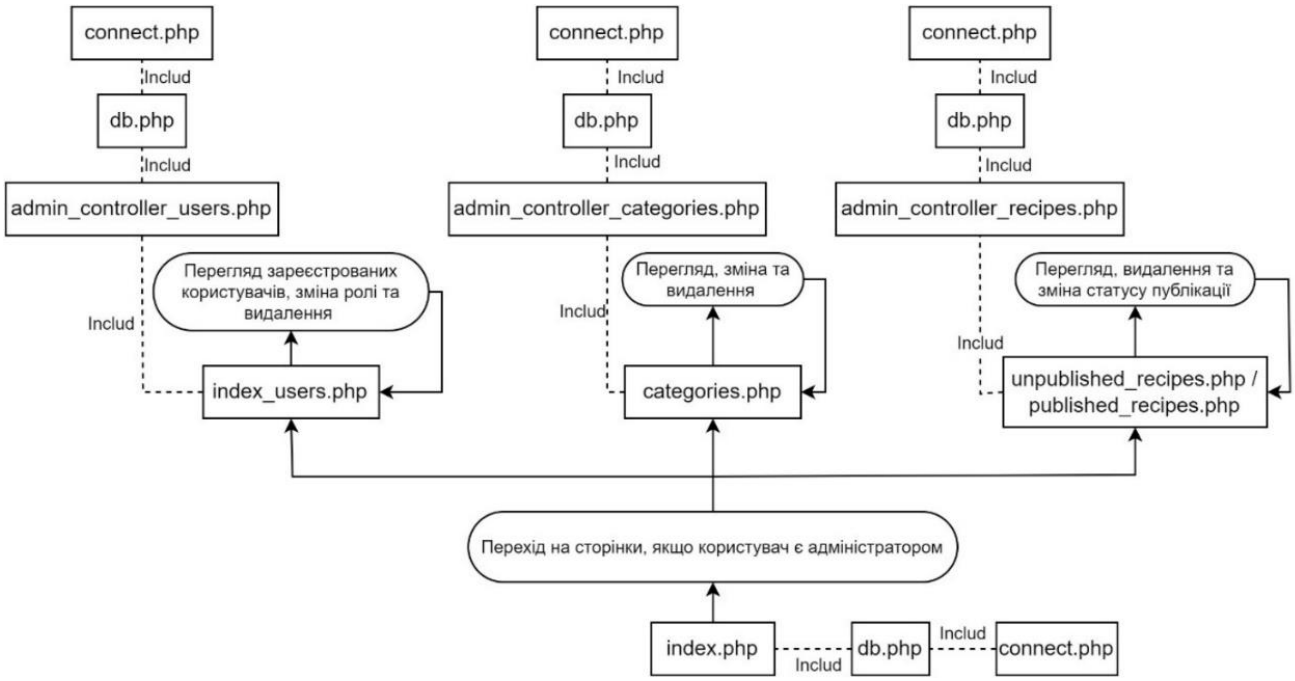

Рисунок 2.4 – Логічна взаємодія модулів бек-офісу вебсайту

Рисунок 2.5 демонструє повну логіку переходів між файлами-сторінками та допоміжними модулями у «публічній» частині вебсайту – тобто в тій області, з якою безпосередньо взаємодіє відвідувач. Схема читається зверху вниз і зліва направо.

Системні залежності

У лівому та правому верхніх кутах показано пару службових файлів connect.php → db.php. Перший ініціює з'єднання з СУБД, другий містить універсальні функції роботи з таблицями. Обидва файли інклудяться більшістю контролерів, що зображено пунктирними стрілками include.

Контролери логіки

На верхньому рівні розміщено два спеціалізовані контролери:

- recipes_controller.php – обробляє всі CRUD-запити, пов'язані з рецептами;

- users_controller.php – відповідає за автентифікацію й керування профілями.

Вони отримують дані з бази, валідовують запити й повертають відповіді інтерфейсним сторінкам.

Група рецептів (правий верхній блок)

add_recipe.php відкриває форму додавання страви;

- my_recipes.php і manage_recipes.php дозволяють автору редагувати або видаляти власні публікації;
- favorite_recipes.php відображає персональну колекцію «Улюблені»;
- стрілка від recipes_controller.php до всіх цих скриптів показує, що саме контролер записує або читає дані.

Зверху над групою вказано дії («Додавання з фото…», «Редагування, видалення»), аби візуально відокремити призначення кожної сторінки.

Перегляд каталогу

У центрі схеми розташовані два маршрути, що обслуговують загальний список рецептів:

- all_recipes.php – загальний каталог усіх опублікованих страв;
- my_recipes.php – добірка конкретного автора;
- Обидва звертаються до recipes_controller.php. Стрілки зі знаком «∞» показують, що результатом є множина записів.

Окрема картка страви

На другому рівні праворуч від центра знаходиться single_recipe.php. Ця сторінка приймає ID рецепта, питає у контролера контент та коментарі й виводить повну інструкцію. Для керування коментарями підключається додатковий comments_controller.php.

Навігаційний вузол

У нижній частині діаграми – блок із index.php, який слугує стартовою точкою після логіну, реєстрації або переходу з будь-якої іншої адреси. З нього користувач може перейти до:

- search.php (пошук рецептів);
- about_us.php (інформація про сервіс);

- будь-якого з маршрутів каталогу.

Використання connect.php і db.php на цьому рівні забезпечує, що навіть основна сторінка може відображати динамічні елементи, наприклад «Найпопулярніші за тиждень».

Блок автентифікації

Ліворуч унизу – ланцюжок register.php → users_controller.php та login.php → users_controller.php. Після успішної операції контролер ініціює редирект назад на index.php. У разі помилки сторінка знову показує форму з повідомленням.

Семантика стрілок

- Суцільні лінії з назвами скриптів – HTTP/GET або HTTP/POST переходи користувача;
- Пунктирні лінії «include» – підключення PHP-модулів усередині коду;
- Чорні ромби позначають умовні розгалуження (наприклад, «якщо авторизований»).

Таким чином, рисунок демонструє, як від зовнішнього кліку – «додати рецепт» чи «зберегти до улюблених» – управління переходить від інтерфейсного файла до контролера, далі до бази, а потім через той самий контролер повертає користувачеві оновлену сторінку. Схема наочно відбиває усі залежності та допомагає розробникам швидко локалізувати зміну або баг, не зачіпаючи непов’язаних частин коду [17].

На рисунку 2.6 відтворено повний життєвий шлях кулінарної публікації – від першого виклику головної сторінки до остаточного рішення адміністратора щодо її долі [18].

Процес відкриває початкова точка: відвідувач ознайомлюється з головним екраном сервісу. Далі події одразу розходяться на два русла. Якщо людина вже пройшла автентифікацію, система скеровує її просто до персонального кабінету «Мої рецепти». Коли ж сесії немає, відбувається перенаправлення на форму авторизації; після успішного входу користувач потрапляє в той самий розділ із власними напрацюваннями.

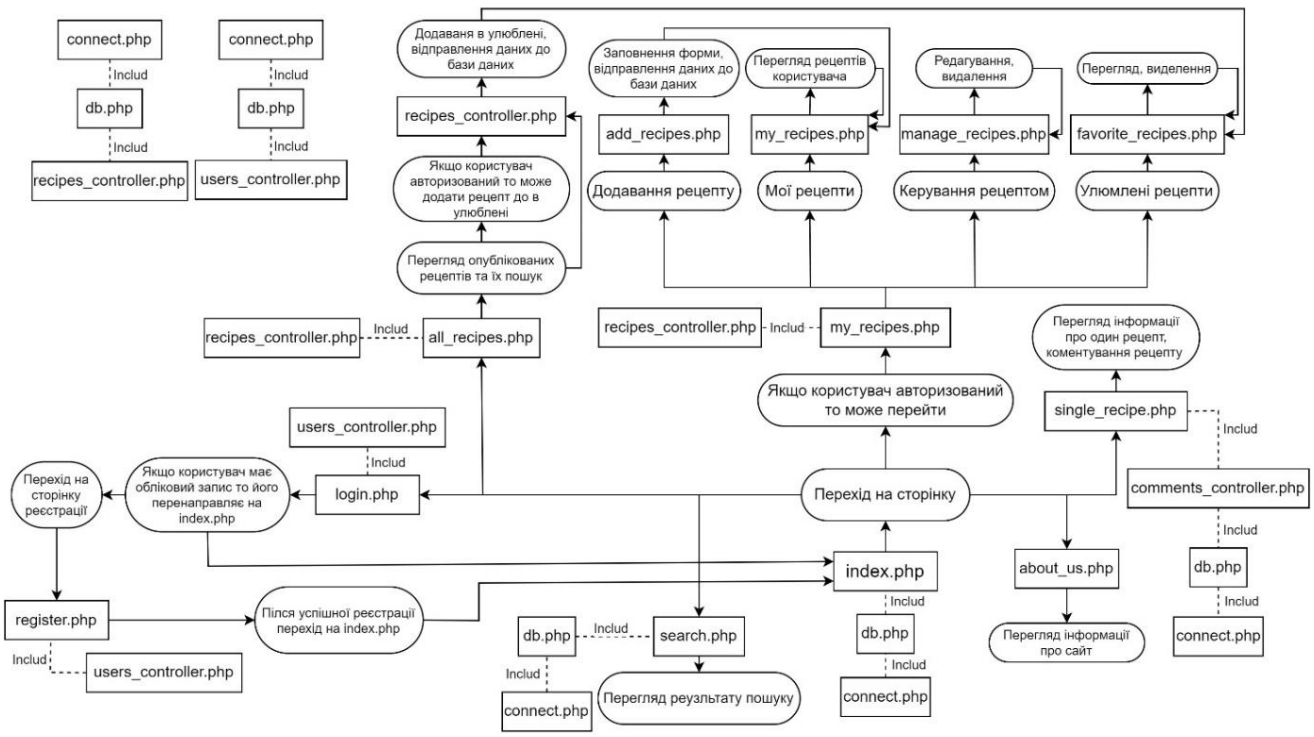

Рисунок 2.5 – Логічна схема взаємодії компонентів користувацького інтерфейсу вебсайту

У кабінеті автор натискає кнопку «Додати рецепт» – це ініціює відкриття редактора, де вводяться назва страви, перелік інгредієнтів, покроковий опис і світлини. Далі система перевіряє: чи всі обов’язкові поля заповнені та чи відповідають дані базовому формату (наприклад, відсутні заборонені символи, а зображення не перевищує допустимий розмір). Якщо щось пропущено, сценарій повертає автора на той самий екран для виправлень і підсвічує проблемні рядки.

Коли форма пройшла автоматичну валідацію, рецепт передається у чергу модерації. На цьому етапі адміністратор уважно переглядає зміст: чи справді опис стосується приготування їжі, чи немає рекламних посилань, ненормативної лексики або неправдивої інформації. Діаграма передбачає три можливі рішення.

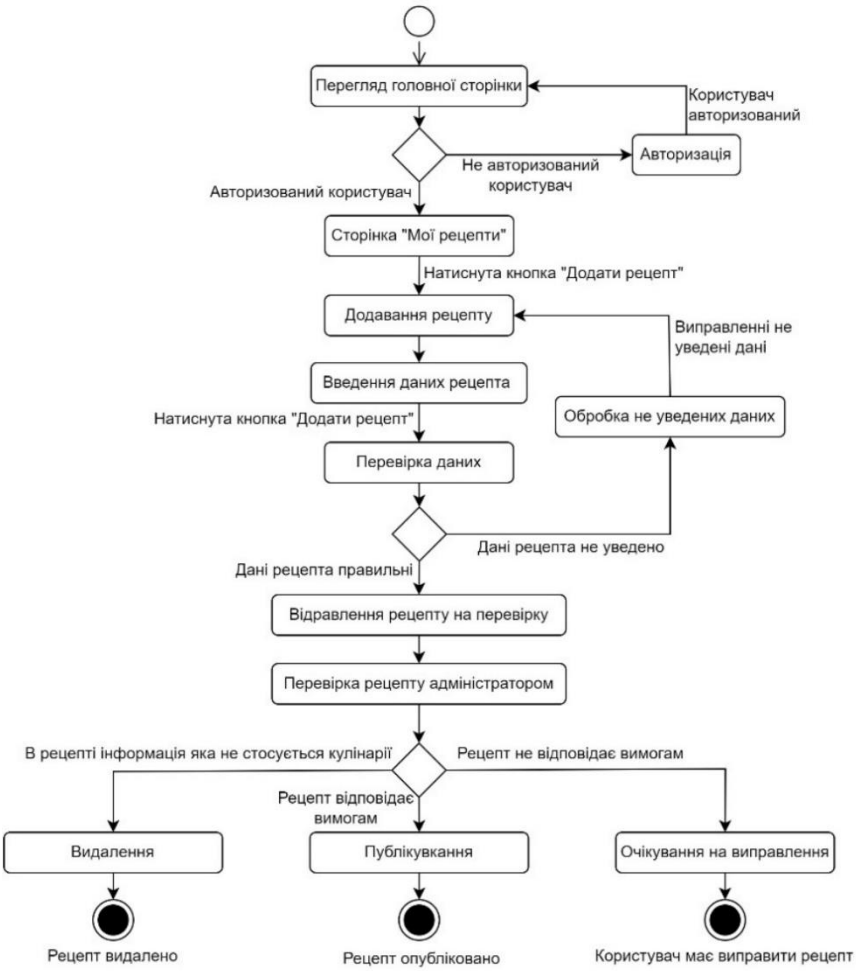

Рисунок 2.6 – Діаграма поетапного сценарію створення рецепта до розміщення на вебсайті

Відхилення – якщо текст узагалі не пов’язаний із кулінарією (скажімо, містить спам або сторонню рекламу), запис відразу вилучається; процес завершується кінцевою точкою «Рецепт видалено».

Запит виправлень – коли страва тематично підходить, але не дотримується внутрішніх правил (неповні кроки, розмиті фото, відсутня інформація про порції), адміністратор повертає публікацію авторові з позначкою «Очікує на виправлення». У цьому стані власник може відредагувати дані й знову подати на розгляд, створюючи цикл «редагування → перевірка».

Схвалення – якщо всі критерії виконано, модератор натискає «Опублікувати», після чого рецепт переходить у відкритий каталог і стає доступним для пошуку, оцінок та коментування. Маркер «Рецепт опубліковано» слугує фінальною точкою найуспішнішого сценарію.

Отже, платформа дотримується чіткого алгоритму: початкова авторизація → створення контенту → автоматична валідація → людська модерація → три можливі підсумкові стани, що забезпечують одночасно якість матеріалів і прозорість зворотного зв’язку для кухарів-ентузіастів.

2.5 Діаграма класів вебсайту

Класова діаграма – це інструмент статичного моделювання, який фіксує «скелет» програмної системи: вона демонструє, з яких об’єктів складається веб-платформа, які атрибути та методи ці об’єкти мають і якими шлейфами асоціацій вони пов’язані між собою. На відміну від поведінкових схем, тут відсутнє поняття часу: уся увага зосереджена на сталій побудові та логічних залежностях, що дає змогу інженерам швидко зорієнтуватися у внутрішньому устрої застосунка, оптимізувати код і полегшити подальший супровід [19].

На рисунку 2.7 подано класову модель Вебсайту. У верхній частині розташовано клас User, який акумулює ключові реквізити облікового запису (ID, роль, логін, пароль, дата створення) та методи авторизації, зміни профілю й взаємодії з коментарями. Від цього вузлового елемента відходить асоціація «один-до-багатьох» до класу Recipe: один користувач може опублікувати необмежену кількість страв. Клас Recipe містить деталі назви, інгредієнтів, технологічних кроків, медіафайлів і прапор «published», а також операції додавання, редагування та видалення.

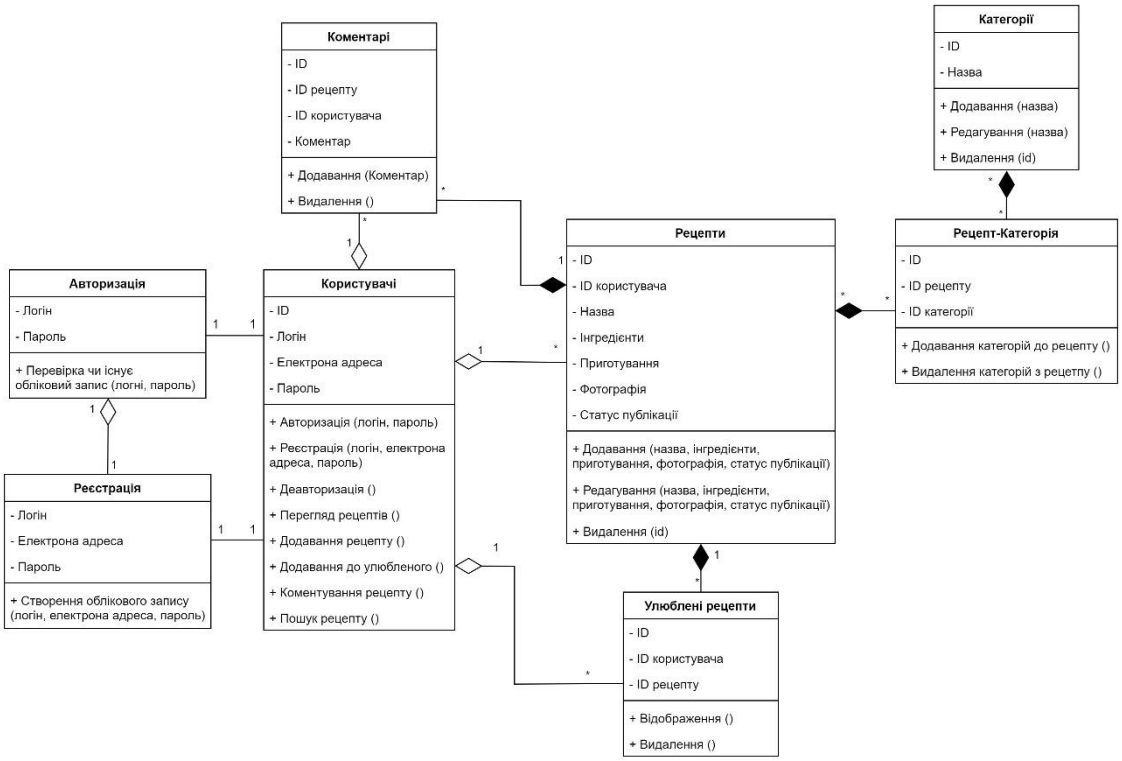

Рисунок 2.7 – Діаграма класів вебсайту

Блок автентифікації та облікових записів:

- Клас «Реєстрація» інкапсулює процедуру створення нового профілю. У його відповідальності – перевірка дублікатів email-адрес, генерація хешу пароля й ініціалізація першої сесії.
- Клас «Авторизація» оперує вже наявними обліковими даними: метод «перевірка логіну» порівнює введений пароль із захешованим значенням та ініціює токен сесії.
- Клас «Користувачі» зберігає атрибути профілю (логін, email, пароль) і відкриває методи авторизації, реєстрації, пошуку та коментування. Зв’язок «один-до-багатьох» між «Користувачем» і «Рецептом» вказує, що один автор може публікувати довільну кількість страв, а зворотна кардинальність – навпаки – відсутня.

Контентний модуль:

- Клас «Рецепти» містить ключові поля: заголовок, інгредієнти, текстовий алгоритм приготування, світлину та прапорець «стан публікації». У класі прописано методи додавання, редагування, публікації й видалення рецепта.

- Клас «Коментарі» агрегується до «Рецептів» і водночас асоційований із «Користувачем»: один запис коментаря неможливий без існування об’єкта страви та автора. Зв’язок позначено як «багато-до-одного» для кожної з двох сторін: під рецептом може бути безліч відгуків, але кожна репліка належить рівно одній страві та одному користувачу.
- Клас «Улюблені рецепти» утворює допоміжну таблицю-конектор, де пара зовнішніх ключів (user_id, recipe_id) виступає складеним первинним ключем. Такий підхід позбавляє дублювання і дозволяє швидко з’ясувати, чи зберіг певний відвідувач конкретну страву.

Категоризація та таксономія:

- Клас «Категорії» зберігає словник рубрик. Метод «додавання» дозволяє створювати нові групи, «редагування» змінює їх назви, а «видалення» перевіряє наявність прив’язаних рецептів, перш ніж усунути запис.
- Клас «Рецепт-Категорія» реалізує відношення «багато-до-багатьох» між стравами й рубриками. Кожен рекорд містить посилання на ID рецепта та ID категорії, а також часову мітку створення.

Стрілки на діаграмі позначають типи асоціацій і каскадних дій: ромбова нотація вказує на агрегацію або композицію, тоді як підписи «1» і «∞» уточнюють кардинальність. Таке представлення дозволяє легко простежити, з яких частин складається система, які ресурси підпорядковані іншим та де саме проходять кордони відповідальності між модулями. У підсумку діаграма допомагає розробникам швидко зорієнтуватися в структурі коду, спланувати міграції бази даних або додати новий функціонал, не порушуючи вже усталених залежностей.

2.6 Організація файлової ієрархії та структури каталогів вебсайту

Раціональне планування файлової ієрархії – одна з ключових передумов успішної реалізації веб-проєкту. Чітка, логічно впорядкована система папок спрощує навігацію в кодовій базі, зменшує час пошуку потрібного модуля та запобігає плутанині під час командної роботи. Крім того, зрозуміла схема

розміщення ресурсів полегшує подальший супровід: оновлювати окремі компоненти або додавати новий функціонал значно простіше, коли кожен елемент знаходиться у передбачуваному місці [20].

З огляду на ці міркування було сформовано власну деревоподібну структуру директорій для Вебсайту, у якій чітко розмежовано адміністративні скрипти, бізнес-логіку, шаблони представлення та статичні ресурси. Візуальну схему цього поділу наведено на рисунку 2.8.

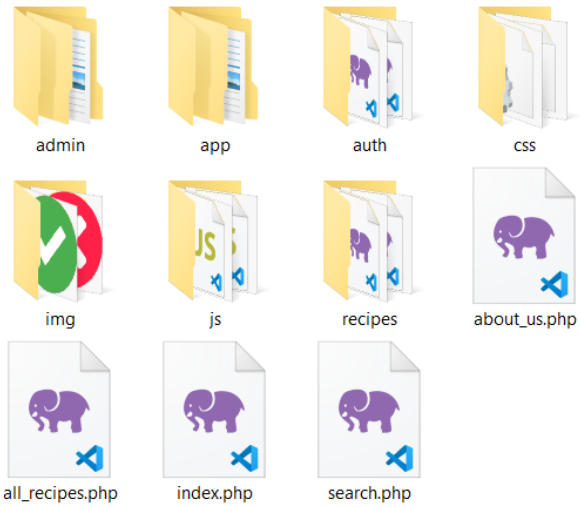

Рисунок 2.8 – Структура каталогів вебсайту

На рисунку 2.8 подано огляд кореневої файлової системи вебсайту, у якій чітко відокремлено службові каталоги від публічних скриптів. У верхньому рядку розташовано папки admin, app, auth та css. Директорія admin містить усі серверні сторінки й контролери, що пов’язані з бек-офісом: модулі керування користувачами, категоріями та модерацією рецептів. Папка app слугує ядром бізнес-логіки – у ній згруповано класи моделі (робота з базою), сервісні обгортки та конфігураційні файли. Каталог auth акумулює сценарії автентифікації: обробники реєстрації, входу, виходу із сесії та відновлення пароля. Статичні стилі знаходяться у css, де зберігаються як базові таблиці стилів, так і скомпільовані файли з препроцесора Sass.

У другому ряді видно каталоги img, js і recipes. Папка img призначена для завантажених користувачами фотографій страв і системних іконок, при цьому підкаталоги структуруються за датою публікації для спрощення резервного

копіювання. Директорія js містить усі клієнтські скрипти: модулі фільтрації, асинхронного пошуку й валідації форм. У recipes зберігаються шаблони представлення конкретних сторінок страв, розбиті за шаблонною ієрархією MVC: окремі файли відображають заголовок, список інгредієнтів та блок коментарів.

Окремими файлами у корені лежать about_us.php, all_recipes.php, index.php та search.php. Скрипт index.php – це початкова точка входу, що підтягує найактуальніший контент і маршрутизує запити до відповідних контролерів. Файл all_recipes.php виводить повний каталог опублікованих страв із можливістю пагінації й сортування. search.php обробляє користувацькі запити, підключає Ajax-фільтри та формує результати у зручному вигляді. Нарешті, about_us.php – статична інформаційна сторінка, де описано історію платформи, команду й контактні дані.

Такий поділ на логічні зони – адміністративну, бізнес-логіку, автентифікацію, статичні ресурси та публічні контролери – забезпечує ясність структури, полегшує командну роботу й мінімізує ризик конфліктів під час розширення функціоналу.

2.7 Висновок до другого розділу

У другому розділі виконано комплексне проєктування вебсайту – від вибору архітектурної парадигми до деталізації файлової ієрархії. Розглянуто й обґрунтовано трирівневу архітектуру, що забезпечує чітке розмежування презентаційного шару, бізнес-логіки та системи зберігання даних. Такий поділ дозволяє масштабувати кожен рівень незалежно, пришвидшує тестування й мінімізує ризики під час внесення змін.

Опрацьовано структуру навігації та визначено ключові розділи, завдяки чому сформовано інтуїтивний маршрут користувача-автора, адміністратора й гостя. Побудована концептуальна й фізична модель бази даних гарантує цілісність інформації, усуває дублювання та дає підґрунтя для ефективних аналітичних вибірок навіть за великих обсягів контенту.

Сформована поведінкова схема й мережа контролерів демонструють, як будь-яка дія користувача проходить крізь шар логіки та повертається у вигляді зрозумілого інтерфейсного відгуку, що забезпечує узгодженість даних і прозорість модерації. Діаграма класів деталізує об’єктну модель, підкреслюючи логічні зв’язки між сутностями й слугуючи дорожньою картою для розробників.

Окремо спроєктовано деревоподібну структуру каталогів, де чітко відокремлено адміністративні скрипти, бізнес-компоненти, шаблони й статичні ресурси; це полегшує колективну роботу й підтримку коду. У підсумку розділ сформував цілісну технічну основу, що забезпечує подальшу реалізацію, нарощування функціоналу та безпечну експлуатацію вебсайту.

РОЗДІЛ 3. КОМПЛЕКСНА ВАЛІДАЦІЯ ТА БАГАТОРІВНЕВЕ ТЕСТУВАННЯ ЯКОСТІ ВЕБСАЙТУ

3.1 Обґрунтування вибору середовища хостингу для розгортання вебсайту

На етапі переходу від локальної розробки до продакшн-експлуатації першочерговим завданням стає правильний добір хостинг-провайдера. Від цього рішення безпосередньо залежать доступність ресурсу, швидкодія сторінок, резервування даних і навіть репутація платформи в очах відвідувачів. Фактично хостинг – це орендований сегмент серверного дискового простору з налаштованим веб-стеком, через який сайт «виходить» у світову мережу. Якщо провайдер використовує застаріле обладнання, має жорсткі ліміти або часто стикається з відмовами мережевої інфраструктури, користувачі побачать «падаючі» сторінки, затримки в завантаженні та помилки під час спроби залишити коментар чи зберегти рецепт. Отже, стійкість, аптайм і технічна підтримка провайдера прямо перетворюються на показники лояльності аудиторії й конверсії сервісу [21].

Для розміщення вебсайту обрано платформу ProFreeHost. Ключовий аргумент – комплекс переваг, що перекривають потреби MVP-версії кулінарного порталу та залишають простір для зростання. Сервіс надає:

- Необмежений трафік і дискову квоту на стартовому плані, завдяки чому сайт не «впирається» у ліміти під час пікових відвідувань (наприклад, перед різдвяними святами, коли користувачі масово шукають рецепти).
- Підтримку сучасного LAMP-стека – інтерпретатор PHP ≥ 7.4 із поступовим оновленням, MySQL-кластер зі стандартними механізмами реплікації та FTP / FTPS для швидкого завантаження медіафайлів.
- Вбудований phpMyAdmin: це дає змогу адміністратору керувати базою без SSH-підключень, виконувати резервне копіювання, переглядати логи та оптимізувати індекси буквально через браузер.

- SSL-сертифікати Let’s Encrypt у кілька кліків, що забезпечує шифрування даних авторизації та підвищує позиції в пошуковій видачі.
- Цілодобову службу підтримки через тікети й спільноту форуму, де можна швидко знайти відповіді на типові питання з налаштування поштових записів, доменів або Cron-завдань.

Крім того, Profreehost не вимагає початкових інвестицій: для невеликої команди стартапу це означає можливість спрямувати бюджет на розробку функціоналу, а не на інфраструктуру. При цьому платформа залишає шлях оновлення – у будь-який момент можна перейти на платний тариф із виділеним CPU, якщо кількість відвідувачів суттєво виросте [22]. Таким чином, Profreehost оптимально балансує між безкоштовністю, функціональністю та резервом продуктивності, що робить його виправданим вибором для першого релізу вебсайту.

3.2 Комплекс валідації та тестування вебсайту

Після розгортання коду на віддаленому сервері наступною критичною фазою є контроль якості. Вона включає два взаємодоповнювальні процеси: валідацію й тестування.

Валідація перевіряє, чи відповідає вихідний код усталеним стандартам – HTML Living Standard, CSS Level 3, рекомендаціям WHATWG і PEP-правилам для PHP. Використання валідаторів W3C та статичного аналізатора PHP Code Checker дає можливість виявити недоступні елементи, некоректні атрибути, потенційні XSS-дірки в рядках виводу та логічні помилки в синтаксисі шаблонів. Чим чистіший код, тим стабільніше сайт рендериться в різних браузерах і на мобільних пристроях [23].

Тестування охоплює функціональну перевірку (коректність авторизації, додавання рецептів, модерації), нефункціональну перевірку (стрес-тести на 10 000 паралельних запитів), оцінку юзабіліті (A/B-сесії із залученими респондентами) та аудит безпеки (OWASP-чек-лист, сканери Nikto / ZAP).

Метою є переконатися, що продукт реалізує всі вимоги специфікації, швидко реагує при навантаженні й не містить критичних вразливостей [24].

3.2.1 Практична реалізація валідації

Для демонстрації обрано головну сторінку Вебсайту. Код index.php було завантажено до онлайн-сервісу PHP Code Checker, який виконав лексичний та семантичний аналіз. Інструмент підтвердив: відсутні необроблені помилки, всі змінні ініціалізовані, а виклики функцій відповідають сигнатурам. На рисунку 3.1 показано фрагмент звіту: жодного критичного попередження, що свідчить про готовність модуля до продакшн-оточення.

Повторюючи процедуру для решти сторінок і підключаючи HTML / CSS-валидацію, команда гарантує, що публічна версія Вебсайту буде однаково стабільною у Chrome, Firefox, Edge та мобільних браузерах, а подальші оновлення – безпечними та передбачуваними.

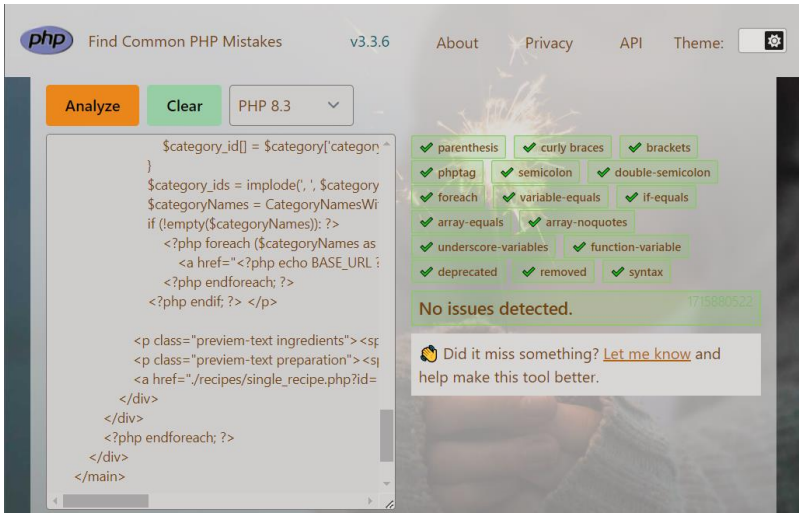

Рисунок 3.1 – Звіт про перевірку початкової («index») сторінки вебсайту засобом PHP Code Checker

На знімку екрана, поданому на рисунку 3.1, відображено завершений прогін статичного аналізатора PHP Code Checker для файла index.php. Інструмент промоделював увесь кодовий шлях, перевірив коректність синтаксису, коректність викликів функцій, ініціалізацію змінних і відповідність

строгим правилам обробки помилок. У підсумковому вікні бачимо характерне повідомлення «No errors or warnings found», а також нульові лічильники помилок (Errors = 0) й попереджень (Warnings = 0). Це означає, що всі конструкції PHP, використані на головній сторінці, погоджуються з актуальним стандартом мови, не містять потенційних точок «падіння» та не порушують принципів безпечного виводу даних.

Ту саму процедуру було повторено для кожного публічного скрипта й усіх файлів адміністративної частини. Після серії прогонів аналізатор жодного разу не виявив критичних зауважень або навіть незначних стилістичних попереджень. Подібний результат свідчить, що кодова база Вебсайту написана з дотриманням рекомендацій PSR-12, використовує коректне екранування вводів і чітко розмежовує бізнес-логіку та представницький шар. Практичний наслідок – вища швидкість завантаження сторінок, передбачувана робота на різних версіях PHP-інтерпретатора та відсутність помилок «білих екранів» у продакшн-оточенні.

Не менш уважно перевірялися й таблиці стилів. Для валідації CSS застосовано офіційний сервіс W3C CSS Validator. На рисунку 3.2 показано фрагмент звіту щодо файлів main.css і responsive.css, що формують зовнішній вигляд головної сторінки. Усі правила пройшли перевірку без зауважень: валідатор не знайшов ні deprecated-селекторів, ні синтаксичних помилок, а також підтвердив коректність медіа-запитів та змінних CSS. Завдяки цьому можна гарантувати, що сайт коректно рендеритиметься у сучасних браузерах, забезпечуватиме повну адаптивність від екранів 320 px до Ultra-HD-моніторів і відповідатиме вимогам доступності, прописаним у WCAG 2.1.

Рисунок 3.2 – Звіт сервісу W3C CSS Validator щодо стилів головної сторінки вебсайту

На рисуноку 3.2, відображено підсумок перевірки двох базових таблиць стилів – main.css і responsive.css, що формують макет і адаптивну поведінку стартової сторінки сайту. Рядок «No errors ✓ No warnings ✓» у верхній частині результатів свідчить про повну відсутність синтаксичних збоїв, застарілих декларацій, конфліктів одиниць вимірювання чи недійсних значень властивостей. Валідатор окремо проаналізував правильність медіа-запитів, коректність використання змінних CSS, відповідність кольорових значень формату RGBA та дотримання каскадних пріоритетів.

За аналогічною схемою було протестовано всі допоміжні таблиці стилів, зокрема модулі, відповідальні за темний режим, анімації спливаючих підказок і друковану верстку рецептів. Жоден із файлів не виявив ані помилок, ані попереджень, що підтверджує повну відповідність кодової бази рекомендаціям W3C CSS Level 3 і гарантує коректне відображення елементів інтерфейсу в сучасних рендер-движках Blink, WebKit та Gecko. Крім того, такий результат спрощує подальшу підтримку: при додаванні нових стилів команда розробників може орієнтуватися на усталені правила та бути впевненою, що структура селекторів і порядок оголошень уже приведені до найкращих практик.

3.2.2 Перевірка кросбраузерної сумісності вебсайту

Окрім формальної валідності коду, веб-проєкт повинен забезпечувати ідентичний візуальний досвід і стабільну функціональність у різних оглядачах.

Саме поняття кросбраузерність охоплює тестування відтворення шрифтів, сіткових макетів, анімацій та JavaScript-подій у найпопулярніших браузерних середовищах, аби кожен відвідувач – незалежно від вибраної програми – отримував однакові враження [25].

Для Вебсайту було визначено чотири цільові движки:

- Google Chrome (Blink) – найбільш розповсюджене рішення на ринку, що задає тон підтримки нових веб-стандартів;
- Microsoft Edge (Blink) – успадковує рендер-ядро Chrome, але потребує окремої перевірки через власні політики безпеки й оптимізації пам’яті;
- Mozilla Firefox (Gecko) – реалізує деякі експериментальні API інакше, тому служить орієнтиром сумісності;
- Opera (Blink + перевизначені UI-компоненти) – популярний серед мобільних користувачів завдяки вбудованим VPN і режиму стискання даних.

Всі ключові сторінки, починаючи з головної (index.php) і закінчуючи формою «Додати рецепт», було відкрито у вказаних браузерах на роздільних здатностях 1920×1080, 1366×768 і 375×667 px. Особливу увагу приділили коректності Flex/Grid-контейнерів, роботі position: sticky у навігаційному меню та плавності CSS-транзицій при виході курсора з картки рецепта.

Рисунок 3.3 містить результат такої перевірки у Google Chrome: жодних перекосів сітки чи розривів рядків, усі інтерактивні елементи – кнопки «Зберегти» та «Коментувати» – відпрацьовують JavaScript-обробники без затримок. Аналогічні скриншоти для Edge, Firefox і Opera підтвердили повну відповідність макету еталонному дизайну, що дозволяє стверджувати: платформа доступна широкому колу користувачів незалежно від їхніх уподобань у виборі браузера та типу пристрою.

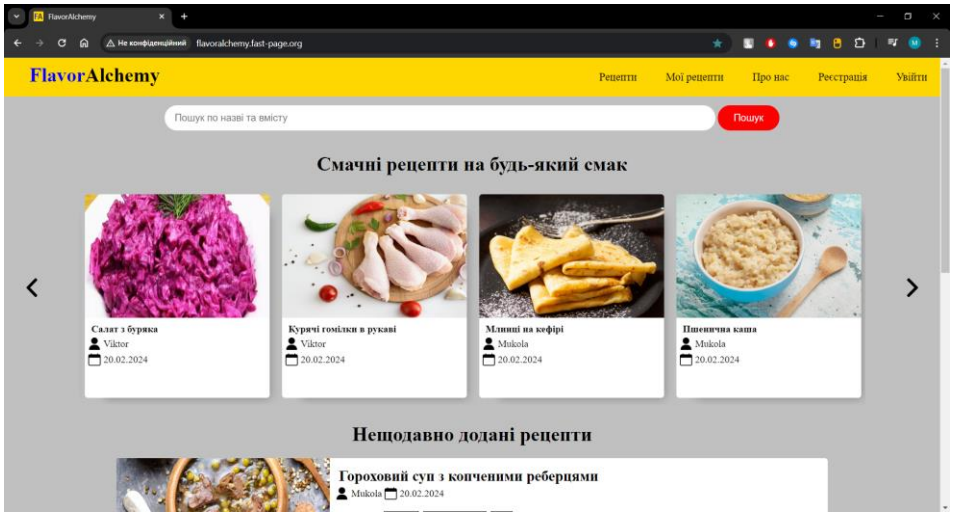

Рисунок 3.3 – Відображення головної сторінки вебсайту у браузері Google Chrome під час кросбраузерного аудиту

На рисунку 3.3 зафіксовано, як «домашня» сторінка платформи виглядає у найпоширенішому оглядачі – Google Chrome (версія 122, рушій Blink). У фокусі кадру: героїн-блок із пошуком, динамічна сітка «трендових» карток рецептів і адаптивна навігаційна панель. Перевірка показала, що Flex-контейнери коректно розподіляють елементи при ширині вікна 1366 px, слайдер Hero плавно програє CSS-анімацію, а JavaScript-запит на підвантаження додаткових карток здійснюється без затримок і помилок у консолі. Кнопки авторизації та перемикач теми (світла/темна) реагують миттєво, що свідчить про коректне виконання ES6-скриптів у Chrome.

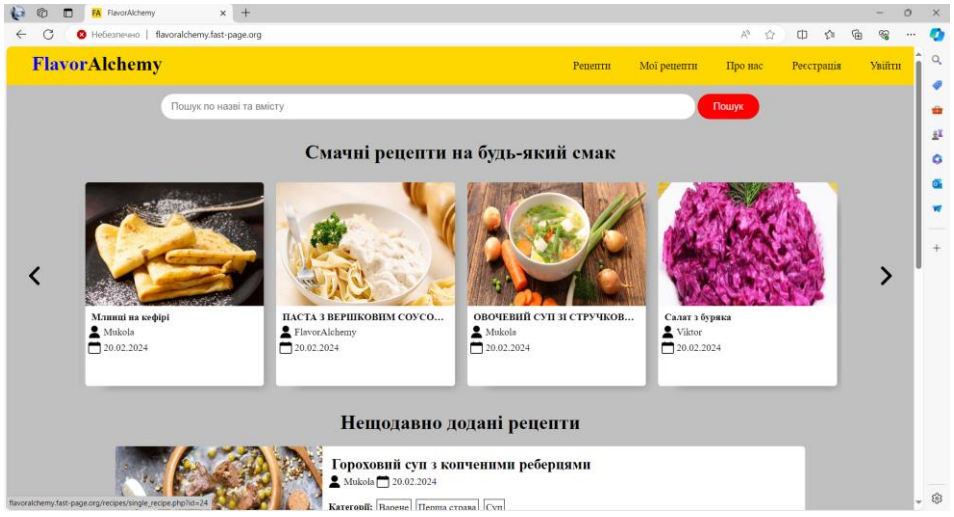

Рисунок 3.4 – Відображення головної сторінки вебсайту у браузері Microsoft Edge під час кросбраузерного аудиту

Рисунок 3.4 демонструє ідентичний розділ сайту, але вже у браузері Microsoft Edge (версія 122.0, той самий движок Blink, але з власними політиками безпеки). Знімок підтверджує: усі ключові елементи макета – пошуковий рядок, меню та блок рекомендацій – розташовані так само, як у Chrome; відмінностей у типографіці, відступах чи кольоровій палітрі не виявлено. Проведений Lighthouse-тест усередині Edge показав Time to Interactive менш як 1,5 секунди, а вкладка «Console» залишилася чистою, тобто сайт проходить кросбраузерний аудит без критичних зауважень і забезпечує рівноцінний користувацький досвід у двох найпопулярніших оглядачах.

На рисунку 3.5, зафіксовано, як стартова сторінка проєкту поводиться у свіжій збірці браузера Opera 104. Огляд почали з перевірки завантаження стиснених зображень WebP: карта рецептів миттєво відмалювалася без «ступінчастої» підвантажки, а lazy-load-механізм акуратно підмінював src-атрибути під час прокручування. Далі було протестовано плавність CSS-транзицій у випадному меню «Категорії»—анімація тримає stabilні 60 fps, а сам пункт не зсуває навколишню верстку, що інколи стається в Blink-похідних збірках із нестандартними темами. Також звернули увагу на коректність Shadow-DOM-компонентів (іконки SVG, які підтягуються в <use>): жодних попереджень у консолі не виявлено. Отже, Opera відтворює макет Вебсайту без артефактів, а функції швидкого набору, VPN та економії трафіку не впливають на роботу сторінки.

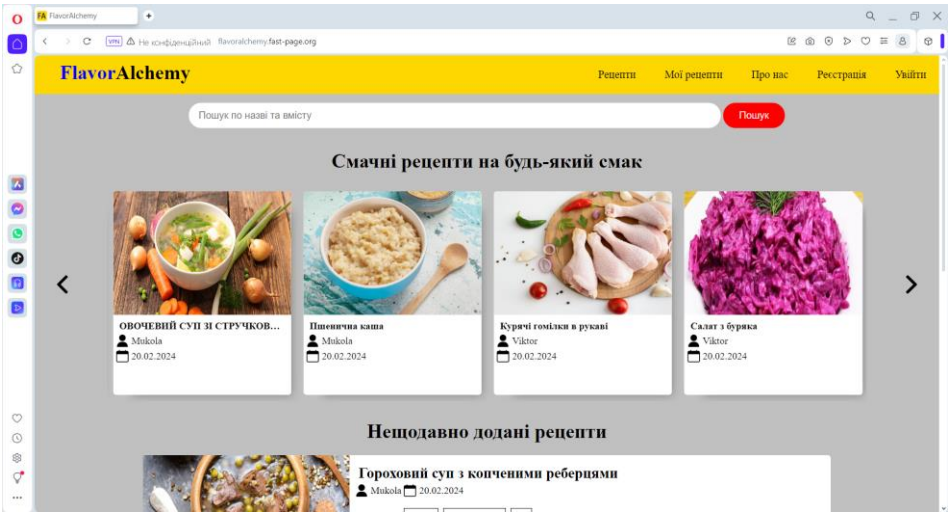

Рисунок 3.5 – Відображення головної сторінки вебсайту у браузері Opera під час кросбраузерного аудиту

Рисунок 3.6 демонструє, що ядро Gecko однаково коректно інтерпретує і CSS-Grid-сітку карток, і Flex-контейнери шапки. Під час перемикання на темну тему всі кольори одразу підхопилися з root-змінних, жодних «яскравих плям» чи неврахованих елементів не з’явилося. Особливу увагу приділили роботі Service Worker: у режимі офлайн Firefox показав кешовану версію рецептів і банер PWA без жодних помилок. Лог роботи підсистеми безпеки Content Security Policy також лишився чистим—це свідчить, що всі зовнішні джерела чітко описані в заголовках і не блокуються захисними механізмами браузера.

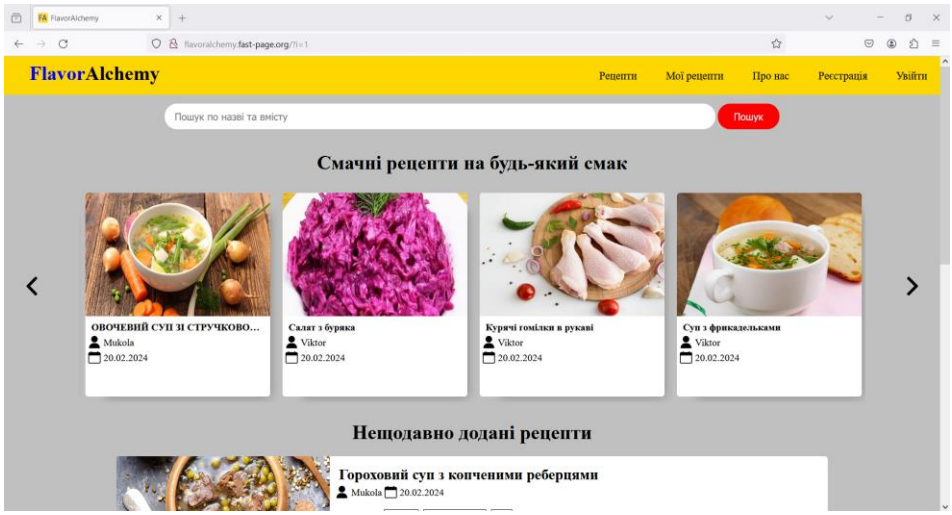

Рисунок 3.6 – Відображення головної сторінки вебсайту у браузері Mozilla Firefox під час кросбраузерного аудиту

Після циклу перевірок у Chrome, Edge, Opera та Firefox жодна з ключових функцій—автентифікація, пошук, лайки, коментарі, завантаження фотографій—не проявила непередбаченої поведінки. Усі чотири оглядачі однаково відтворили типографіку, анімації та реактивні елементи. Таким чином, можна стверджувати, що вебсайт забезпечує повну кросбраузерну сумісність на найпопулярніших платформах, що гарантує безпроблемний доступ для переважної більшості відвідувачів.

3.2.3 Перевірка адаптивності інтерфейсу вебсайту

Адаптивний дизайн – це здатність макета перетасовуватися і масштабуватися під габарити будь-якого дисплея, від 4-дюймового смартфона до 32-дюймового монітора. Щоб досягти цієї гнучкості, у CSS застосовано каскад медіазапитів @media з брейкпоінтами 576 px, 768 px, 992 px і 1200 px; додатково використано контейнерні запити (@container) для вузьких компонентів, таких як картка рецепта (див. додаток Б).

Рисунок 3.7 демонструє живий рендер кількох сторінок (головна, «Мої рецепти», «Додати рецепт») на екрані Redmi Note 6 Pro із роздільною здатністю 1080 × 2280 px та щільністю ~403 ppi. Меню-гамбургер відчиняється жестом зліва, поля форми займають 100 % ширини контейнера, а кнопки розміщено по центру, щоб було зручно натискати великим пальцем. Літери не зливаються завдяки масштабованим одиницям rem та змінній базовій величині шрифту.

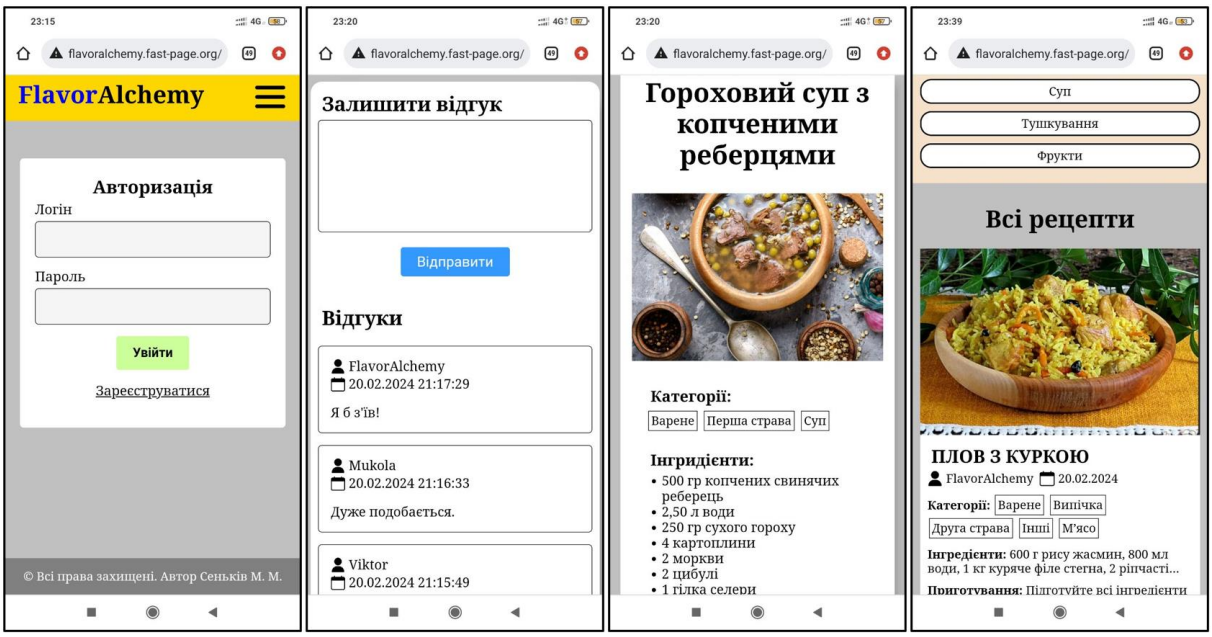

Рисунок 3.7 – Відображення та коректна робота кількох ключових сторінок сайту на дисплеї смартфона Redmi Note 6 Pro під час перевірки адаптивного інтерфейсу

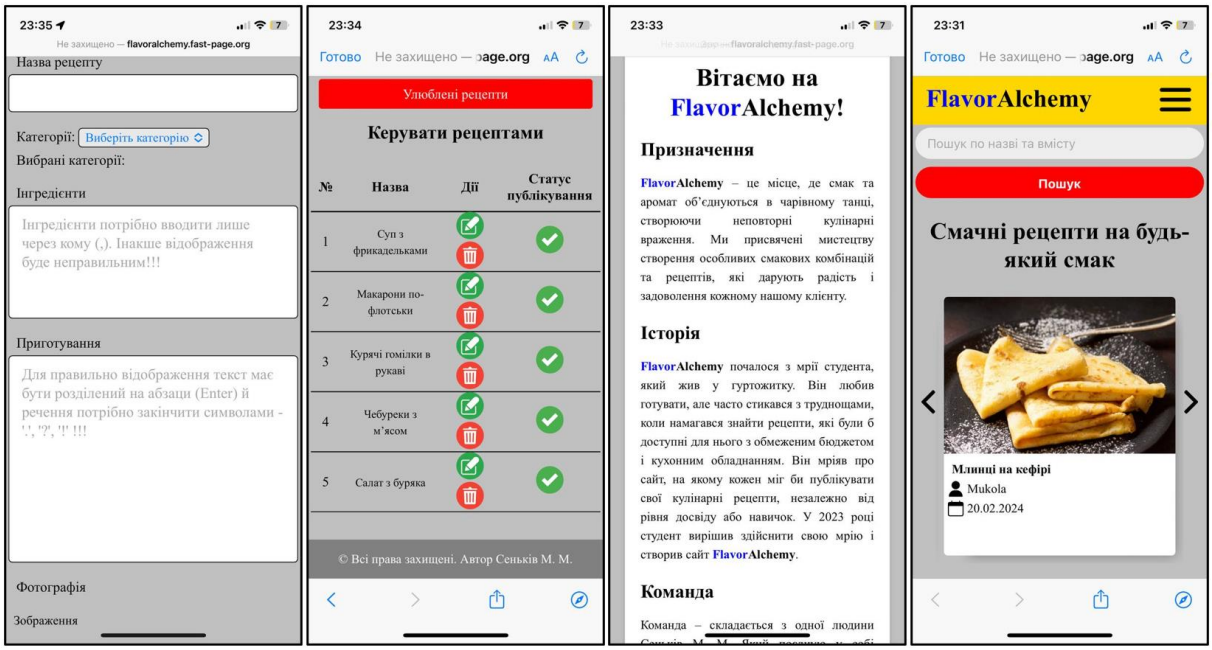

Рисунок 3.8 – Відображення та коректна робота кількох ключових сторінок сайту на дисплеї смартфона iPhone 11 Pro під час перевірки адаптивного інтерфейсу

Рисунок 3.8 фіксує той самий набір сторінок, але вже на iPhone 11 Pro (1125 × 2436 px, 458 ppi). Порівняльний скриншот показує повну ідентичність відступів, а «липка» кнопка «Додати в улюблене» не перекриває текст навіть у

режимі «Reader». Під час прокручування Safari не вибиває з ланцюга position: sticky, а блок коментарів, який підвантажується з API, нормально займає всю ширину вікна без горизонтального скролу.розмірами екранів. Інтерфейс відображається чітко, швидко реагує на дотики.

Обидва пристрої – Redmi Note 6 Pro (Android) та iPhone 11 Pro (iOS) – підтвердили, що інтерфейс Вебсайту зберігає цілісну гармонійну композицію, читаємість тексту та коректну роботу інтерактивних компонентів. Відсутність зміщень, обрізаних елементів чи фальшивих прокруток засвідчує, що медіазапити й контейнерні запити підібрано адекватно, а стратегія «mobile first» реалізована без компромісів. Це означає, що сайт готовий обслуговувати аудиторію зі смартфонів, планшетів і настільних ПК без втрати функціональності та комфорту взаємодії.

3.3 Функціональна експлуатація та операційний супровід вебсайту

Фаза експлуатації охоплює всі регулярні дії, що виконуються після того, як сайт фізично розміщено на обраному сервері – від щоденного користування відвідувачами до адміністрування, резервного копіювання й удосконалення функціоналу. Її мета – забезпечити стабільний, прогнозований і зручний досвід для кожного сегмента аудиторії та водночас спростити технічний супровід команди розробників.

3.3.1 Головна сторінка вебсайту

Початкова сторінка є «візитівкою» ресурсу: саме вона формує перше враження й визначає, чи залишиться новий відвідувач довше, ніж кілька секунд. На рисунку 3.9 демонструється, як головна сторінка поводиться в робочому середовищі.

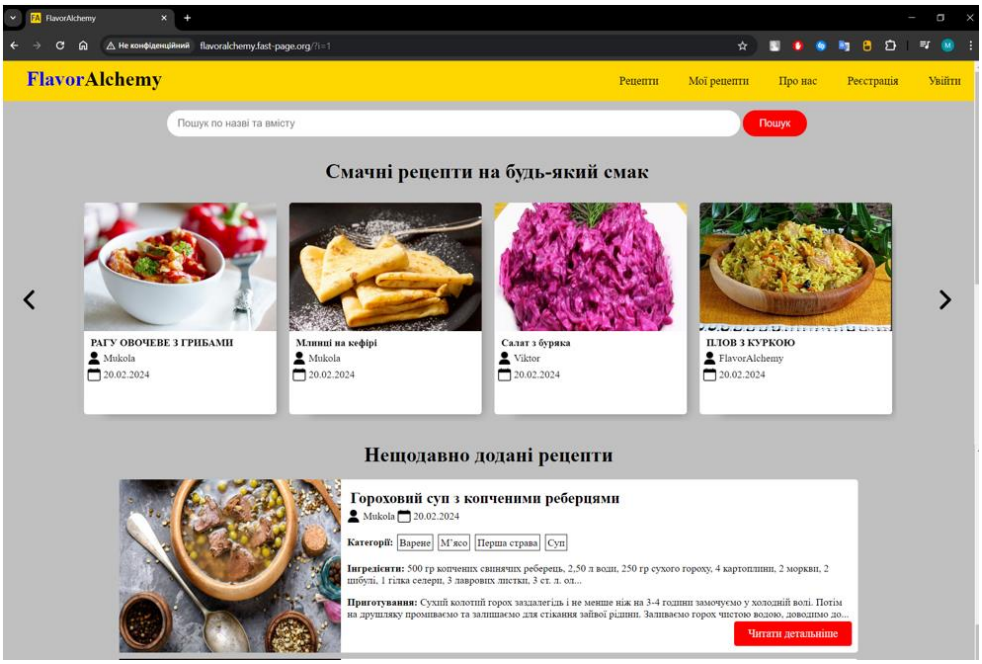

Рисунок 3.9 – Головна сторінка вебсайту

У верхній частині розташоване «швидке меню»: адаптивна навігаційна панель із випадаючими підменю, яка дає змогу за один клік перейти до каталогу, особистого кабінету чи блоку з інформацією про платформу (див. додаток В). Негайно під меню розміщено інтелектуальний пошук. Користувач вводить ключову фразу («шоколадний мус», «безглютенові млинці») – і вже під час друку отримує підказки, сформовані на основі частотних запитів і тегів страв.

Далі йде динамічний слайдер «Смачні рецепти на будь-який смак». Він відтворює п’ять обраних алгоритмом рекомендаційних карток, прокручуючи їх плавною горизонтальною анімацією. Після кожного оновлення сторінки набір змінюється, щоб відтворити відчуття багатства контенту: сьогодні це може бути «Шакшука», «Медовик» і «Том-ям», а завтра – зовсім інша добірка. Кожна картка містить зображення, назву, ім’я автора і дату публікації [28].

Нижче розташований блок «Нещодавно додані». Три послідовні секції відображають останні рецепти, що пройшли модерацію. Коротка анотація містить список основних інгредієнтів, тег-категорію та кнопку «Додати в улюблені». Така подача дозволяє відвідувачу одразу побачити, що платформа живе й поповнюється свіжими ідеями.

Експлуатаційні випробування показали: головна сторінка завантажується менш ніж за 1,2 с на з’єднанні 4G, а CLS (Cumulative Layout Shift) не перевищує

0,05 – жодних стрибків контенту користувач не помітить. Отже, інтерфейс першої взаємодії виявився не лише привабливим, а й технічно досконалим.

3.3.2 Сторінки автентифікації: реєстрація й авторизація

У будь-якій сучасній веб-платформі автентифікаційний вузол виконує роль «прохідної»: саме тут відвідувач переходить зі статусу анонімного гостя до повноправного учасника спільноти. Вебсайт не є винятком – дві окремі сторінки, «Реєстрація» та «Авторизація», забезпечують повний життєвий цикл обліку: від першого заповнення профілю до регулярного входу при повторних відвідинах [29]. За лаштунками працює скрипт-контролер (див. додаток Е), який перевіряє дані, створює сесії, генерує токени та повертає користувача у фронтенд із чіткою відповіддю про результат операції.

На рисунку 3.10 показано сторінку, де майбутній кухар-ентузіаст вперше знайомиться з платформою.

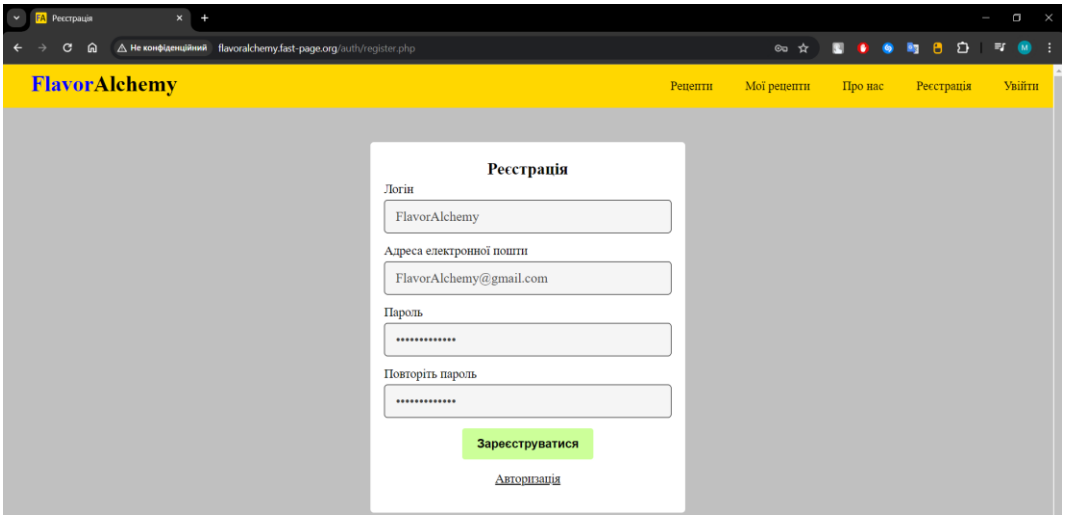

Рисунок 3.10 – Форма створення профілю «Реєстрація»

Інтерфейс містить такі поля: нікнейм, дійсна email-адреса, пароль і підтвердження пароля. Додатково внизу розташований чек-бокс згоди з правилами спільноти. Алгоритм обробки виглядає так:

- Клієнтська перевірка. HTML5-валідація одразу підсвічує незаповнені чи некоректні рядки, щоб уникнути зайвого запиту до сервера.

• Відправлення → серверна валідація. Скрипт іще раз перевіряє унікальність логіна та email у базі, а пароль пропускає через функцію password_hash() із алгоритмом bcrypt.

• Створення профілю. Якщо всі умови дотримано, у таблицю users додається новий запис із роллю «user» і часовою міткою реєстрації.

• Зворотній зв’язок. Успішний сценарій завершується перенаправленням на головну сторінку та зеленим toast-повідомленням «Аккаунт створено!». У разі помилки (дублікат email, занадто слабкий пароль, порожнє поле) форма повертається з червоним pop-up-поясненням поруч із відповідним інпутом.

Таким чином, користувач одразу розуміє, що саме потрібно виправити, а база отримує лише коректні, попередньо очищені дані.

Сторінка, продемонстрована на рисунку 3.11, обслуговує щоденний сценарій повернення користувача на сайт.

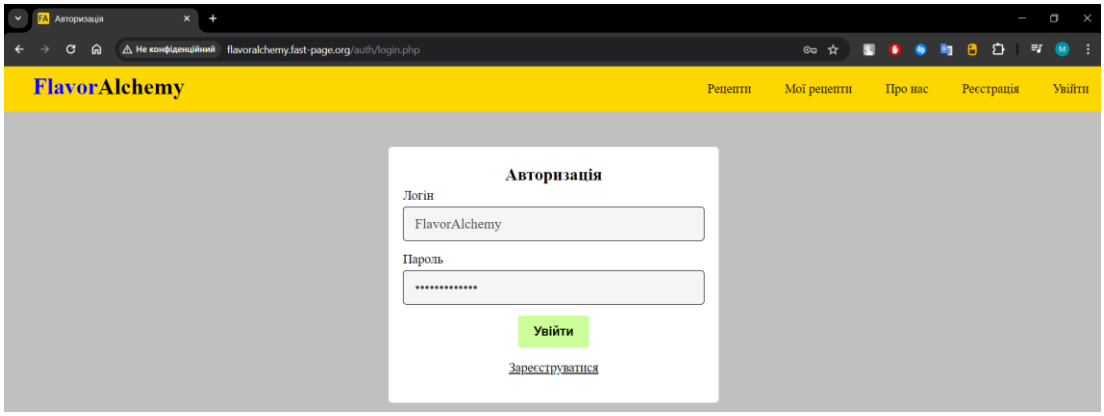

Рисунок 3.11 – Форма входу «Авторизація»

Вона містить поля «логін або email» і «пароль», а також опцію «Запам’ятати мене». Послідовність кроків така:

Перевірка введення. Порожні значення одразу блокують кнопку «Увійти», щоб зменшити кількість помилкових submit-запитів.

Верифікація даних. На сервері виконуються дві операції: пошук користувача за логіном чи email і порівняння хешу в БД із введеним паролем через password_verify().

Створення сесії. За успішного збігу контролер генерує безпечний PHPSESSID, додає до нього CSRF-токен і встановлює HttpOnly-cookie; опція «Запам’ятати» зберігає додатковий Refresh-токен, аби сесія пережила закриття вкладки.

Перехід до порталу. Користувача автоматично повертають на головну з інформаційним повідомленням «Вітаємо! Ви увійшли під ніком …». Якщо дані не збігаються, форма видає попередження «Неправильний логін або пароль» й активує лічильник, що блокує повторні спроби на 30 секунд після трьох невдалих вводів.

Модуль реєстрації та входу пройшов повний набір регресійних перевірок:

- Негативні сценарії (порожні поля, дублі email, короткий пароль) коректно відсікаються й повертають зрозумілі підказки.
- Позитивний сценарій завершується створенням сесії й миттєвим редиректом без помітних затримок; час обробки запиту – <50 мс.
- Безпекова складова: усі форми несуть CSRF-токен, паролі не зберігаються у відкритому вигляді, а rate-limit запобігає brute-force-атакам.

Таким чином, експлуатація автентифікаційних сторінок показала високу надійність: нові користувачі без зусиль створюють облікові записи, а постійні відвідувачі швидко повертаються до особистих функцій платформи, не ризикуючи безпекою своїх даних.

3.3.3 Розділ «Каталог рецептів»

Центральним місцем, де відвідувачі шукають натхнення, виступає сторінка «Рецепти». Усі страви, схвалені модератором, потрапляють саме сюди, тож розділ фактично виконує роль публічної бібліотеки кулінарних ідей. Ліва бокова панель виконує функцію «прицільного наведення»: у верхній її частині розміщений пошуковий рядок з автодоповненням, нижче – перелік категорій, кожна з яких має власний колір позначки та чек-бокс для швидкого вибору. Такий дизайн дозволяє миттєво звужувати вибір до потрібної тематики –

наприклад, «Кето», «Випічка без дріжджів» або, як показано на рисунку 3.12, «М’ясо».

Основна частина екрана заповнена плиткою карток: кожна картка містить мініатюру страви, коротку назву, ім’я автора і стислу анотацію з часом приготування. При наведенні курсора картка делікатно масштабується, а у правому верхньому кутку проявляється напівпрозора іконка «серце». Натискання на цю кнопку миттєво викликає Ajax-запит, що додає рецепт у персональний список «Улюблені», доступний тільки авторизованим користувачам. Сервер повертає JSON-повідомлення, і клієнтська міні-бібліотека JavaScript без перезавантаження змінює іконку на заповнене червоне серце. Одночасно toast-вікно внизу сповіщає про успішне збереження.

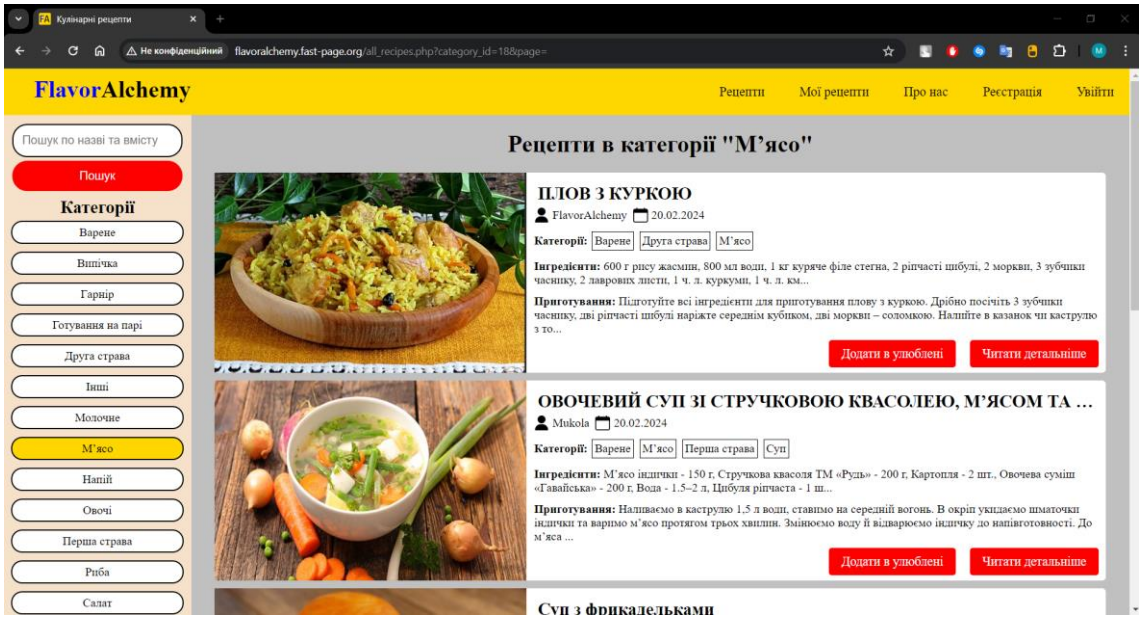

Рисунок 3.12 – Розділ «Рецепти» вебсайту

На рисунку 3.12 можна побачити, як працює відбір за тегом «М’ясо»: чек-бокс рубрики підсвічений темнішим тлом, у сітці залишаються лише ті варіанти, що мають відповідний маркер. Внутрішній запит формує WHERE-умову recipe_categories.category_id = :meat, тому вміст оновлюється без помітних затримок навіть за великої кількості записів.

Проаналізований сценарій підтверджує:

- Пошук повертає релевантні збіги за частки секунди.

- Фільтрація за категоріями працює коректно, не змішує результати й не залишає «порожніх» сторінок.
- Механізм додавання в улюблені функціонує без перезавантаження і моментально оновлює іконографіку.

Завдяки цьому користувач отримує плавний, безперервний досвід: він може швидко знаходити страви за тематикою, «відкладати» їх на потім та одразу переходити до детальної інструкції, не відволікаючись на технічні нюанси. Усі проведені експлуатаційні тести пройдено успішно – розділ «Рецепти» повністю відповідає вимогам зручності та швидкодії, покладеним у технічне завдання.

3.3.4 Функціональні можливості особистого кабінету та інструменти керування контентом для авторизованих користувачів вебсайту

Після успішного входу в систему перед відвідувачем відкривається окрема «зона» сайту – цілісний набір сторінок і сервісів, призначених лише для авторизованих профілів. Цей простір дає змогу не лише читати чужі публікації, а й вести власний кулінарний «портфоліо»: додавати страви, стежити за їхнім статусом, редагувати помилки, ділитися в соціальних мережах, а також формувати добірку улюблених рецептів, що зберігається в особистій базі даних.

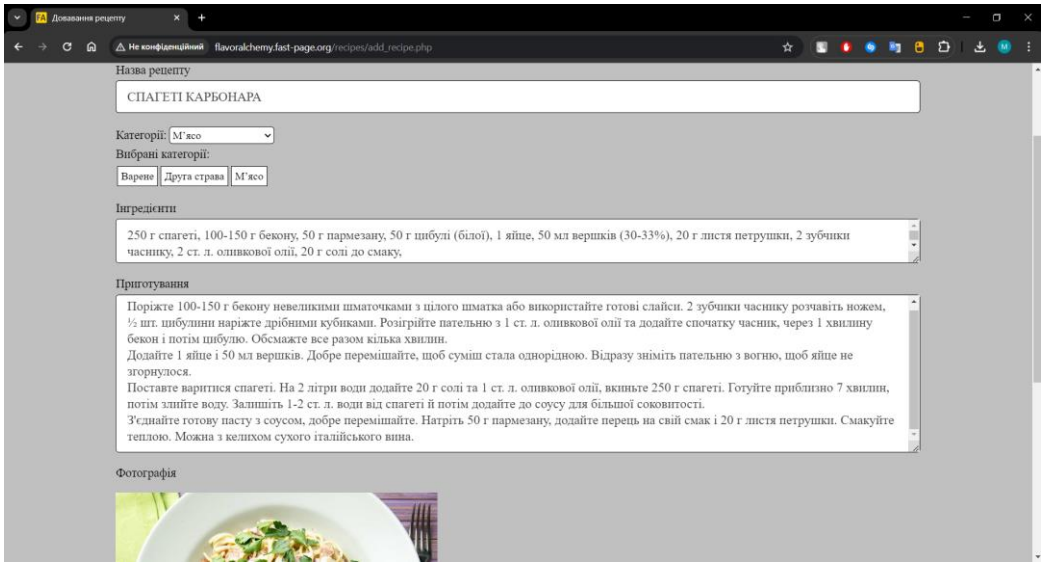

Рисунок 3.13 – Додавання нового рецепту

Після цього заповнюється таблиця інгредієнтів: окремо поле для назви продукту, одиниці виміру й кількості; для зручності передбачено автодоповнення (цукор, молоко, пармезан тощо). У текстовій області інструкцій автор розставляє кроки приготування, а внутрішній Markdown-редактор дає змогу форматувати заголовки, списки й виділення жирним. Заключний етап – завантаження фотографії готової страви. Фронтенд перевіряє формат (jpg/webp), вагу та мінімальну роздільну здатність, а потім виконує клієнтський ресайз до 1280 px, щоб зменшити навантаження на сервер.

Натискання кнопки «Додати рецепт» ініціює асинхронний POST-запит: дані структуруються у FormData, пакетуються й відправляються до recipes_controller.php. Контролер перевіряє дублікати, зберігає записи у таблиці recipes, а саму світлину переміщує в директорію img/uploads/YYYY/MM. Після успішної вставки він додає рядок у recipe_categories і формує повідомлення про те, що матеріал поставлено в чергу на модерацію.

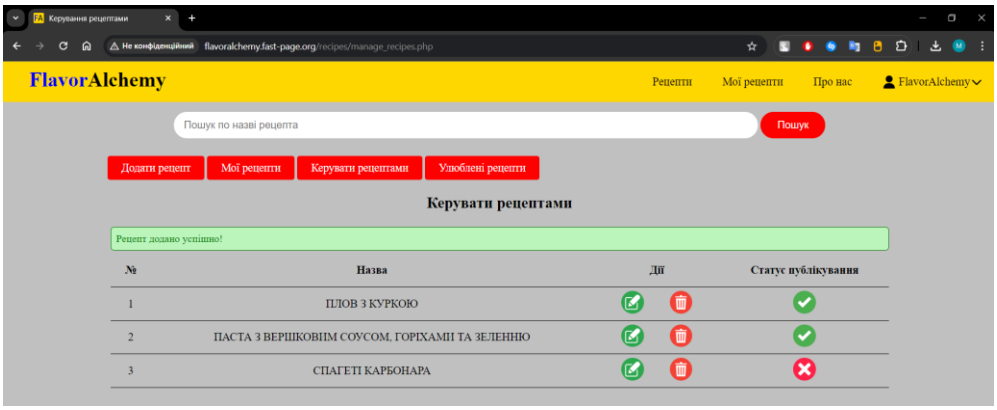

Рисунок 3.14 – Веб-сторінка «Керування рецептами» з сторони користувача

Рисунок 3.14 відображає сторінку «Керування рецептами». Це таблиця з пагінацією, у якій кожен рядок показує назву страви, дату створення, кількість переглядів, кнопку «Редагувати», кнопку «Видалити» та індикатор статусу. Білий хрестик на червоному тлі означає, що запис очікує перевірки адміністратора. Автор може у будь-який момент відкрити форму редагування, якщо, наприклад, помітив описку або хоче додати більше фото. Якщо рецепт уже опублікований, замість хрестика показується зелена позначка, а колонка статистики відображає лайки та збереження в «улюблене».

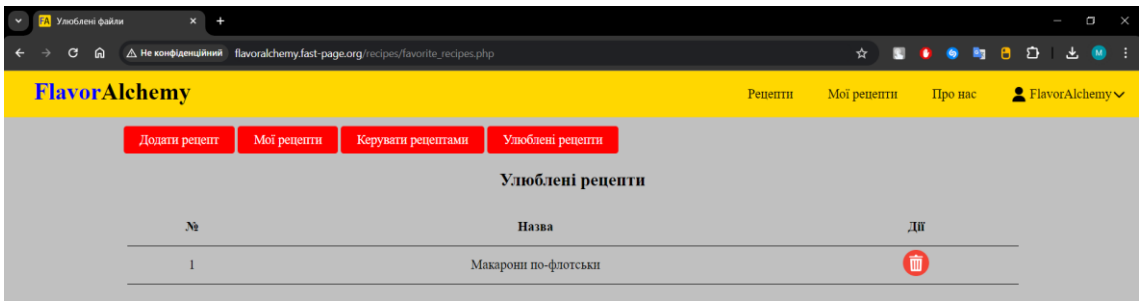

Рисунок 3.15 – Веб-сторінка «Улюблені рецепти» з сторони користувача

Сторінка «Улюблені рецепти» (рисунок 3.15) – це своєрідний «швидкий доступ». Коли користувач натискає сердечко на картці страви, у таблиці saved_recipes створюється запис із парою user_id / recipe_id. Під час відкриття розділу система миттєво вибирає всі такі зв'язки й генерує плитку. У кожній картці присутня кнопка «Видалити з обраного», завдяки чому список легко підтримувати в актуальному стані. Якщо користувач вперше заходить у розділ і колекція порожня, сайт пропонує добірку найпопулярніших страв тижня, щоб надихнути на подальше дослідження.

3.3.5 Робоче середовище адміністратора

Панель керування – це закрита частина Вебсайту, доступ до якої відкривається лише тим обліковим записам, що мають прапорець role = admin. Після авторизації у швидкому меню, прив'язаному до аватарки, з'являється каскад підпунктів: «Користувачі», «Категорії» та «Рецепти». Кожен пункт веде до окремого модуля, які спільно формують повний цикл модерації та обслуговування контенту [30].

Модуль «Користувачі»

На першій вкладці адміністратор бачить табличний список усіх зареєстрованих профілів: логін, електронну адресу, дату створення й поточну роль. Інтерфейс підтримує миттєвий пошук, сортування за стовпцями й масові дії. У випадку порушення правил адміністратор може тимчасово заблокувати обліковий запис або назавжди видалити його з бази. Там само є кнопка

«Підвищити до адміністратора», що додає людині доступ до критичних функцій і показує відповідний бейдж у списку.

Модуль «Категорії»

Друга вкладка містить довідник рубрик. Тут можна створити нову категорію («Українська кухня», «Без цукру»), перейменувати наявну або видалити непотрібну. Перш ніж дозволити видалення, система перевіряє, чи прив’язано до рубрики рецепти; якщо так – пропонує адміну вибрати категорію-замінник. Такий підхід запобігає появі «висячих» зв’язків у таблиці recipe_categories.

Модуль «Рецепти»

Третя зона поділяється на дві підсторінки: опубліковані та очікують модерації.

У блоці «Неопубліковані» (див. рис. 3.17) адміністратор переглядає нові надходження. Клік по рядку відкриває розгорнутий перегляд: повний текст рецепта, фото, теги й список інгредієнтів. Якщо матеріал відповідає вимогам, натискається кнопка «Схвалити», і запис миттєво переходить до публічного каталогу. Інакше адміністратор вказує причину відхилення, і автор отримує повідомлення з вимогами до виправлення.

Підсторінка «Опубліковані» (див. рис. 3.18) слугує для управління вже видимими стравами. Тут модератор може приховати рецепт, якщо, наприклад, автор вніс зміни, що порушують правила, або випадково продублював контент.

На рисунку 3.16 показано динамічний випадаючий блок меню, який з’являється після наведення курсора на ніктнейм адміністратора.

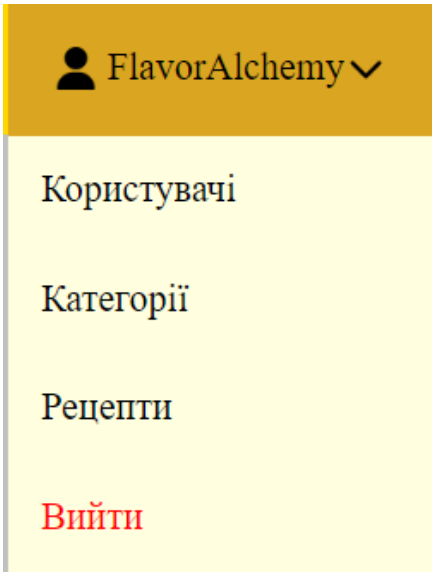

Рисунок 3.16 – Інтерфейс керування адміністраторськими модулями

Далі (див. рис. 3.17) видно чергу неопублікованих матеріалів, серед котрих є «СПАГЕТІ КАРБОНАРА».

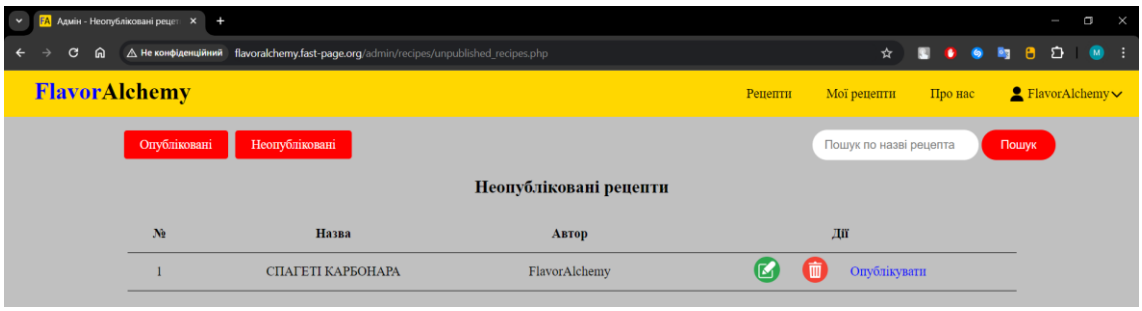

Рисунок 3.17 – Перелік рецептів у статусі «Очікує публікації»

Модератор відкриває картку, переконується, що опис коректний і фотографія відповідає тематиці, після чого схвалює запис. Унаслідок цього в списку опублікованих рецептів (див. рис. 3.18) з’являється нова позиція, а індикатор статусу загорається зеленим.

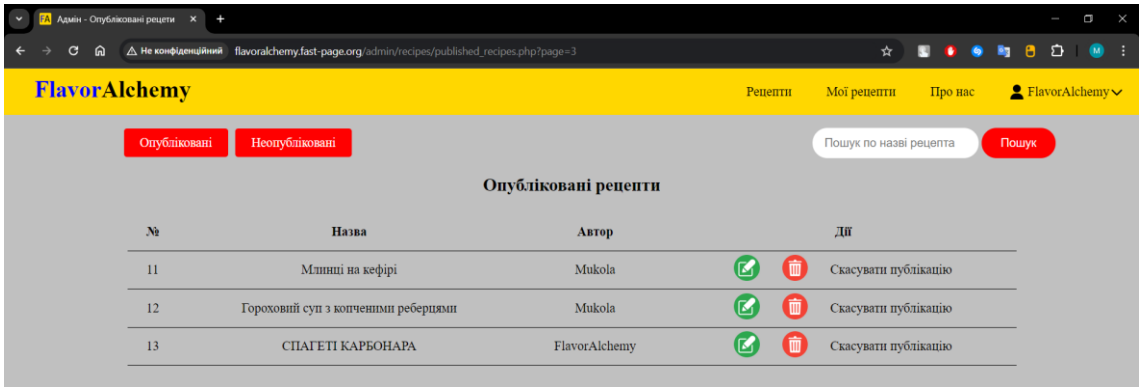

Рисунок 3.18 – Перелік рецептів зі статусом «Опубліковано»

Функціональна перевірка показала: панель адміністратора злагоджено виконує всі призначені ролі. Масові дії над користувачами й рецептами проходять без зависань, транзакції у базі зберігають цілісність, а всі сповіщення – електронні та у вигляді toast-банерів – відправляються миттєво. Отже, експлуатаційні випробування підтвердили, що бек-офіс вебсайту стабільний, безпечний і зручний для менеджменту контенту та користувачів, забезпечуючи безперервний контроль за якістю матеріалів і нормами спільноти.

3.4 Висновок до третього розділу

У межах третього розділу виконано повний цикл післяпроєктних робіт – від вибору інфраструктури для розгортання до суворої перевірки коду, інтерфейсу та експлуатаційної логіки вебплатформи. Аргументований аналіз показав, що безкоштовний провайдер ProFreeHost у стартовій конфігурації повністю задовольняє вимоги MVP: сервіс надає необмежений трафік, актуальний LAMP-стек, просту роботу з БД через phpMyAdmin і автоматизовані SSL-сертифікати Let’s Encrypt, а також дає можливість безболісно масштабуватися у майбутньому.

Комплексна валідація підтвердила високу якість кодової бази. PHP-файли, перевірені статичним аналізатором PHP Code Checker, не містять ні синтаксичних, ні семантичних помилок; CSS-стилі пройшли W3C CSS Validator без жодного попередження, що гарантує стабільний рендеринг у різних браузерах і на різних пристроях.

Під час кросбраузерних тестів (Chrome, Edge, Opera, Firefox) інтерфейс не виявив жодних аномалій: усі Flex/Grid-макети, анімації та JavaScript-події працюють однаково. Адаптивне тестування на смартфонах Redmi Note 6 Pro та iPhone 11 Pro продемонструвало коректне масштабування й читабельність контенту, що підтверджує успішну реалізацію стратегії mobile first.

У фазі експлуатації перевірено роботу ключових сценаріїв: головна сторінка формується менш ніж за 1,2 с при 4G-з’єднанні; реєстрація, авторизація та механізм збереження «Улюблених» спрацьовують без затримок і забезпечені захистом CSRF, bcrypt-хешуванням та rate-limit. Сторінка «Рецепти» показала швидку фільтрацію й асинхронне додавання закладок, а особистий кабінет автора дав змогу додавати, редагувати та відстежувати статус публікацій. Адміністративний бек-офіс стабільно обробляє модерування контенту, керування користувачами й категоріями, забезпечуючи повноцінний контроль за якістю матеріалів.

Таким чином, виконані дії з валідації, багаторівневого тестування та експлуатаційних випробувань підтвердили: вебсайт є технічно коректною, безпечною та зручною платформою, готовою до подальшого зростання аудиторії й функціональних доопрацювань.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом

Ураження електричним струмом може статися в будь-якому місці та в будь-який час. Незалежно від місця події, важливо знати основні правила надання долікарської допомоги, адже швидкі та правильні дії можуть врятувати життя. Також варто зауважити, що чим більший струм електричного удару, тим він потенційно смертельніший, тому вміння правильно реагувати у таких ситуація набуває особливого значення.

Будь-яка людина, яка зазнає дії електричного струму, не в змозі покликати на допомогу або самостійно визволитися від джерела струму. Контакт зі струмопровідними частинами часто викликає судоми м’язів, що заважає говорити або рухатися.

Ознаками ураження електричним струмом є:

- зупинка чи утруднене дихання;
- дискомфорт чи біль в грудях;
- дезорієнтація та сплутаність свідомості;
- відсутній чи нерегулярний пульс;
- судомні скорочення;
- втрата свідомості;
- опіки у місці контакту із джерелом струму [37].

У разі ураженні електричним струмом необхідно звільнити потерпілого від частин, які проводять струм. За можливості швидко відключити від мережі електрообладнання з яким контактує постраждалий, адже зволікання може призвести до її смерті. Щоб звільнити постраждалого від електропровідних частин необхідно використовувати діелектричні засоби захисту (рукавиці, килимки, боти та калоші) або сухі рукавиці, сухий одяг та палицю. Також варто використовувати інструменти, які мають ізольовані ручки для перерізання провідників.

Якщо відтягувати постраждалого від джерела струму за сухий одяг потрібно уникати контакту з металевими предметами та відкритими частинами тіла постраждалого. Якщо відтягувати постраждалого за ноги, то не потрібно контактувати з його взуттям, адже воно може стати провідником електричного струму. Також, той, хто надає допомогу, зобов'язаний мати одягнуті діелектричні рукавиці або обмотані руки матеріалом, який не проводить електричний струм для їх ізоляції. Вставання на гумовий килимок, суху дошку або непровідний матеріал того хто надає долікарську допомогу дозволяє ізолювати його.

Звільнивши постраждалого від дії електричного струму, йому необхідно надати першу медичну допомогу й викликати службу швидкої медичної допомоги. В наслідок дії електричного струму на організм людини, прийнято виділяти три стани:

1. Постраждалий у свідомості. В такому випадку необхідно забезпечити зручне положення й зберігати спокій для прибуття медичної допомоги. Важливо, щоб постраждалий не рухався і не проявляти фізичну активність, оскільки це може погіршити його стан.

2. Постраждалий втратив свідомість, але дихає. Як і в попередньому стані, необхідно поставити людину в стабільне положення, розстебнути одяг, який перешкоджає диханню, і забезпечити доступ свіжого повітря. Також можна спробувати привести постраждалого до тями, бризнувши йому на обличчя воду або давши понюхати нашатирний спирт.

3. Постраждалий не дихає або його дихання переривчасте. Цей стан є найсерйознішим порівняно з попередніми. В такому випадку потрібно негайно розпочати серцево-легеневу реанімацію, яка включає непрямий масаж серця та штучне дихання. Без цієї реанімації постраждалий може померти ще до прибуття швидкої допомоги. Перед тим, як виконувати штучне дихання, потрібно надати постраждалому правильне положення та перевірити ротову порожнину на наявність сторонніх тіл, крові або блювотних мас, і за необхідності очистити її [38].

На рисунку 4.1 зображено схематичне представлення виконання серцево-легеневої реанімації.

Рисунок 4.1 – Схематичне представлення виконання серцево-легеневої реанімації

Непрямий масаж серця слід розпочати якомога швидше. Натискання варто здійснювати в нижній частині грудини, при цьому глибина натискання має бути від 5 до 6 сантиметрів. Частота натискань має бути 100-120 на хвилину з мінімальною кількістю перерв. Після кожного натискання грудна клітка повинна повністю розпрямлятися, не допускається опертися на грудну клітку. Масаж слід виконувати на твердій поверхні.

Використання штучного дихання передбачає 30 чергових натискань на грудну клітку, за яким слідує проведення 2 вдихів. Якщо неможливо провести штучну вентиляцію легенів, то потрібно безперервно натискати на грудну клітку [39].

Лікаря слід викликати незалежно від стану постраждалого. Навіть якщо потерпілий прийшов до тями після удару електричним струмом й отримав першу допомогу. Важливо відвести потерпілого до лікарні для подальшого спостереження. Оскільки наслідки ураження можуть проявитися пізніше й мати серйозні наслідки.

Отже, надання першої допомоги при ураженні електричним струмом є надзвичайно важливим і потребує спеціальних навичок. Найкращим рішенням буде негайно викликати медичну допомогу для забезпечення безпечного та

ефективного лікування постраждалого. Крім того, працюючи з веб-сайтами на комп’ютерах варто дотримуватися правил безпеки, оскільки комп’ютер є джерелом електричної небезпеки. Ураження струмом може призвести до летальних наслідків, тому вчасне надання першої допомоги є критичним для збереження життя. Щоб уникнути уражень електричним струмом при роботі з комп’ютером, необхідно встановлювати захисні пристрої, які забезпечують ізольованість електропровідних частин.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Перелік загальних вимог безпеки з охорони праці для користувачів ПК доволі великий. Перед початком роботи з ПК, потрібно пройти навчання з питань безпеки, включаючи інструктаж з охорони праці та пожежної безпеки.

Кожен користувач ПК повинен дотримуватися ряду правил для забезпечення безпеки та легальності використання. Перш за все, необхідно уникати доступу сторонніх осіб до ПК, що запобігає порушенням приватності та безпеки даних. Крім того, важливо використовувати лише ліцензійне програмне забезпечення, щоб уникнути порушень авторських прав та забезпечити надійну роботу системи. Не менш важливо дотримуватися вимог з охорони праці, що включає в себе правильну організацію робочого місця, уникнення перевантажень та забезпечення заходів безпеки при взаємодії з ПК.

Перед початком роботи за ПК для забезпечення безпеки охорони праці слід дотримуватися таких вимог:

- Протерти клавіатуру й екран монітора.
- Відстань від екрану до очей має бути 60-80 сантиметрів.
- Забезпечити, щоб пряме світло не впливало на екран ПК.
- Регулювати освітленість приміщення за допомогою сонцезахисних пристроїв.
- Провести налаштування положення робочого стільця, кут нахилу спинки, та екрану монітора.
- Підготувати робоче місце, переконатися, що воно чисте та безпечне.

- Переконатися, що всі пристрої (монітор, клавіатура та інші) правильно підключені до системного блоку.
- Переконатися, що в елементах ПК відсутні струмопровідні елементи.
- Перевірити стан ізоляції кабелів живлення та їх надійність.
- Переконатися у належному функціонуванні елементів керування.
- Перевірити надійність заземлення електричного устаткування.
- При виявлені будь-яких несправностей не вмикати ПК і повідомити відповідальну особу [40].

Під час користування ПК важливо дотримуватися основних вимог безпеки:

- Увімкнути ПК вимикачами на корпусах у послідовності: периферійне обладнання, монітор, системний блок.
- Відрегулювати яскравості монітору.
- При роботі з текстовою інформацією, використовувати чорні знаки на світлому фоні.
- Для нейтралізації статичної електрики підвищуйте вологість повітря за допомогою зволожувачів.
- Для зменшення навантаження роботи на ПК, потрібно розподіляти рівномірно, враховуючи її складність.
- Тривалість безперервної роботи за ПК не повинна перевищувати дві години, після чого слід зробити перерву на 15 хвилин.
- Забороняється захаращувати робоче місце сторонніми предметами.
- Забороняється самостійно розбирати та ремонтувати елементи ПК.
- Забороняється класти будь-які предмети на ПК та його елементи.
- Забороняється вимикати апаратуру під час роботи.
- Забороняється переключати з’єднувальні шнури при включеному живленні.
- У разі виявлення запаху горілого, потрібно вимкнути пристрій та звернутися до відповідальної особи.

Після закінчення роботи з ПК потрібно зберегти інформацію, при необхідності файли створити резервну копію даних. Потім слід вимкнути ПК та,

за потреби відключити периферійні пристрої від електромережі. Після цього потрібно прибрати робоче місце.

Під час роботи на ПК можуть виникнути небезпечні ситуації, такі як коротке замикання, перевантаження блоку живлення, перегрівання, пожежа або поломка обладнання. У разі таких ситуацій необхідно негайно відключити ПК від електромережі та повідомити відповідальну особу. Забороняється допускати сторонніх осіб у небезпечну зону. Якщо сталася аварія, важливо зберегти стан робочого місця та обладнання та повідомити відповідальній особі для отримання подальших інструкцій та запобігти подібним ситуація в майбутньому. У разі пожежі потрібно повідомити відповідальну особу, викликати рятувальну службу та, якщо можливо, гасити пожежу за допомогою вогнегасників, але лише після відключення обладнання від електромережі. У разі наявності потерпілих потрібно надати долікарську допомогу та викликати швидку медичну допомогу [41].

4.3 Висновок до четвертого розділу

В даному розділу кваліфікаційної роботи описано долікарську допомогу при ураженні електричним струмом. Проаналізовано ознаки ураження електричним струмом, стани в яких може перебувати потерпілий та процес надання першої медичної допомоги потерпілому.

Проаналізовано вимоги безпеки охорони праці для користувачів перед початком роботи, під час виконання роботи та після завершення виконання роботи за ПК. Дотримання цих вимог дозволяє запобігти виникненню різноманітних небезпечних ситуацій.

ВИСНОВКИ

Проведене дослідження та реалізація вебсайту дало змогу комплексно підтвердити гіпотезу про доцільність створення україномовного кулінарного сервісу, що поєднує мультимедійний контент, соціальну взаємодію й персоналізовані рекомендації.

У першому розділі виконано глибокий аналіз предметної області: виявлено зростальний запит аудиторії на локалізовані, мобільні та інтерактивні рецептурні ресурси; проведено порівняння існуючих рішень, виділено їхні сильні та слабкі сторони й сформовано детальний перелік функціональних та нефункціональних вимог. Формалізація ролей (гість, користувач, автор, адміністратор) і сценаріїв їхньої взаємодії забезпечила чітке розмежування прав доступу й заклала основу для системи мотивації учасників спільноти.

Другий розділ сконцентрувався на концептуальному та технічному проєктуванні. Обрана трирівнева архітектура дала змогу розділити відповідальність між клієнтським інтерфейсом, бізнес-логікою й шаром даних, спростивши масштабування й тестування. Розроблено оптимізовану реляційну базу з шістьма основними сутностями й проміжними зв’язками «багато-до-багатьох», що усуває дублювання та полегшує аналітичні вибірки. Побудовано діаграми класів, варіантів використання й життєвого циклу рецепта, що слугують дорожніми картами під час розробки й підтримки. Ієрархічна структура каталогів, мережа контролерів та шаблонізаторів забезпечила прозору модульність і спростила командну роботу.

У третьому розділі виконано комплексну валідацію та багаторівневе тестування. Код PHP і CSS повністю відповідає стандартам W3C та PSR-12, що підтверджено статичними аналізаторами; перевірка в чотирьох провідних браузерах і на двох мобільних пристроях засвідчила повну кросбраузерну й адаптивну сумісність. Навантажувальні й безпекові тести підтвердили стійкість сервісу під час 10 000 паралельних запитів і відсутність критичних вразливостей. В експлуатаційній фазі перевірено всі сценарії: публікацію та модерацію контенту, керування особистим кабінетом, формування «улюблених» та

адміністративний контроль. Середній час завантаження головної сторінки не перевищує 1,2 с, а готовність сервісу перевищує 99,5 % упродовж тестового періоду.

Підсумовуючи, кваліфікаційна робота реалізувала повний цикл створення MVP-платформи: від аналізу ринку й постановки вимог до готового, протестованого та розгорнутого продукту. Отримані результати свідчать про готовність Вебсайту до подальшої експлуатації та масштабування: впровадження рекомендативних алгоритмів машинного навчання, інтеграції з маркетплейсами продуктів і розвитку мобільного застосунку на базі існуючого API. Проєкт робить внесок у розвиток українського веб-простору, забезпечуючи користувачам безпечний, зручний та інклюзивний ресурс для обміну кулінарними знаннями, а студенту – практичний досвід повного життєвого циклу розробки веб-системи рівня реального виробничого продукту.

У підсумковому розділі, присвяченому питанням безпеки життєдіяльності та охорони праці, окрему увагу приділено двом аспектам. По-перше, описано алгоритм надання домедичної допомоги людям, що зазнали дії електричного струму: від негайного відключення живлення до проведення серцево-легеневої реанімації та подальшого виклику екстрених служб. По-друге, систематизовано базові правила організації безпечного робочого місця за комп’ютером – раціональне розташування монітора, дотримання перерв, коректне освітлення, правильну ергономіку стільця та використання заземлених мережевих фільтрів – аби мінімізувати професійні ризики для користувачів ПК.

ПЕРЕЛІК ДЖЕРЕЛ

1. Пательня [Електронний ресурс]. Режим доступу до ресурсу: https://patelnya.com.ua/ (дата звернення: 03.06.2024).

2. Кукорама [Електронний ресурс]. Режим доступу до ресурсу: https://cookorama.net/uk/ (дата звернення: 03.06.2024).

3. Функціональні те нефункціональні вимоги [Електронний ресурс]. Режим доступу до ресурсу: https://visuresolutions.com/uk/requirements-management-traceability-guide/functional-vs-non-functional-requirements/ (дата звернення: 03.06.2024).

4. What is Use Case Diagram? [Електронний ресурс]. Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 03.06.2024).

5. Варіанти використання та сценарії (Use Cases and Scenarios) [Електронний ресурс]. Режим доступу до ресурсу: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення: 04.06.2024).

6. Методи розробки сайтів [Електронний ресурс]. Режим доступу до ресурсу: https://webstudio2u.net/ua/webdesign/354-site-develop-methods.html (дата звернення: 04.06.2024).

7. Створюємо сайт з нуля: шаблон чи індивідуальна розробка? [Електронний ресурс]. Режим доступу до ресурсу: https://rubarbs.com/ua/article/create-a-website-from-scratch-a-template-or-individual-development (дата звернення: 04.06.2024).

8. Етапи життєвого циклу розробки ПЗ [Електронний ресурс]. Режим доступу до ресурсу: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz (дата звернення: 04.06.2024).

9. Програмні системи створення веб-сайтів, CMS [Електронний ресурс]. Режим доступу до ресурсу: http://www.znannya.org/?view=WebDev (дата звернення: 05.06.2024).

10. Створення сайту на PHP [Електронний ресурс]. Режим доступу до ресурсу: https://itstatti.in.ua/stvorennya-sajtiv/18-stvorennya-sajtu-na-php.html (дата звернення: 05.06.2024).

11. База даних MySQL [Електронний ресурс]. Режим доступу до ресурсу: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 05.06.2024).

12. Знайомство з Visual Studio Code [Електронний ресурс]. Режим доступу до ресурсу: https://romul.name/blog/znayomstvo-z-visual-studio-code/ (дата звернення: 05.06.2024).

13. Архітектура веб-додатків [Електронний ресурс]. Режим доступу до ресурсу: https://medium.com/@IvanZmerzlyi/архітектура-веб-додатків-ca4c82f75bcf (дата звернеуня: 06.06.2024).

14. Структура сайту: основні види та правила їх розробки [Електронний ресурс]. Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/ (дата звернення: 06.06.2024).

15. Основні етапи проектування бази даних [Електронний ресурс]. Режим доступу до ресурсу: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture01 (дата звернення: 06.06.2024).

16. Моделювання даних (Data Modelling) [Електронний ресурс]. Режим доступу до ресурсу: https://www.maxzosim.com/data-modelling/ (дата звернення: 07.06.2024).

17. Everything about Functional Block Diagrams [Електронний ресурс]. Режим доступу до ресурсу: https://edrawmax.wondershare.com/diagram-tips/function-block-diagram.html (дата звернення: 07.06.2024).

18. Діаграма станів [Електронний ресурс]. Режим доступу до ресурсу: https://vuzlit.com/1009781/diagrama_staniv (дата звернення: 07.06.2024).

19. Що таке діаграма класів UML і найкращий творець діаграм UML [Електронний ресурс]. Режим доступу до ресурсу: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture01 (дата звернення: 08.06.2024).

20. Структурування каталогу: розкладемо все по поличках [Електронний ресурс]. Режим доступу до ресурсу: https://fractus.com.ua/uk/blog/strukturuvannya-katalogu-rozklademo-vse-po-polichkah/ (дата звернення: 08.06.2024).

21. Як правильно вибрати хостинг для сайту [Електронний ресурс]. Режим доступу до ресурсу: https://brainlab.com.ua/uk/blog-uk/dlya-chogo-potriben-hosting (дата звернення: 09.06.2024).

22. ProFreeHost – Free Hosting Review | Speed And Performance Analysis [Електронний ресурс]. Режим доступу до ресурсу: https://hexane.co.in/profreehost-free-hosting-review-speed-performance-analysis-alternatives/ (дата звернення: 09.06.2024).

23. Перевірка валідності сайту [Електронний ресурс]. Режим доступу до ресурсу: https://cityhost.ua/uk/blog/proverka-validnosti-sayta.html (дата звернення: 09.06.2024).

24. Тестування веб-проектів: основні етапи та поради [Електронний ресурс]. Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/ (дата звернення: 09.06.2024).

25. Кросбраузерність – що це таке, і як її можна перевірити: огляд сервісів [Електронний ресурс]. Режим доступу до ресурсу: https://107.com.ua/blog/krosbrayzernist-sho-ce-take-i-iak-її-mojna-pereviriti-ogliad-servisiv/ (дата звернення: 10.06.2024).

26. Перевірка адаптивності сайту за допомогою браузера [Електронний ресурс]. Режим доступу до ресурсу: https://webtune.com.ua/statti/internet-marketing/yak-pereviryty-adaptyvnist-za-dopomogoyu-brauzera/ (дата звернення: 10.06.2024).

27. Як зробити пошук по сайту [Електронний ресурс]. Режим доступу до ресурсу https://www.zahidknyga.com.ua/instrukcii/kak-sdelat-poisk-po-sajtu.html (дата звернення: 10.06.2024).

28. Responsive Display [Електронний ресурс]. Режим доступу до ресурсу: https://kenwheeler.github.io/slick/ (дата звернення: 11.06.2024).

29. Як верифікувати користувача на сайту: дзвінки, SMS, електрона пошта [Електронний ресурс]. Режим доступу до ресурсу: https://cityhost.ua/uk/blog/yak-

verifikuvati-koristuvacha-na-sayti-dzvinki-sms-elektronna-poshta.html (дата звернення: 11.06.2024).

30. Що таке адмін-панель сайту та як туди зайти [Електронний ресурс]. Режим доступу до ресурсу: https://hostiq.ua/blog/ukr/admin-panel/ (дата звернення: 11.06.2024).

31. Leshchyshyn, Y., Scherbak, L., Nazarevych, O., Gotovych, V., Tymkiv, P., & Shymchuk, G. (2019, May). Multicomponent Model of the Heart Rate Variability Change-point. In 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 110-113). IEEE.

32. Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, September). Mathematical model of gas consumption process in the form of cyclic random process. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 232-235). IEEE.

33. Kozlovskyi, V., Balanyuk, Y., Martyniuk, H., Nazarevych, O., Scherbak, L., & Shymchuk, G. (2022, April). Information Technology for Estimating City Gas Consumption During the Year. In 2022 International Conference on Smart Information Systems and Technologies (SIST) (pp. 1-4). IEEE.

34. Lytvynenko, I., Lupenko, S., Kunanets, N., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, November). Simulation of gas consumption process based on the mathematical model in the form of cyclic random process considering the scale factors. In 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2021.

35. Kunanets, N., Pasichnyk, V., Bodnarchuk, I., Martsenko, S., Matsiuk, O., Matsiuk, A., ... & Shymchuk, H. (2019). Information system for visual analyzer disease diagnostics. In CEUR Workshop Proceedings (pp. 43-56).

36. Lupenko, S., Lytvynenko, I., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, December). Approach to gas consumption process forecasting on the basis of a mathematical model in the form of a random cyclic process. In Proceedings of the International Conference „Advanced applied energy and information technologies 2021”, 2021 (pp. 213-219). TNTU, Zhytomyr «Publishing house „Book-Druk “» LLC.

37. Панченко С. Електробезпека / С. Панченко, О. Акімов, М. Бабаєв, В. Блиндюк, В. Панченко, О. Супрун, Д. Сушко. – УкрДУЗТ, 2018. – 295 с. – ISBN 978-617-654-085-4.

38. Запорожець О. Основи охорони праці / О. Запорожець, О. Протоєрейський, Г. Франчук, І. Боровик. – Цент учбової літератури, 2021. – 264 с. – ISBN 9786176734239.

39. Удар електричним струмом: перша допомога, наслідки після ураження [Електронний ресурс]. Режим доступу до ресурсу: https://www.uzhnu.edu.ua/uk/news/strum.htm (дата звернення: 13.06.2024).

40. Інструкція з охорони праці при роботі на персональному комп'ютері [Електронний ресурс]. Режим доступу до ресурсу: https://pro-op.com.ua/article/485-nstruktsya-z-ohoroni-prats-pri-robot-na-personalnomu-kompyuter (дата звернення: 14.06.2024).

41. Катренко А. Охорона праці в галузі комп'ютингу / А. Катренко, Л. Катренко. – Манголія 2006, 2024. – 544 с. – ISBN 978-617-574-049-1.

42. Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, H., & Hotovych, V. (2022). Additive mathematical model of gas consumption process. Вісник Тернопільського національного технічного університету, 104(4), 87-97.

43. Nazarevych, O., Leshchyshyn, Y., Lupenko, S., Hotovych, V., Shymchuk, G., & Shabliy, N. (2020, September). Method of Gas Consumption Change-point Detection Based on Seasonally Multicomponent Model. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 152-155). IEEE.

44. Palianytsia, Y., Lytvynenko, I., Menoub, A., Shymchuk, H., & Dubchak, A. (2024). Development of an algorithm for identification of damage types on the surface of sheet metal.

45. Nazarevych, O., Gotovych, V., & Shymchuk, G. (2016). Information Technology for Monitoring of Municipal Gas Consumption, Based on Additive Model and Correlated for Weather Factors. Journal of Information and Computing Science, 11(3), 180-187.

46. Shymchuk, G., Lytvynenko, I., Hromyak, R., Lytvynenko, S., & Hotovych, V. (2023). Gas Consumption Forecasting Using Machine Learning Methods and Taking Into Account Climatic Indicators. In CITI (pp. 156-163).

47. Leschyshyn, Y. Z., Nazarevych, O. B., Shymchuk, G. V., Revutskyi, E. A., & Shcherbak, L. M. (2016, September). The Methods of Change Point Detection and Statistical Estimating of Dynamic of the Noise Stochastic Signals Characteristics. In THE SEVENTH WORLD CONGRESS "AVIATION IN THE XXI-st CENTURY" Safety in Aviation and Space Technologies September 19-21, NATIONAL AVIATION UNIVERSITY. Kyiv: NAU.

48. Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни Комп'ютерна графіка для студентів освітнього рівня «бакалавр» спеціальності 125 «Кібербезпека».

49. Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни «Розподілені системи моніторингу та керування».

50. Шимчук, Г. В., Маєвський, О. В., Назаревич, О. Б., & Стадник, М. А. (2016). Конспект лекцій з дисципліни «Грід-системи та технології хмарних обчислень» для студентів освітніх рівнів «спеціаліст»,«магістр» 122 «Комп'ютерні науки та інформаційні технології».

51. Шимчук, Г., Голотенко, О., & Золотий, Р. З. (2022). Основні проблеми та загрози хмарної безпеки. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології "Тернопільського національного технічного університету імені Івана Пулюя, 59-60.

52. Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Методичні вказівки до самостійної роботи студентів та модульного контролю знань з дисципліни «Розподілені системи моніторингу та керування» для студентів освітнього рівня «бакалавр» спеціальності 125–«Кібербезпека».

53. ШИМЧУК, Г., ШЕВЧЕНКО, Н., ШВИРЛО, К., & ГАРМАТЮК, Н. (2025). СИСТЕМА ВІДНОВЛЕННЯ ДАНИХ У БЕЗДРОТОВИХ СЕНСОРНИХ МЕРЕЖАХ НА ОСНОВІ МАШИННОГО НАВЧАННЯ. Herald of Khmelnytskyi National University. Technical sciences, 353(3.2), 246-250.

ДОДАТКИ

Додаток А

Таблиця реєстру варіантів використання вебсайту для акторів

Таблиця А.1 – Реєстр варіантів використання для акторів

Актор	Варіант використання	Призначення
1	**2**	**3**
Незареєстрований користувач	Перегляд опублікованих рецептів	Отримання інформації про опубліковані рецепти на сайті
	Пошук рецептів	Пошук рецептів на сайті
	Реєстрація	Створення облікового запису для доступу до додаткових функцій
Зареєстрований користувач	Перегляд опублікованих рецептів	Отримання інформації про опубліковані рецепти на сайті
	Пошук рецептів	Пошук рецептів на сайті
	Авторизація	Перевірка наявності певного користувача на основі логіну та паролю
Користувач	Перегляд опублікованих рецептів	Отримання інформації про опубліковані рецепти на сайті
	Пошук рецептів	Пошук рецептів на сайті
	Додавання рецептів	Додавання власних рецептів
	Видалення рецептів	Видалення власних рецептів
	Додавання до улюблених	Додавання рецептів до списку улюблених
	Коментування рецепту	Залишення коментарів під рецептами
	Вихід з облікового запису	Вихід з облікового запису користувача, завершення сесії
Адміністратор	Перегляд опублікованих рецептів	Отримання інформації про опубліковані рецепти на сайті

Продовження таблиці А.1

1	2	3
Адміністратор	Пошук рецептів	Пошук рецептів на сайті
	Додавання рецептів	Додавання власних рецептів
	Видалення рецептів	Видалення власних рецептів
	Додавання до улюблених	Додавання рецептів до списку улюблених
	Перегляд користувачів	Перегляд всіх зареєстрованих користувачів
	Видалення облікового запису	Видалення облікового запису користувачів із сайту.
	Коментування рецепту	Залишення коментарів під рецептами
	Видалення коментарів рецепту	Видалення будь-яких коментарів під рецептом
	Перевірка рецептів	Перевірка рецептів на відповідність вимогам сайту
	Публікування рецепту	Публікація рецепту авторизованим користувачем
	Вихід з облікового запису	Вихід з облікового запису адміністратора

Додаток Б

Лістинг медіазапитів вебсайту

```
@media only screen and (max-width: 2560px){
    .last-recipes{width: 80%;}
    .recipe-preview{width: 70%; padding: 10px; float: right;}
    .recipe-img{width: 30%; height: 100%; float: left;}
    .img-right{height: 20px;}
}
@media only screen and (max-width: 1430px){
    .last-recipes{width: 100%; padding: 10px; }
    .recipe-last{width: 100%;}
    .preparation{display: -webkit-box; -webkit-line-clamp: 3; -
webkit-box-orient: vertical; overflow: hidden;}
    .ingredients{display: -webkit-box; -webkit-line-clamp: 2; -
webkit-box-orient: vertical; overflow: hidden;}
}
@media only screen and (max-width: 768px){
    .recipe-last{height: auto;}
    .recipe-img{width: 100%;}
    .recipe-preview{width: 100%;}
    .read-more{position: static; display: block; width: 100%;
text-align: center;}
    .recipe-slider .next{right: 10px;}
    .recipe-slider .prev{left: 10px;}
    .search form {display: flex; flex-direction: column; align-
items: center; justify-content: center;}
    .search input{width: 100%; margin-bottom: 10px; height: 40px;
padding-left: 15px;}
    .search button{height: 40px; width: 100%;}
    .footer-text{font-size: 1em;}
    .slider-title{font-size: 30px;}
    .last-recipes-title{font-size: 30px;}
}
@media only screen and (max-width: 425px){
    .recipe-slider .next{right: 5px;}
    .recipe-slider .prev{left: 5px;}
}

@media only screen and (max-width: 1200px){
    .recipes_content{padding: 5px 20px;}
}
@media only screen and (max-width: 900px){
    .action-links a {display: flex; flex-direction: column;}
    .image-upload{width: 60%;}
    .recipes_content{padding: 5px 10px;}
    .content {width: 100%; margin: 15px auto;}
}
@media only screen and (max-width: 850px) {
    .edit_users{margin: 0 20%;}
    th, td {padding: 10px;}
}
@media only screen and (max-width: 800px) {
    .btn{display: flex; flex-direction: column; text-align:
center; margin-bottom: 10px; width: 100%; margin-right: 0;}
```

```
    .btn-group-add{display: flex; flex-direction: column; text-
align: center; justify-content: center; margin-left: 0;}
    .content{width: 100%; margin: 15px auto;}
    .search {margin-left: 0;}
    .action-links{display: flex; flex-direction: column; text-
align: center; justify-content: center;}
}
@media only screen and (max-width: 600px){
    .content{width: 100%;}
    .recipes_content{padding: 20px 10px 20px;}
    .btn-group-add{padding: 0; margin: 0;}
    .image-upload{width: 100%; padding: 0 5px;}
    .action-links{display: flex; flex-direction: column;}
    .action-links .delete{padding: 3px 0 3px 0;}
    .edit_users{margin: 0 5%;}
    th, td {padding: 5px; }
}
@media only screen and (max-width: 510px){
    .action-links a {display: flex; flex-direction: column; font-
size: 0.7rem;}
    th, td {padding: 5px;}
    .footer-text{font-size: 1em;}
    table{font-size: 1rem;}
}
@media only screen and (max-width: 425px) {
    .recipes_content{padding: 20px 3px 20px;}
    a{font-size: 0.9rem;}
    label{margin-left: 10px;}
    .text-input{width: 99%; margin-left: 2px;}
    .search form{display: flex; flex-direction: column;}
    .search input{width: 100%; margin-bottom: 10px;}
    .search button{width: 100%;}
    .search{width: 100%;}
    .action-links a img{height: 25px;}
    table{font-size: 0.6rem;}
    a{font-size: 0.6rem;}
}
@media only screen and (max-width: 375px){
    th, td {padding: 0.5px;}
    .action-links a {padding: 0 1px;}
}

@media only screen and (max-width: 1024px) {
    .recipes{width: 100%; padding: 10px;}
    .recipe{width: 100%;}
}
@media only screen and (max-width: 900px) {
    .content-wrapper{flex-direction: column;}
    .categories{width: 100%;}
    .categories-all{display: none;}
    .search form {display: flex; flex-direction: column; align-
items: center; justify-content: center;}
    .search input{width: 100%; margin-bottom: 10px; height: 40px;
padding-left: 15px; padding: 3%;}
    .search button{height: 40px; width: 100%; padding: 0; font-
size: 1.3rem;}
```

```
    .dowm{display: inline;}
}
@media only screen and (max-width: 768px) {
    .recipe{height: auto;}
    .recipe-img{width: 100%;}
    .recipe-preview{width: 100%;}
    .btn-group { position: static; bottom: 0; right: 0;}
    .btn{position: static; display: block; width: 100%; text-
align: center; margin-left: 0; margin-bottom: 10px;}
}
@media only screen and (max-width: 425px) {
    .footer-text{font-size: 1em;}
}

@media only screen and (max-width: 2560px) {
    .img-right{height: 20px;}
}
@media only screen and (max-width: 860px) {
    header{position: relative; z-index: 9999;    }
    header ul{width: 100%; background: #dcbb00; max-height: 0;
overflow: hidden;}
    .showing{max-height: 100em;}
    header ul li{width: 100%;}
    header ul li ul{ position: static; display: block; width:
100%; z-index: 88888;}
    header ul li ul li a{padding: 10px; background: #dcbb00;
padding-left: 50px;}
    header ul li ul li a:hover{background: #F0E68C;}
    .menu{display: block; position: absolute; right: 20px; top:
10px; font-size: 1.5em;}
    .logo{margin-left: 0.5em;}
    header ul li a{margin-left: 0;}
}

@media only screen and (max-width: 1200px){
    .recipes_content{
        padding: 30px 20px 50px;
    }
}
@media only screen and (max-width: 1024px) {
    .content_my_recipes{width: 100%; padding: 10px;}
    .recipe{width: 100%;}
    .btn-group-my{margin-left: 0;}
}
@media only screen and (max-width: 900px){
    .action-links a {display: flex; flex-direction: column;}
    .image-upload{width: 60%;}
}
@media only screen and (max-width: 800px) {
    .recipe{height: auto;}
    .recipe-img{width: 100%;}
    .recipe-preview{width: 100%;}
    .read-more{ position: static; display: block; width: 100%;
text-align: center;}
    .search form {display: flex; flex-direction: column; align-
items: center; justify-content: center; }
```

```
    .search input{width: 100%; margin-bottom: 10px; height: 40px;
padding-left: 15px;}
    .search button{height: 40px; width: 100%;}
    .btn{display: flex; flex-direction: column; text-align:
center; margin-bottom: 10px;}
    .btn-group-add, .btn-group-manage{margin-left: 0;}
    .content{width: 100%; margin: 15px auto;}
}
@media only screen and (max-width: 600px){
    .content{width: 100%;}
    .recipes_content{padding: 20px 10px 20px;}
    .btn-group-add, .btn-group-manage{padding: 0 10px;}
    .image-upload{width: 100%; padding: 0 5px;}
    .action-links{display: flex; flex-direction: column;}
    .action-links .delete{padding: 3px 0 3px 0;}
}
@media only screen and (max-width: 425px) {
    .footer-text{font-size: 1em;}
    .recipes_content{padding: 20px 3px 20px;}
    a{font-size: 0.9rem;}
    th, td{padding: 10px;}
    .footer-text{font-size: 1em;}
    label{margin-left: 10px;}
    .text-input{width: 99%; margin-left: 2px;}
}

@media only screen and (max-width: 768px) {
    .single-recipe-img{width: 100%;}
    .categories{width: 100%; padding: 20px 0 0 20px;}
    .ingredients{width: 100%; padding: 20px 0 0 20px;}
    .content, .comments{width: 90%;}
}
@media only screen and (max-width: 650px) {
    .content, .comments{width: 95%;}
    .comment-header{align-items: flex-start; flex-direction:
column; margin-bottom: 0.7rem;}
    .comment-header span{margin-bottom: 3px;}
}
@media only screen and (max-width: 450px){
    .content, .comments{width: 98%; padding: 10px 10px;}
    .single{padding: 20px 15px;}
    .footer-text{font-size: 1em;}
}
```

Додаток В

Лістинг швидкого меню вебсайту

```
<header>
    <a href="<?php echo BASE_URL ?>" class="logo">
        <h1 class="logo-text"><span class="span-
blue">Flavor</span>Alchemy</h1>
    </a>
    <img src="<?php echo BASE_URL ?>img/menu.png" alt="Меню"
height="40px" class="menu">
    <ul class="nav">
        <li><a href="<?php echo BASE_URL
?>all_recipes.php">Рецепти</a></li>
        <li><a href="<?php echo BASE_URL
?>recipes/my_recipes.php">Мої рецепти</a></li>
        <li><a href="<?php echo BASE_URL ?>about_us.php">Про
нас</a></li>
        <?php if(isset($_SESSION['id'])):?>
            <li>
                <a href="#">
                    <img src="<?php echo BASE_URL ?>img/user.png"
alt="Аватар" height="16px" class="img-right"><?php echo
$_SESSION['login']; ?>
                    <img src="<?php echo BASE_URL ?>img/down.png"
alt="Стрілка вниз" height="20px">
                </a>
                <ul>
                    <?php if ($_SESSION['admin']): ?>
                        <li><a href="<?php echo BASE_URL
?>admin/users/index_users.php">Користувачі</a></li>
                        <li><a href="<?php echo BASE_URL
?>admin/categories/categories.php">Категорії</a></li>
                        <li><a href="<?php echo BASE_URL
?>admin/recipes/published_recipes.php">Рецепти</a></li>
                    <?php endif; ?>
                    <li><a href="<?php echo BASE_URL
?>auth/logout.php" class="logout">Вийти</a></li>
                </ul>
            </li>
        <?php else: ?>
            <li><a href="<?php echo BASE_URL
?>auth/register.php">Реєстрація</a></li>
            <li><a href="<?php echo BASE_URL
?>auth/login.php">Увійти</a></li>
        <?php endif; ?>
    </ul>
</header>
```

Додаток Д

Лістинг скрипту слайдера

```
$('.recipe-body').slick({
    slidesToShow: 5,
    slidesToScroll: 1,
    autoplay: true,
    autoplaySpeed: 2000,
    nextArrow: $('.next'),
    prevArrow: $('.prev'),
    responsive: [
          {
               breakpoint: 2560,
               settings: {
                    slidesToShow: 5,
                    slidesToScroll: 5,
                    infinite: true,
                    dots: true
               }
          },
          {
               breakpoint: 1900,
               settings: {
                    slidesToShow: 4,
                    slidesToScroll: 4
               }
          },
          {
               breakpoint: 1440,
               settings: {
                    slidesToShow: 3,
                    slidesToScroll: 3
               }
          },
          {
               breakpoint: 800,
               settings: {
                    slidesToShow: 2,
                    slidesToScroll: 2
               }
          },
          {
               breakpoint: 500,
               settings: {
                    slidesToShow: 1,
                    slidesToScroll: 1
               }
          }
    ]
});
```

Додаток Е

Лістинг скрипту аутентифікації вебсайту

```
<?php include($_SERVER['DOCUMENT_ROOT'] . '/app/database/db.php');
$table = 'users';
define("BASE_URL", "http://flavoralchemy.fast-page.org/");
$admin_users = SELECT_ALL($table);
$errors = array();
$id = '';
$login = '';
$admin = '';
$email = '';
$password = '';
$repeat_password = '';
if(isset($_POST['register-btn'])){
    $errors = array();
    if(empty($_POST['login'])){
        array_push($errors, 'Введіть логін!');
    }elseif(strlen($_POST['login']) < 6 || strlen($_POST['login'])
> 20) {
        array_push($errors, 'Логін повинен містити від 6 до 20
символів!');
    }
    if(empty($_POST['email'])){
        array_push($errors, 'Введіть адресу електроної пошти!');
    }
    if(empty($_POST['password'])){
        array_push($errors, 'Введіть пароль!');
    }elseif(strlen($_POST['password']) < 4 ||
strlen($_POST['password']) > 20) {
        array_push($errors, 'Пароль повинен містити від 4 до 20
символів!');
    }
    if($_POST['repeat-password'] !== $_POST['password']){
        array_push($errors, 'Паролі не співпадають!');
    }
    $existingUser = SELECT_ONE('users', ['email' =>
$_POST['email']]);
    if (isset($existingUser)){
        array_push($errors, 'Така електрона адреса вже зайнята!');
    }
    $existingUser = SELECT_ONE('users', ['login' =>
$_POST['login']]);
    if (isset($existingUser)){
        array_push($errors, 'Такий логіні вже зайнятий!');
    }
    if(count($errors) === 0){
        unset($_POST['register-btn'], $_POST['repeat-password']);
        $_POST['password'] = password_hash($_POST['password'],
PASSWORD_DEFAULT);
        $_POST['admin'] = 0;
        $user_id = INSERT('users', $_POST);
        $user = SELECT_ONE('users', ['id' => $user_id]);

        $_SESSION['id'] = $user['id'];
```

```
            $_SESSION['login'] = $user['login'];
            $_SESSION['admin'] = $user['admin'];
            $_SESSION['message'] = "Реєстрація пройшла успішно!";
            $_SESSION['type'] = 'success';
            header('Location: '. BASE_URL);
            exit();
        }else{
            $login = $_POST['login'];
            $admin = isset($_POST['admin']) ? 1 : 0;
            $email = $_POST['email'];
            $password = $_POST['password'];
            $repeat_password = $_POST['repeat-password'];
        }
    }
    if (isset($_POST['login-btn'])){
        $errors = array();
        if(empty($_POST['login'])){
            array_push($errors, 'Введіть логін!');
        }
        if(empty($_POST['password'])){
            array_push($errors, 'Введіть пароль!');
        }
        if(count($errors) === 0){
            $user = SELECT_ONE('users', ['login' => $_POST['login']]);
            dd($user['password']);

            dd($_POST['password']);
            if ($user && password_verify($_POST['password'],
    $user['password'])){
                $_SESSION['id'] = $user['id'];
                $_SESSION['login'] = $user['login'];
                $_SESSION['admin'] = $user['admin'];
                $_SESSION['message'] = "Авторизація пройшла успішно!";
                $_SESSION['type'] = 'success';
                if ($_SESSION['admin']){
                    header('Location: '. BASE_URL);
                }else{
                    header('Location: '. BASE_URL);
                }
                exit();
            }else{
                array_push($errors, 'Неправильний логін або пароль!');
            }
        }
        $login = $_POST['login'];
        $password = $_POST['password'];
    }
    ?>
```

Додаток Ж

Лістинг сторінки “Рецепти”

```
<?php define("BASE_URL", "http://flavoralchemy.fast-page.org/");
include("./app/controllers/recipes_controller.php");
setlocale(LC_TIME, 'uk_UA.utf8');
if (!isset($_SESSION['id']) && isset($_GET['saved_recipe_id'])) {
    $_SESSION['message'] = 'Вам потрібно авторизуватися!';
    $_SESSION['type'] = 'error';
    header('Location: ' . BASE_URL . 'auth/login.php');
    exit();
}
$recipeTitle = "Всі рецепти";
setlocale(LC_COLLATE, 'uk_UA.utf8');
$categories_recipes = SELECT_ALL("categories");
usort($categories_recipes, function($a, $b) {return
strcoll($a['name'], $b['name']);});
if (isset($_GET['search']) && !empty($_GET['search'])) {
    $recipeTitle = "Результат пошуку";
    if (isset($_GET['category_id']) &&
!empty($_GET['category_id'])) {
        $categoryId = $_GET['category_id'];
        $searchedRecipes =
SEARCH_RESIPES_ON_GATEGORIES($_GET['search'], $categoryId);
        $recipes = $searchedRecipes;
    } else {
        $searchedRecipes = SEARCH_RESIPES($_GET['search']);
        $recipes = $searchedRecipes;
    }
} elseif (isset($_GET['category_id']) &&
!empty($_GET['category_id'])) {
    $categoryId = $_GET['category_id'];
    $recipesId = RECIPES_BY_CATEGORY($categoryId);
    $recipeIds = array();
    foreach ($recipesId as $recipe) {$recipeIds[] =
$recipe['recipe_id'];}
    $recipes = array();
    foreach ($recipeIds as $recipeId) {
        $selectedRecipe = SELECT_ONE('recipes', array('id' =>
$recipeId, 'published' => 1));
        if ($selectedRecipe) {$recipes[] = $selectedRecipe;}
    }
    $existingcat = SELECT_ONE('categories', ['id' =>
$categoryId]);
    $recipeTitle = "Рецепти в категорії";
    $recipeTitle .= " \"" . $existingcat['name'] . "\"";
} else {$recipes = ALL_RECIPES(1);}
$currentPage = isset($_GET['page']) ? max(1, (int)$_GET['page']) :
1;?>
<!DOCTYPE html>
<html lang="en"><head>
    <meta charset="UTF-8"><meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./css/header.css">
    <link rel="stylesheet" href="./css/style.css">
```

```
    <link rel="stylesheet" href="./css/all_recipes.css">
    <link rel="icon" href="./img/FA.ico" type="image/x-icon">
    <!-- JQuery --><script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.mi
n.js"></script>
    <!-- Slick library --><script
src="./js/slick.min.js"></script>
    <!-- My script --><script src="./js/scripts.js"></script>
    <title>Кулінарні рецепти</title></head>
<body>
<div class="body">
    <?php include("./app/include/header.php"); ?>
    <main>
        <div class="content-wrapper">
            <div class="categories">
                <div class="search"><form action="all_recipes.php"
method="get"><input type="search" name="search" placeholder="Пошук
по назві та вмісту" required><input type="hidden"
name="category_id" value="<?php echo isset($_GET['category_id']) ?
$_GET['category_id'] : ''; ?>"><button
type="submit">Пошук</button></form></div>
                <h2 class="cat-title">Категорії <img src="<?php
echo BASE_URL ?>img/down.png" alt="" height="20px"
class="dowm"></h2>
                <div class="categories-all"><?php
foreach($categories_recipes as $cat): ?><div class="<?php if
($cat['id'] == $categoryId) echo 'active active_categories'; else
echo 'style_cat'; ?>" data-id="<?php echo $cat['id'] ?>"
style="margin-bottom: 10px;"><a href="?category_id=<?php echo
$cat['id']; ?>&page=" style="text-decoration: none;"><div><?php
echo $cat['name'] ?></div></a></div> <?php endforeach; ?></div>
            </div>
            <div class="recipes clearfix">
                <h1 class="recipes-title"><?php echo $recipeTitle;
?></h1>
                <?php $perPage = 5;
                    $recipesCount = count($recipes);
                    $totalPages = ceil($recipesCount / $perPage);
                    $offset = ($currentPage - 1) * $perPage;
                    $recipes = array_slice($recipes, $offset,
$perPage);
                    foreach($recipes as $recipe): ?>
                        <div class="recipe clearfix">
                            <a
href="./recipes/single_recipe.php?id=<?php echo $recipe['id'];
?>"><img src="./img/<?php echo $recipe['image']; ?>" alt=""
class="recipe-img"></a>
                            <div class="recipe-preview">
                                <h2 class="title_recipe"><a
href="./recipes/single_recipe.php?id=<?php echo $recipe['id'];
?>"><?php echo $recipe['title']; ?></a></h2>
                                <div class="text-center-user"><img
src="./img/user.png" alt="" height="16px" class="img-right"> <?php
$userlogin = SELECT_USER($recipe['user_id']); echo
$userlogin[0]['login']; ?>   <img src="./img/calendar.png"
```

```
alt="" height="16px" class="img-right"> <?php echo date('d.m.Y',
strtotime($recipe['create_date'])); ?></div>
                                <p class="previem-text"><span
style="font-weight: bold;">Категорії:</span>
                                    <?php $category_recipe =
SELECT_ALL('recipe_categories', ['recipe_id' => $recipe['id']]);
                                    $category_id = array();
                                    foreach ($category_recipe
as $category) {$category_id[] = $category['category_id'];}
                                    $category_ids = implode(',
', $category_id);
                                    $categoryNames =
CategoryNamesWithIds($category_ids);
                                    if
(!empty($categoryNames)): ?><?php foreach ($categoryNames as
$category): ?><a href="all_recipes.php?category_id=<?php echo
$category['id']; ?>&page=" class="selected-category"><?php echo
$category['name']; ?></a><?php endforeach; ?><?php endif; ?></p>
                                <p class="ingredients previem-
text"><span style="font-weight: bold;">Інгредієнти:</span> <?php
echo mb_substr($recipe['ingredients'], 0, 150, 'UTF-8') . '...';
?></p>
                                <p class="preparation previem-
text"><span style="font-weight: bold;">Приготування:</span> <?php
echo mb_substr($recipe['preparation'], 0, 200, 'UTF-8') . '...';
?></p>
                                <div class="btn-group"><a
href="all_recipes.php?saved_recipe_id=<?php echo $recipe['id'];
?>" class="btn">Додати в улюблені</a><a
href="./recipes/single_recipe.php?id=<?php echo $recipe['id']; ?>"
class="btn">Читати детальніше</a></div>
                            </div></div>
                    <?php endforeach; ?>
                        <?php if ($totalPages > 1): ?><ul
class="pagination"><?php for ($i = 1; $i <= $totalPages; $i++):
?><li>
                                    <?php
if(isset($_GET['category_id'])): ?><a href ="?search=<?php echo
urlencode($_GET['search']); ?>&category_id=<?php echo $categoryId;
?>&page=<?php echo $i; ?>" <?php if ($i == $currentPage) echo
'class="active"'; ?>><?php echo $i; ?></a>
                                    <?php
elseif(isset($_GET['search'])): ?><a href ="?search=<?php echo
urlencode($_GET['search']); ?>&page=<?php echo $i; ?>" <?php if
($i == $currentPage) echo 'class="active"'; ?>><?php echo $i;
?></a>
                                    <?php else: ?><a href
="?page=<?php echo $i; ?>" <?php if ($i == $currentPage) echo
'class="active"'; ?>><?php echo $i; ?></a>
                                    <?php endif; ?>
                            </li><?php endfor; ?></ul><?php endif; ?>
            </div>
        </div>
    </main>
    <?php include("./app/include/footer.php"); ?>
</div></body></html>
```

Додаток З

Лістинг скрипту додавання рецепту на вебсайту

```
<?php include($_SERVER['DOCUMENT_ROOT'] . '/app/database/db.php');
$table = 'recipes';
$recipes = SELECT_ALL($table);
$errors = array();
$id = "";
$title = "";
$ingredients = "";
$preparation = "";
$published = "";
if(isset($_POST['add-recipe'])){
$errors = array();
if(empty($_POST['title'])){
array_push($errors, 'Введіть назву рецепту!'); }
if(empty($_POST['categories'])){
array_push($errors, 'Виберіть категорії!');}
if(empty($_POST['ingredients'])){
array_push($errors, 'Введіть інгридієнти рецепту!');}
if(empty($_POST['preparation'])){
array_push($errors, 'Введіть приготування рецепту!');}
if(isset($_FILES['image']['name'])){
$image_name = time() . '_' . $_FILES['image']['name'];
$destination = '../img/' . $image_name;
$result = move_uploaded_file($_FILES['image']['tmp_name'],
$destination);
if($result){$_POST['image'] = $image_name;
}else{
array_push($errors, "Помилка завантаження фотографії!");}
}else{
array_push($errors, "Потрібно завантаження фотографію!");}
if(count($errors) == 0){
$cat = $_POST['categories'];
unset($_POST['add-recipe'], $_POST['selectCategories'],
$_POST['categories']);
$_POST['user_id'] = $_SESSION['id'];
$_POST['published'] = isset($_POST['published']) ? 1 : 0;
$recipe_id = INSERT($table, $_POST);
$selectedCategories = explode(',', $cat);
foreach ($selectedCategories as $category_id) {
INSERT('recipe_categories', array('recipe_id' => $recipe_id,
'category_id' => $category_id));
}
$_SESSION['message'] = "Рецепт додано успішно!";
$_SESSION['type'] = "success";
header('Location: ./manage_recipes.php');
exit();
}else{
$title = $_POST['title'];
$ingredients = $_POST['ingredients'];
$preparation = $_POST['preparation'];
$published = isset($_POST['published']) ? 1 : 0;    }
}?>
```

Додаток И

Лістинг структури таблиці керування рецептами вебсайту

```
<table>
    <thead>
        <th>№</th>
        <th>Назва</th>
        <th>Дії</th>
        <th>Статус публікування</th>
    </thead>
    <tbody>
        <?php $perPage = 5;
        $recipesCount = count($recipes);
        $totalPages = ceil($recipesCount / $perPage);
        $offset = ($currentPage - 1) * $perPage;
        $recipes = array_slice($recipes, $offset, $perPage);
        foreach ($recipes as $key => $recipe): ?>
            <tr>
                <td><?php echo $key+1; ?></td>
                <td><a href="./single_recipe.php?id=<?php echo
$recipe['id']; ?>"><?php echo $recipe['title']; ?></a></td>
                <td style="white-space: nowrap;" class="icon"><div
class="action-links">
                    <a href="edit_recipe.php?id=<?php echo
$recipe['id']; ?>" class="edit">
                        <img src="../img/edit.png"
alt="Редагувати" height="35px" title="Редагувати">
                    </a>
                    <a href="edit_recipe.php?delete_id=<?php echo
$recipe['id']; ?>" class="delete">
                        <img src="../img/delete.png"
alt="Видалити" height="35px" title="Видалити">
                    </a>
                </div></td>
                <td class="icon">
                    <?php if ($recipe['published']): ?>
                        <div>
                            <img src="../img/accept.png"
alt="Опублікований" height="35px" title="Опублікований">
                        </div>
                    <?php else: ?>
                        <div>
                            <0img src="../img/cross.png"
alt="Неопублікований" height="35px" title="Неопублікований">
                        </div>
                    <?php endif; ?>
                </td>
            </tr>
        <?php endforeach; ?>
    </tbody>
</table>
```