МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ

Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

МАРКЕВИЧ КАТЕРИНА ОЛЕКСАНДРІВНА

РОЗРОБКА ВЕБСАЙТУ ДЛЯ ПОШУКУ РЕЦЕПТІВ ЗА ІНГРЕДІЄНТАМИ З ІНТЕРАКТИВНИМ ІНТЕРФЕЙСОМ ЗАСОБАМИ ВЕБТЕХНОЛОГІЙ

Спеціальність: 122 «Комп'ютерні науки»
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технологій
Кваліфікаційна робота на здобуття освітнього ступеня бакалавр

Науковий керівник:
ГЛИНЧУК ЛЮДМИЛА ЯРОСЛАВІВНА,
кандидат фізико-математичних наук, доцент
кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №
Засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2025 р.
Завідувач кафедри
(_____) Гришанович Т. О.

Луцьк 2025

ЗМІСТ

ВСТУП

У повсякденному житті ми дедалі частіше звертаємося до онлайн-платформ, які допомагають покращити якість нашого життя. Особливо це стосується підтримки здорового способу життя, що передбачає правильне харчування. Актуальність теми полягає в тому, що серед великої кількості кулінарних ресурсів лише небагато пропонують можливість пошуку рецептів за інгредієнтами, що суттєво полегшує вибір страв, коли не вистачає ідей для приготування. Розробка зручного вебсайту з такою функцією задовольняє потреби користувачів у простоті, швидкості та ефективності пошуку кулінарних рецептів.

Створення інтуїтивно зрозумілого вебсайту з фокусом на пошук рецептів за інгредієнтами та відповідними фільтрами дозволить задовольнити потреби широкої аудиторії користувачів, включаючи людей з дієтичними обмеженнями, початківців у кулінарії або тих, хто хоче зекономити продукти. Розробка такого програмного забезпечення сприяє подальшому розвитку вебтехнологій та їх застосуванню в повсякденному житті.

Мета дослідження – розробка вебсайту, для пошуку рецептів за інгредієнтами, що забезпечує зручний користувацький досвід, інтуїтивний інтерфейс

Для реалізації поставленої мети виділено такі **завдання:**

- здійснити аналіз існуючих програмних продуктів;
- дослідити архітектуру вебдодатків, орієнтованих на роботу з базами даних та фільтрацією інформації;
- реалізувати інтерфейс користувача з підтримкою пошуку та фільтрації рецептів;
- реалізувати базу даних рецептів із можливістю пошуку за інгредієнтами;
- визначити середовище розробки та інструментальні засоби;
- протестувати функціональність вебсайту.

Об’єкт дослідження – веборієнтовані програмні системи для пошуку інформації за критеріями користувача.

Предмет дослідження – архітектура, функціональність та реалізація вебсайту з пошуку рецептів за інгредієнтами.

Апробація результату роботи. Результати дослідження опубліковані у вигляді тез на тему «Розробка вебінтерфейсу для персоналізованого пошуку кулінарних рецептів», які увійшли до збірника матеріалів II Міжнародної науково-практичної конференції «Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем». Конференція проходила 9-11 червня 2025 року на базі практик табору «Гарт» Волинського національного університету імені Лесі Українки (Шацький район, с. Світязь).

РОЗДІЛ 1. АНАЛІЗ ВЕБТЕХНОЛОГІЙ ТА БАЗ ДАНИХ ДЛЯ РОЗРОБКИ ІНТЕРАКТИВНИХ ВЕБСАЙТІВ

1.1 Актуальність та огляд сучасних тенденцій у веброзробці

Сучасна веброзробка переживає період швидких змін, де нові технології, інструменти та підходи постійно трансформують процес створення вебсайтів і вебзастосунків. Розвиток інтернету, змінювані вимоги до користувацького досвіду та постійне вдосконалення інфраструктури надають нові можливості для розробників.

Один з основних напрямків, що визначають сучасні тенденції веброзробки, – це підвищення вимог до швидкості завантаження сайтів та їх взаємодії з користувачем. Вебсайти повинні бути не тільки функціональними, але й інтуїтивно зрозумілими, швидкими та адаптивними. У зв'язку з цим популяризуються такі технології, як React, Next.js і Vue.js, що дозволяють створювати інтерактивні та швидкі інтерфейси, а також застосовувати серверний рендеринг для підвищення швидкості завантаження сторінок. Зокрема, Next.js є одним з лідерів у реалізації серверного рендерингу на фронтенді, що дає можливість пришвидшити перший рендер сайту (сайт завантажується значно швидше, що в свою чергу покращує користувацький досвід). [1]

Однією з основних тенденцій є також інтеграція фронтенду та бекенду з використанням API та баз даних. Вебсайти вже не можуть працювати без підтримки великих обсягів даних, зокрема для динамічних вебсайтів, таких як онлайн-магазини чи платформи для пошуку рецептів, де важливо зберігати та обробляти великі набори інформації про продукти та інгредієнти. У цьому контексті MongoDB стає популярним рішенням завдяки своїй гнучкості та здатності ефективно працювати з неструктурованими даними. Вебзастосунки, побудовані з використанням MongoDB, дозволяють зберігати рецепти, інгредієнти, інформацію про користувачів та їх взаємодії без необхідності

жорсткої структури даних. Цей підхід особливо корисний для проектів, що передбачають велике та швидко змінюване коло даних. [2]

Ще однією важливою тенденцією є постійний розвиток мобільної адаптивності та респонсивного дизайну. Оскільки більшість користувачів доступаються до Інтернету через мобільні пристрої, важливо, щоб вебсайти були адаптовані під різні розміри екрану та операційні системи. Технології CSS3 і HTML5 дозволяють створювати адаптивні інтерфейси, а також використовувати нові можливості мультимедіа та анімації для покращення взаємодії з користувачем.

Завдяки таким технологіям, як JavaScript та новим фреймворкам для фронтенду, включаючи React і Vue.js, веброзробка перетворюється на високопродуктивний процес, який дозволяє створювати швидкі, інтерактивні та привабливі сайти.

Таким чином, веброзробка не лише адаптується до нових вимог користувачів, але й постійно вдосконалюється завдяки інноваціям у технологіях, що дозволяють створювати більш ефективні та інтерактивні вебзастосунки.

1.2 Аналіз технологій фронтенду та бекенду для інтерактивних вебзастосунків

У сучасній веброзробці інтерактивні вебзастосунки створюються за допомогою різних технологій, які забезпечують ефективну взаємодію користувача з інтерфейсом, швидке завантаження сторінок та обробку даних. Вебзастосунки поділяються на два основні рівні: Front-end (клієнтська частина) та Back-end (серверна частина), кожен із яких виконує окремі завдання.

1.2.1. Front-end технології

Front-end – це публічна частина вебзастосунку, з якою безпосередньо взаємодіє користувач. Вона відповідає за візуальне представлення інтерфейсу та

реалізацію функціональних можливостей на стороні клієнта. Саме frontend визначає, як виглядає вебсторінка та як вона реагує на дії користувача – введення даних, натискання кнопок, навігацію тощо. [2]

Основні технології, що використовуються для розробки фронтенду:

- HTML (HyperText Markup Language) – основа веб-сторінок, що визначає їх структуру;
- CSS (Cascading Style Sheets) – відповідає за оформлення та візуальне представлення елементів сторінки;
- JavaScript (JS) – забезпечує динамічну взаємодію користувача з вебсторінкою.

Вебфреймворки є важливими інструментами, оскільки вони спрощують процес створення додатків і дозволяють зосередитись на специфічних аспектах проекту, замість того щоб писати повторюваний код. Серед найбільш популярних фреймворків можна виділити React.js, Angular.js, Vue.js, Django та Express.js. React.js, розроблений компанією Facebook, дозволяє створювати компоненти вебінтерфейсу, використовуючи віртуальний DOM для ефективного оновлення сторінок. Angular.js, у свою чергу, пропонує повноцінний набір інструментів для створення складних односторінкових додатків (SPA), включаючи двосторонній зв'язок даних і підтримку AJAX. Vue.js поєднує в собі переваги React та Angular, пропонуючи простоту і гнучкість для інтеграції в існуючі проекти. [3]

Ці технології спрощують розробку складних, інтерактивних вебінтерфейсів та покращують користувацький досвід.

1.2.2. Back-end технології

Back-end – це серверна частина вебзастосунку, яка прихована від користувача та забезпечує реалізацію бізнес-логіки. Вона відповідає за обробку запитів, збереження та отримання даних, управління користувачами, автентифікацію, а також взаємодію з зовнішніми сервісами та API. Розробка

бекенду зазвичай здійснюється з використанням фреймворків, створених на основі мов програмування Python, Java, PHP та інших. Важливою складовою цієї частини є системи управління базами даних, зокрема PostgreSQL, MySQL або MongoDB, які забезпечують зберігання та ефективне опрацювання інформації. [2]

Популярні Back-end технології охоплюють різноманітні підходи до розробки серверної частини вебзастосунків, зокрема Node.js, Next.js, Express.js та Django. Кожна з цих технологій має свої особливості та переваги, що робить їх ідеальними для різних типів проектів.

Node.js – це середовище виконання JavaScript на сервері, створене на основі рушія V8 від Google, яке дозволяє розробляти серверні додатки з використанням JavaScript. Воно підтримує асинхронну, подієво-орієнтовану модель програмування, що забезпечує ефективну обробку великої кількості одночасних запитів без блокування виконання. Завдяки великій екосистемі модулів через менеджер пакетів npm та сумісності з популярними фреймворками, такими як Express.js, Node.js широко застосовується для створення веб-серверів, мікросервісів, інструментів командного рядка й навіть настільних додатків, що робить його однією з найпопулярніших платформ для серверної розробки. [4]

Next.JS дозволяє створювати потужні та масштабовані сайти з використанням статичного рендерингу (SSG) для статичних сторінок, таких як сторінки про компанію або контактна інформація, та серверного рендерингу (SSR) для більш динамічних розділів – новин чи блогів. Використання таких підходів значно підвищує продуктивність застосунку, оскільки зменшується час завантаження сторінок, а також поліпшується SEO. Next.js є ідеальним вибором для вебсайтів, що потребують високої швидкості завантаження, таких як блоги, інтернет-магазини або корпоративні портали. [5]

Express.js – це легкий і гнучкий фреймворк для створення вебзастосунків на базі Node.js, який вирізняється простотою та ефективністю. Його мінімалістичний підхід дозволяє розробникам будувати додатки з нуля,

використовуючи лише необхідні компоненти. Фреймворк легко розширюється завдяки великій кількості сторонніх модулів. Завдяки цим перевагам Express.js став одним із найпоширеніших рішень для створення серверної частини вебпроєктів. [6]

Django – фреймворк з відкритим кодом для мови Python, який реалізовано за архітектурною моделлю MVT (Model-View-Template). Завдяки модульному підходу він спрощує розробку та підтримку вебзастосунків, надаючи багато готових рішень, таких як система автентифікації, адміністративна панель, шаблонізатор і потужна ORM. [7]

1.2.3. Використання хмарних сервісів у вебзастосунках

У сучасному світі хмарні технології стають невід'ємною частиною розробки вебдодатків. Інтеграція хмарних сервісів дозволяє розробникам створювати більш гнучкі, масштабовані та безпечні додатки. Хмарні сервіси – це програмні рішення та інфраструктура, що надаються через Інтернет. Вони дозволяють зберігати дані, виконувати обчислення, розгортати додатки та використовувати різні інструменти без необхідності управляти фізичними серверами. Найпопулярніші постачальники хмарних сервісів включають Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP). [8]

Однією з основних функцій хмарних сервісів є масштабоване зберігання даних. Такі сервіси, як Amazon S3, Google Cloud Storage і Azure Blob Storage, дозволяють зберігати великі обсяги структурованої та неструктурованої інформації з високим рівнем доступності, надійності та безпеки. Це особливо корисно для збереження зображень, відео, резервних копій або статичних ресурсів вебзастосунків. [8]

Крім того, хмарні провайдери надають обчислювальні ресурси у вигляді віртуальних машин які дозволяють виконувати складні розрахунки, запускати сервери додатків чи здійснювати обробку великих обсягів даних. У поєднанні з керованими базами даних, такими як Amazon RDS, Google Cloud SQL чи Azure

SQL Database, це створює повноцінне середовище для бекенд-логіки застосунків. [8]

Сучасні вебпроєкти також дедалі частіше використовують серверлес обчислення, що дозволяє запускати окремі функції у відповідь на події без необхідності обслуговування серверів. Сервіси на кшталт AWS Lambda, Google Cloud Functions та Azure Functions забезпечують автоматичне масштабування і значне зниження витрат. [8]

Кожен тип хмарного сховища має свої особливості, переваги та недоліки, які залежать від конкретних потреб користувача.

До переваг інтеграції хмарних сервісів належать:

- масштабованість – можливість оперативно змінювати обсяги ресурсів відповідно до навантаження;
- гнучкість – широкий набір інструментів для реалізації як простих, так і складних архітектур;
- зниження витрат – оплата лише за використані ресурси;
- висока доступність і надійність – завдяки глобальним дата-центрам і автоматичному резервному копіюванню;
- безпека – вбудовані механізми шифрування, автентифікації, моніторингу;
- прискорення розробки – завдяки використанню готових інструментів і сервісів (CI/CD, DevOps, API-менеджмент). [8]

Таким чином, хмарні сервіси забезпечують фундамент для побудови сучасних вебзастосунків, дозволяючи створювати надійні, безпечні та ефективні рішення, які відповідають вимогам динамічного цифрового середовища. Їх застосування є особливо доцільним у проєктах, що потребують високої доступності, обробки великих обсягів даних або швидкого масштабування. Успішне впровадження хмарних технологій значно підвищує конкурентоспроможність вебпродукту та скорочує час виходу на ринок.

1.3 Використання баз даних у вебпроєктах, їх особливості зберігання та обробки даних

У сучасних вебзастосунках бази даних відіграють ключову роль у зберіганні, обробці та управлінні інформацією. Вони забезпечують збереження користувацьких даних, історії транзакцій, налаштувань тощо. Залежно від потреб застосунку використовують реляційні або нереляційні бази даних.

Реляційна база даних (SQL) – це база, у якій дані організовані в таблиці з чіткою структурою та взаємозв'язками між ними. Усі записи поділені на рядки та стовпці, де кожне поле має визначений тип. Такі бази відзначаються високою надійністю, гарантують цілісність даних завдяки відповідності принципам ACID (атомарність, непротирічність, ізольованість, довговічність), і є зручними для роботи зі структурованою інформацією, яка рідко змінюється. До таких баз належать популярні системи управління базами даних (СУБД) як MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database та SQLite. [9]

Нереляційна база даних (NoSQL) – це база, яка зберігає дані без суворих зв'язків і чіткої структури. Замість таблиць вона використовує документи, графи або ключ-значення, що дозволяє зберігати будь-які типи даних – від текстів до зображень. Завдяки гнучкій схемі та розподіленій архітектурі такі бази добре підходять для масштабування, швидкої розробки та роботи з великими обсягами неструктурованої інформації. Такі бази даних особливо підходять для додатків, що обробляють великі обсяги інформації, наприклад, для аналізу соціальних мереж або зберігання мультимедійного контенту. До NoSQL належать такі системи, як MongoDB, Cassandra, DynamoDB, Redis, CouchDB. [9]

1.4 Архітектура сучасних вебзастосунків

Сучасна архітектура вебзастосунків базується на клієнт-серверній моделі, яка передбачає розподіл функціональності між клієнтською та частинами. У цій моделі клієнт (браузер або мобільний застосунок) надсилає запити до сервера,

який обробляє їх, взаємодіє з базою даних і повертає результат – зазвичай у форматі JSON або HTML. Такий підхід дозволяє створювати масштабовані, надійні та інтерактивні системи, де кожен компонент відповідає за чітко визначені завдання.

Одним із ключових досягнень сучасної веб-розробки є Single Page Application (SPA) – тип вебзастосунків, у яких взаємодія з користувачем відбувається на одній HTML-сторінці без повторного завантаження. У SPA вміст динамічно оновлюється за допомогою JavaScript та AJAX-запитів, що дозволяє швидко відображати змінені дані та забезпечує високий рівень інтерактивності. Завдяки цьому значно покращується користувацький досвід (UX) – сторінка працює швидко, реагує миттєво та особливо ефективна на мобільних пристроях. Також SPA надає можливість створення кросплатформних застосунків, що є важливою перевагою в сучасних умовах. [10]

Для реалізації такої архітектури вебзастосунки активно використовують API (Application Programming Interface) – документований інтерфейс, який забезпечує взаємодію між клієнтом та сервером. Використання API є найбільш поширеним підходом до організації клієнт-серверної архітектури. Завдяки API реалізується ефективна взаємодію користувача із серверною частиною. Далі розглянемо детальніше, як працює API. [11]

Існує кілька основних видів API, які використовуються у веброзробці для організації взаємодії між клієнтом і сервером. Найпоширенішими REST API (Representational State Transfer) та GraphQL. REST API – це стиль архітектури для створення вебсервісів, який використовує стандартні HTTP-методи (GET, POST, PUT, DELETE) для доступу до ресурсів через URL. [12] GraphQL – мова запитів і маніпулювання даними для API , що дає змогу клієнту точно вказати, які дані йому потрібні від сервера, зменшуючи обсяг переданої інформації. [13]

1.5 Аналіз існуючих рішень та конкурентних платформ для пошуку рецептів

Здійснимо огляд та аналіз вебплатформ з рецептами, які використовують пошук рецептів за інгредієнтами. Порівняємо їхні функціональні можливості, особливості користувацького досвіду та підхід до взаємодії з аудиторією. Вивчення цих рішень дозволить оцінити їх переваги та недоліки, що стане основою для розробки власної конкурентоспроможної платформи. Важливими практичними характеристиками які ми будемо розглядати є наявність пошуку за інгредієнтами, пошуку без певних інгредієнтів, наявність фільтрів та сортування, зручність інтерфейсу та сучасність дизайну.

1.5.1 Кросплатформне програмне забезпечення «Cookpad»

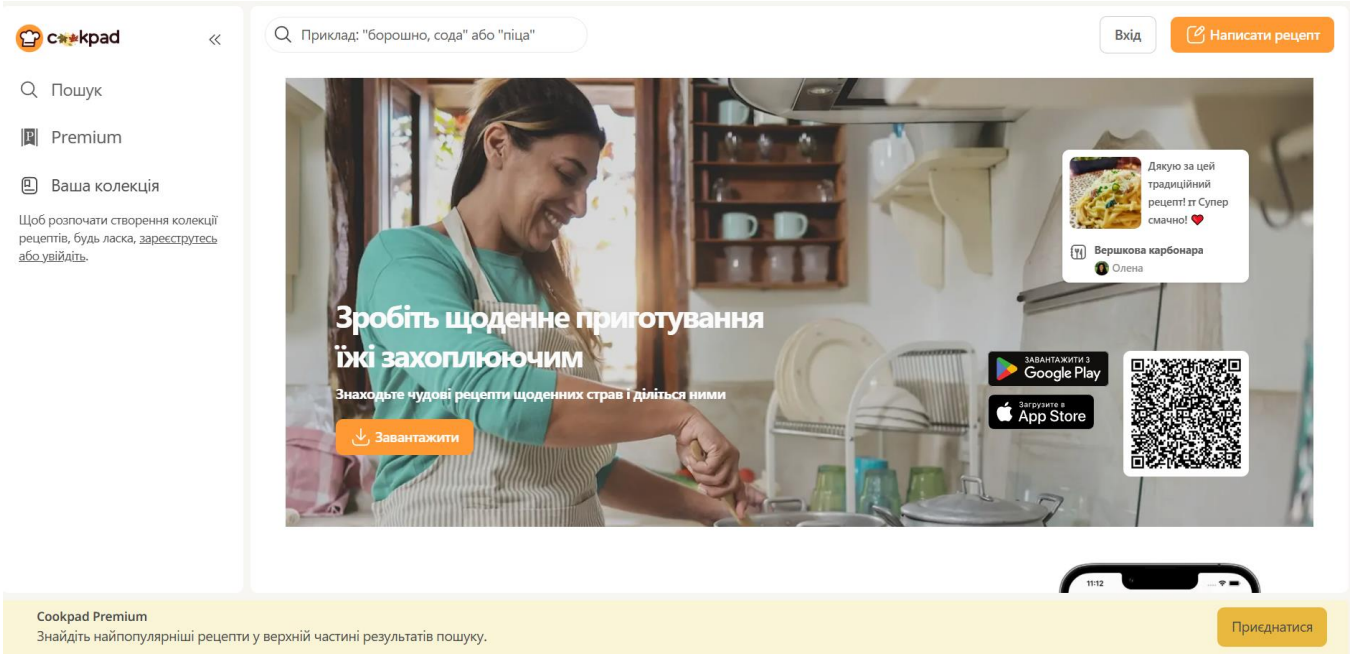


Рисунок 1.1 – Вебсайт «Cookpad»

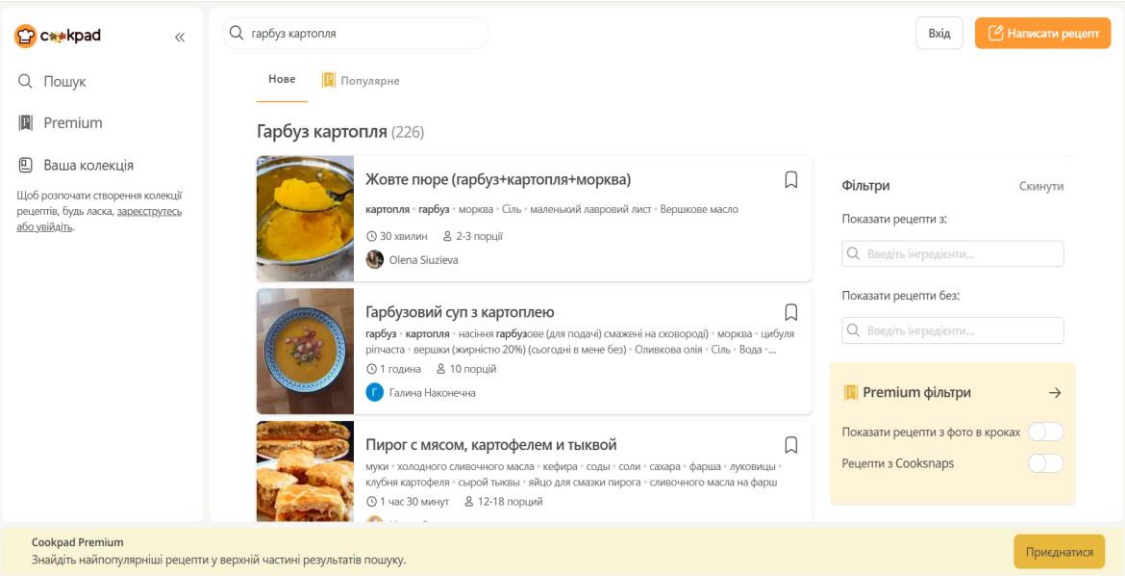


Рисунок 1.2 – Пошук за інгредієнтами по сайті «Cookpad»

Розробник: Cookpad Inc.

Архітектура: desktop application,web application.

Функції: пошук рецептів за назвою або інгредієнтами, пошук рецептів без певних інгредієнтів, сортування рецептів за популярністю, рецепти з фото в кроках. Користувач має можливість створення профілю та взаємодії з профілями інших користувачів, залишати коментарі, реакції та фото до рецепту, зберігати рецепту в свою «колекцію», можна додати свій рецепт на сайт при наявності профілю. Рецепти мають інформацію про кількість порцій, час приготування, список інгредієнтів, можливість роздрукувати рецепт, покрокову інструкцію з приготування.

Джерело: https://cookpad.com/ua/homepage

Результати тестування:

- виконання пошуку за ключовими словами, введення одного або декількох інгредієнтів працює коректно;
- введення небажаних інгредієнтів у фільтр виключає рецепти, що містять зазначені інгредієнти;
- реєстрація працює без збоїв. Користувачі можуть редагувати профілі, переглядати профілі інших і взаємодіяти через підписки;

- функція додавання рецепту через форму із заповненням всіх обов'язкових полів працює коректно.

Переваги вебсайту:

- сучасний, інтуїтивний та зрозумілий інтерфейс;
- можливість роздрукувати рецепт;
- можливість залишати коментарі, ставити оцінки, підписуватися на інших користувачів, що робить платформу привабливою для активного спілкування;
- наявність фільтру що виключає рецепти з певним інгредієнтом, корисно для алергіків;
- наявність фото у покрокових інструкціях приготування.

Недоліки вебсайту:

- функції сортування рецептів за популярністю та рецепти з фото в кроках доступні лише для користувачів «Premium»;
- не має фільтрів для пошуку за часом приготування, складністю, дієтичними опціями;
- рецепти не місять нутріціологічної інформації.

1.5.2 Вебсайт «rozdil»

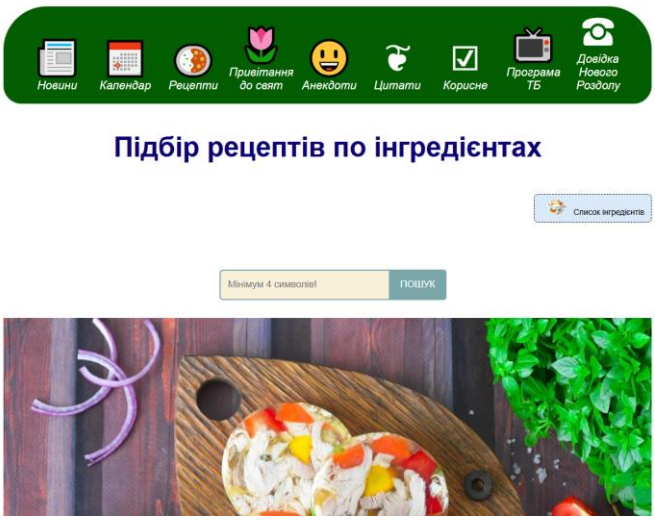


Рисунок 1.3 – Вебсайт «rozdil»

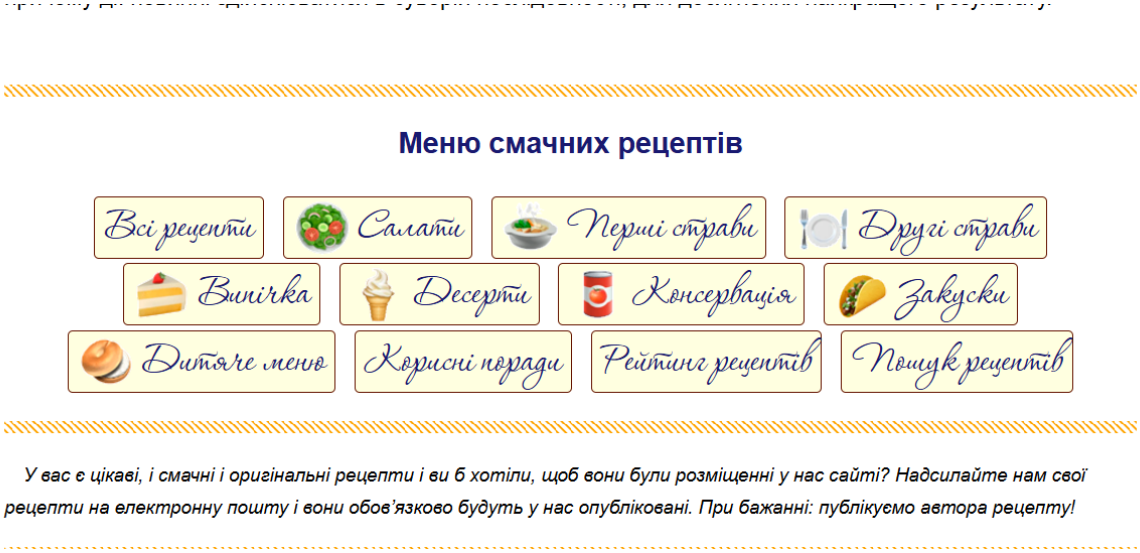


Рисунок 1.4 – Фільтр рецептом за типом страви

Розробник: Романа Зотенко.

Архітектура: web application.

Функції: пошук рецептів за назвою або інгредієнтами, сортування рецептів за новизною і рейтингом. Користувач має можливість залишити реакцію до рецепту. Рецепти мають список інгредієнтів та покрокову інструкцію з приготування. Фільтр рецепту за типом страви. Наявний пошук рецепту при виборі інгредієнтів зі списку.

Джерело: https://rozdil.lviv.ua/statti/search/

Результати тестування:

- виконання пошуку за ключовими словами, введення одного або декількох інгредієнтів працює коректно;
- виконання пошуку за інгредієнтами зі списку працює коректно;
- фільтр та сортування працюють.

Переваги вебсайту:

- інтуїтивний та зрозумілий інтерфейс;
- фільтр за типом страви.

Недоліки вебсайту:

- пошук рецепту при виборі інгредієнтів зі спику займає велику частину сторінку і має обмеження у виборі 6 інгредієнтів;
- не має фільтрів для пошуку за часом приготування, складністю, дієтичними опціями;
- рецепти не місять нутріціологічної інформації, не зазначено кількість порцій.

1.5.3 Кросплатформне програмне забезпечення «Cookorama.net»

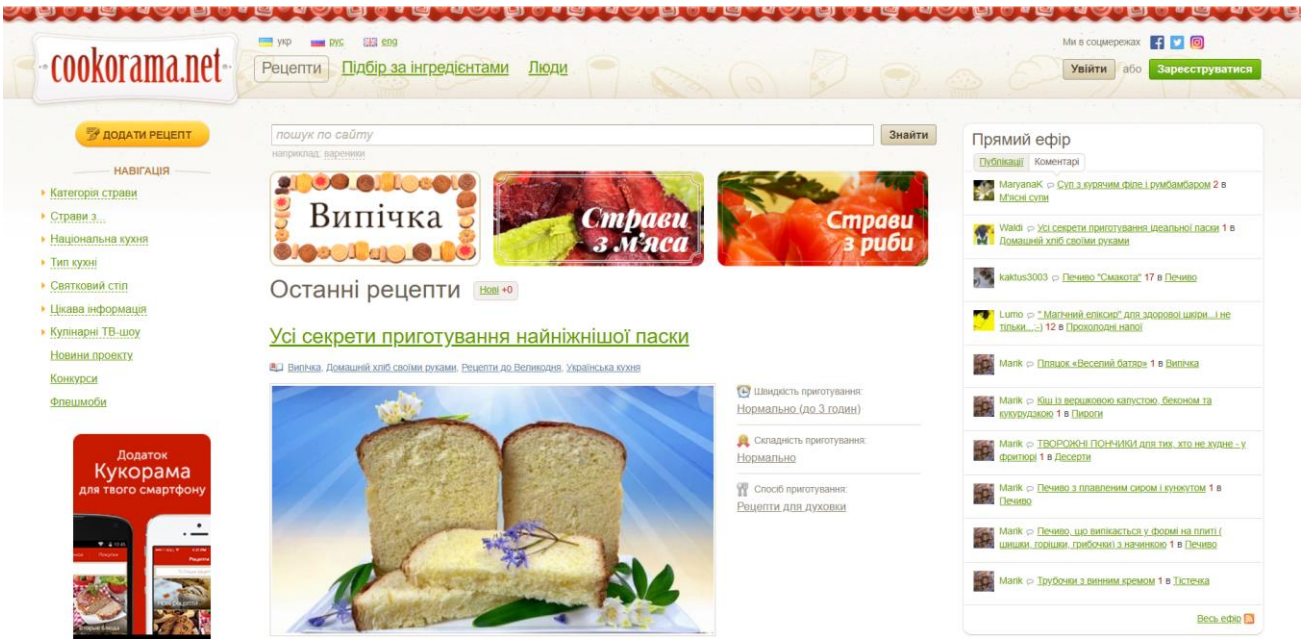


Рисунок 1.5 – Вебсайт «Cookorama.net»

Розробник: вебстудія stfalcon.com.

Архітектура: desktop application,web application.

Функції: пошук рецептів за назвою або інгредієнтами. Користувач має можливість створення профілю та взаємодії з профілями інших користувачів, залишати коментарі, залишати реакції, зберігати рецепт «обране», можна додати свій рецепт на сайт при наявності профілю. Рецепти мають інформацію про

кількість порцій, час приготування, список інгредієнтів, рівень складності та покрокову інструкцію з приготування.

Джерело: https://cookorama.net/uk/

Результати тестування:

- виконання пошуку за ключовими словами, введення одного або декількох інгредієнтів працює коректно;
- користувачі можуть редагувати профілі, переглядати профілі інших і взаємодіяти через підписки;
- функція додавання рецепту через форму із заповненням всіх обов'язкових полів працює коректно.

Переваги вебсайту:

- можливість залишати коментарі, підписуватися на інших користувачів, що добре для активного спілкування.
- Недоліки вебсайту:
- нагромаджений і не зручний інтерфейс, що робить платформу непривабливою;
- рецепти не місять нутріціологічної інформації.

1.5.4 Висновки огляду та аналізу аналогічних розробок

Аналіз аналогічних вебсайтів для пошуку рецептів дозволив визначити ключові функціональні можливості. Зокрема, більшість існуючих платформ пропонують:

- реєстрація та авторизація користувачів;
- можливість залишати оцінки та писати відгуки про рецепти;
- можливість користувачам додавати власні рецепти після реєстрації;
- можливість зберігати улюблені рецепти для швидкого доступу;
- організацію за типом страви;
- детальні сторінки рецепту з інгредієнтами, інструкціями та фотографіями;

- пошук за інгредієнтами;
- можливість виключати певні інгредієнти при пошуку.

Оскільки зазначені функції становлять базовий набір для сучасних платформ рецептів, вони будуть присутні у розробці вебсайту.

Для покращення та розширення функціоналу додамо наступні функції:

- розширимо функціонал пошуку додавши фільтрацію рецептів за часом приготування, складністю, калорійністю, сортування результатів пошуку за популярністю;
- дієтичні опції (веганство, вегетеріанство, безлактозні та безглютенові страви) у фільтри;
- інформацію про калорійність, харчову цінність, дієтичні опції і кухню до детальної сторінки рецепту.

Розробка цих нововведень допоможе виділитися серед аналогів, залучити ширшу аудиторію та покращити користувацький досвід.

РОЗДІЛ 2. РОЗРОБКА ІНТЕРАКТИВНОГО ВЕБСАЙТУ ДЛЯ ПОШУКУ РЕЦЕПТІВ ЗА ІНГРЕДІЄНТАМИ З ВИКОРИСТАННЯМ ВЕБТЕХНОЛОГІЙ

2.1 Постановка задачі, призначення та вимоги до розробки інтерактивного вебсайту

Завданням роботи є розробка вебзастосунку, який дозволяє пошук рецептів страв за інгредієнтами чи назвою, використовувати фільтри за категоріями (тип страви, дієта, калорійність), а також переглядати та ставити оцінки, залишати відгуки, додавати власні рецепти.

Розроблюваний вебзастосунок призначений для широкого кола користувачів, які прагнуть швидко знаходити рецепти відповідно до своїх потреб. Сервіс спростить процес планування страв, пошуку кулінарних ідей та організації продуктів.

Програмний продукт також може бути корисним для людей із дієтичними обмеженнями (вегани, вегетаріанці, користувачі з алергіями), для планування харчування або для кулінарних блогерів, які хочуть ділитися рецептами.

Функціональні вимоги:

- авторизація та реєстрація профілю для користувача за допомогою email та мати змогу редагування профілю;
- можливість для користувача публікувати власні рецепти, додавати в обране, оцінювати і коментувати інші рецепти;
- перегляд профілів інших користувачів, підписка і перегляд доданих рецептів користувача;
- пошук рецептів за назвою, одним або декількома інгредієнтами та виключенням рецептів з небажаними інгредієнтами;
- фільтрація результатів за типом страви, часом приготування, рівнем складності, дієтичними опціями та калорійністю;
- сортування результатів за рейтингом, часом приготування;

- рецепти повинні мати детальну сторінку. Яка міститиме фото, список інгредієнтів, покрокову інструкцію, інформацію про кількість порцій, тип кухні, дієтичні опції, час приготування, покрокову інструкцію приготування, нутріціологічну інформацію.

Вимоги до інтерфейсу користувача:

- адаптивність під пристрої різних розмірів;
- інтуїтивно зрозумілий дизайн з врахуванням сучасних принципів UX/UI;
- кросбраузерна сумісність і відповідність сучасним стандартам.

2.2 Загальний опис проєкту

Проєкт «EasyIngrecipes» – це вебплатформа для пошуку кулінарних рецептів за наявними інгредієнтами, яка поєднує функціональність фільтрації, сортування, можливості ділитися свої рецептами та взаємодії користувачів. Платформа орієнтована на зручність, швидкість і практичність, щоб зробити процес приготування їжі простішим і доступнішим для кожного.

Цільовою аудиторією такої платформи є користувачі, які бажають оптимізувати процес приготування їжі, використовуючи лише ті продукти, що є під рукою. Платформа також орієнтована на людей із певними дієтичними потребами, такими як веганство, безглютенове або низьковуглеводне харчування, що дозволяє їм знаходити рецепти відповідно до власних обмежень чи вподобань. Завдяки цьому аудиторія сайту охоплює широкий спектр користувачів із різними кулінарними інтересами та цілями.

Головною ідеєю вебсайту є створення простого та зручного інструменту для пошуку рецептів, який мінімізує харчові відходи, економить час і враховує індивідуальні харчові вподобання користувача (наприклад, вегетаріанські, безглютенові рецепти тощо). Сервіс також дозволяє зберігати улюблені рецепти, ділитися ними з іншими користувачами.

Архітектура системи побудована за принципом клієнт-серверної моделі, що забезпечує розподіл відповідальностей між інтерфейсом користувача (Frontend), логікою обробки запитів (Backend) та сховищем даних (базою даних). Такий підхід дає змогу забезпечити масштабованість, зручність у супроводі, повторне використання компонентів та чітке розмежування обов'язків.

Система поділяється на кілька основних логічних компонентів:

- клієнтська частина – відповідає за взаємодію з користувачем, відображення інформації, прийом введених даних та передачу запитів до сервера. Інтерфейс адаптовано до роботи на різних пристроях (десктоп, планшет, смартфон), з урахуванням сучасних принципів UX/UI;
- серверна частина – обробляє запити, виконує бізнес-логіку, перевірку даних, роботу з рецептами, пошук за інгредієнтами, тощо. Вона також забезпечує взаємодію між клієнтом і базою даних за допомогою чітко визначених точок доступу;
- база даних – зберігає інформацію про рецепти, користувачів, інгредієнти, відгуки, рейтинги тощо. Доступ до даних реалізовано через інтерфейс, що ізолює бізнес-логіку від фізичної структури сховища.

Взаємодія між компонентами здійснюватиметься за допомогою стандартних HTTP-запитів у форматі, що дозволяє зручно обмінюватися даними між клієнтом і сервером. Усі частини системи можуть бути розгорнуті окремо, що дозволяє масштабувати проєкт відповідно до навантаження.

Центральною ідеєю архітектури є модульність – кожен функціональний блок реалізований як окрема частина системи, що дозволяє легко додавати нові можливості, тестувати і оновлювати систему без впливу на інші компоненти. Такий підхід також полегшує розподілену розробку та майбутню підтримку платформи.

Для кращого розуміння архітектури програмної розробки продемонстровані діаграма варіантів використання (Use Case Diagram) рисунок 2.1 та діаграма послідовності (Sequence Diagram) рисунок 2.2.

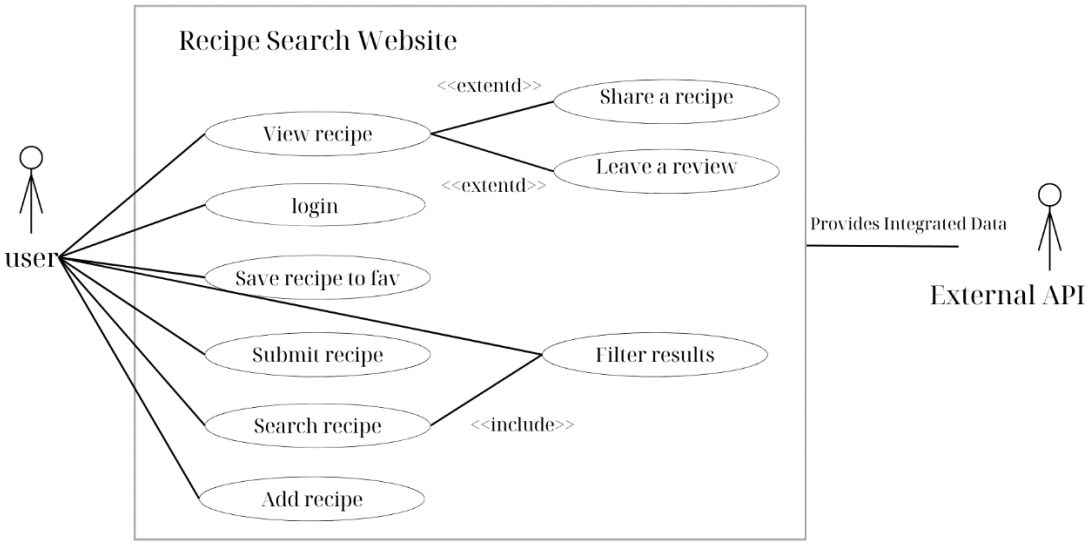


Рисунок 2.1 – Діаграма Use Case Diagram

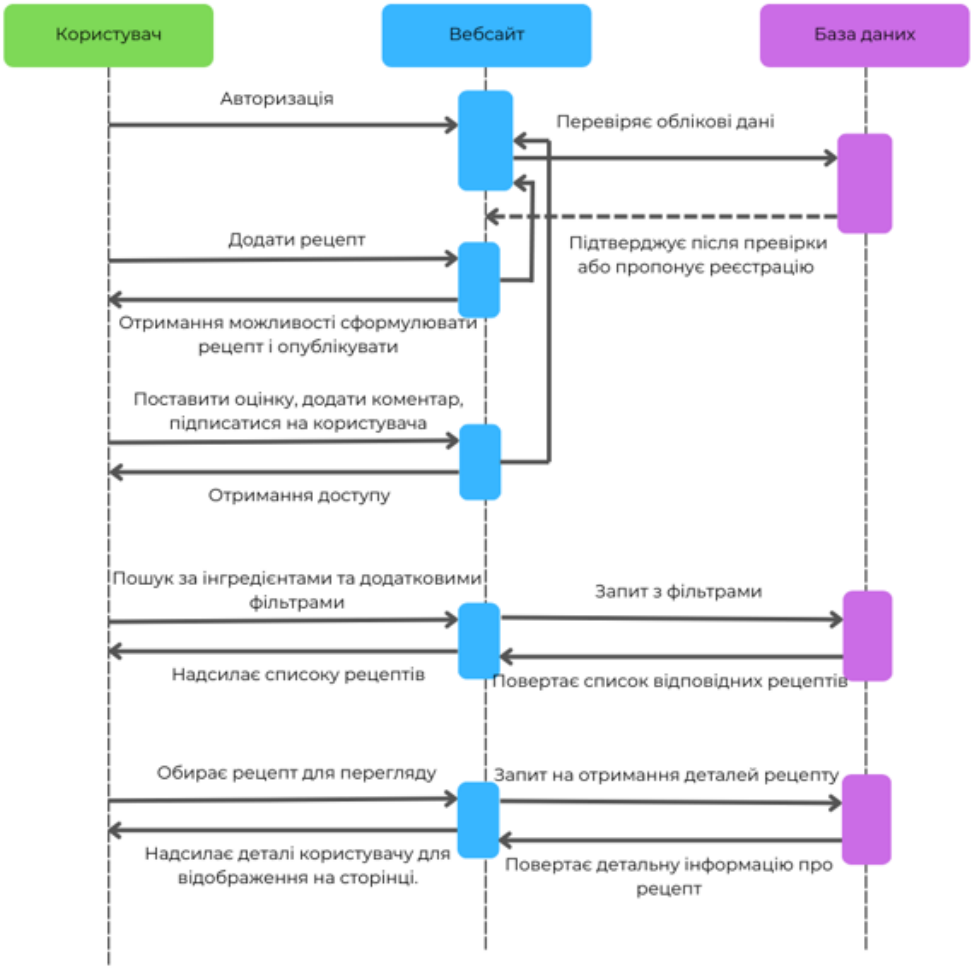


Рисунок 2.2 – Діаграма Sequence Diagram

Для зберігання та обробки даних у системі передбачено використання структурованого сховища, яке забезпечує ефективне збереження, пошук та оновлення інформації. База даних проєкту містить кілька ключових сутностей, кожна з яких відповідає певному типу даних, необхідному для функціонування платформи.

Основними інформаційними об’єктами є:

- користувачі – зберігається інформація про зареєстрованих осіб, включно з їх унікальними ідентифікаторами, іменами, даними профілю;
- рецепти – включають назву, короткий опис, перелік інгредієнтів, покрокову інструкцію приготування, тип кухні, час приготування, складність, автора рецепту, нутріціологічну інформацію, а також додаткову інформацію для фільтрації чи персоналізації результатів пошуку;
- оцінки та відгуки – призначені для зберігання рецензій користувачів щодо рецептів, що дозволяє формувати рейтинги та покращувати рекомендації.

Логічна структура бази даних спроєктована таким чином, щоб забезпечити гнучку роботу з динамічними даними, оптимізовану під швидкий доступ до часто використовуваної інформації та масштабованість у разі збільшення обсягу даних.

Для забезпечення зручного користування реалізовано інтуїтивну навігацію, що містить основні сторінки:

- головна сторінка – шапку, hero-секцію, пошук, популярні рецепти, кнопка «Показати більше», footer;
- сторінка рецепта – зображення, час приготування, тип кухні, складність, кількість порцій, дієти, нутріціологічну інформацію, рейтинг, інгредієнти, інструкцію з приготування, коментарі;
- сторінка всіх рецептів та результатів пошуку і фільтрації, сортування;

- сторінка користувача – аватар, інформація про автора, всі його опубліковані рецепти;
- реєстрація / вхід;
- сторінка додавання рецепта;
- результати пошуку.

2.3 Обґрунтування інструментальних засобів розробки «EasyIngrecipes»

Для реалізації вебплатформи «EasyIngrecipes» були обрані ефективні інструментальні засоби, які забезпечують високу швидкість розробки, масштабованість, зручність у підтримці та відповідність сучасним вимогам до вебзастосунків. До основних технологій належать: Next.js, як фреймворк, що забезпечує повноцінну інтеграцію клієнтської та серверної логіки, React як бібліотека для побудови інтерфейсу користувача, а також MongoDB як документно-орієнтована база даних для зберігання інформації.

Основною платформою для реалізації інтерфейсу та логіки взаємодії з сервером було обрано Next.js – фреймворк для React, який забезпечує як статичний, так і серверний рендеринг, що сприяє кращій продуктивності, SEO-оптимізації та швидкому завантаженню сторінок. Завдяки можливостям SSR (Server-Side Rendering) та API-роутів, Next.js дозволяє реалізувати повноцінний Fullstack-застосунок, де фронтенд і бекенд інтегруються в єдиній екосистемі. [5]

React.js – це популярна JavaScript-бібліотека, яка має ряд суттєвих переваг для розробки вебзастосунків. Однією з ключових є компонентний підхід, що дозволяє структурувати інтерфейс як набір незалежних блоків. Це значно полегшує повторне використання коду та його обслуговування. Ще однією важливою особливістю є віртуальний DOM, який забезпечує високу продуктивність завдяки ефективному оновленню лише змінених частин інтерфейсу. Крім того, React відзначається гнучкістю та хорошою інтеграцією з

іншими бібліотеками та фреймворками, що дає змогу використовувати його як у невеликих проєктах, так і у масштабних системах з широким функціоналом. [14]

MongoDB – документно-орієнтовану систему зберігання даних, яка ідеально підходить для структурування інформації про рецепти, користувачів та оцінки. Завдяки своїй гнучкій моделі зберігання JSON-подібних документів, MongoDB дозволяє зручно працювати з вкладеними структурами, такими як масиви інгредієнтів чи кроків приготування, а також легко масштабувати застосунок при розширенні функціональності.

Порівняно з іншими варіантами (наприклад, класичними MVC-фреймворками на PHP чи Python), Next.js забезпечує тісну інтеграцію клієнтської та серверної частини застосунку, можливість реалізації API без окремого бекенд-сервера, підтримку SEO-оптимізованого SSR (Server Side Rendering) та SPA-архітектури.

Переваги обраного стеку (Next.js + React + MongoDB):

- висока швидкість завантаження сторінок завдяки гібридному рендерингу (SSR/SSG);
- простота масштабування й обслуговування проєкту;
- реалізація серверної логіки через API-роути безпосередньо в Next.js;
- гнучка робота з неструктурованими або змінними даними в MongoDB;
- широка екосистема, велика спільнота та наявність готових рішень;
- можливість розгортання як статичного сайту, так і повноцінного серверного застосунку.

Таким чином, вибір Next.js, React та MongoDB для створення повнофункціонального fullstack-застосунку є технічно обґрунтованим та оптимальним для реалізації цілей і вимог, визначених у рамках платформи EasyIngrecipes.

Для створення дизайн-макетів використано Figma – сучасний інструмент для роботи з UX/UI дизайном. Однією з переваг є можливості розробки дизайну продукту, прототипування, ілюстрації та моушн-дизайну. Все це дозволяє

ефективно спланувати вигляд і структуру вебдодатка перед початком його реалізації. [15]

У проєкті також використано хмарне сховище Cloudinary для завантажувати зображення страв та автарів користувачів, які автоматично обробляються та оптимізуються для швидкого відображення на сайті. Cloudinary – це компанія SaaS, яка надає послуги керування хмарними медіа для веб-сайтів і програм. [16] Хмарна платформа для завантаження, зберігання, оптимізації та доставки зображень і відео. Вона забезпечує автоматичну обробку медіафайлів, включаючи зміну розміру, обрізку, стиснення та трансформації, що дозволяє адаптувати контент до різних пристроїв і умов мережі. Завдяки CDN-мережі, Cloudinary забезпечує швидку та надійну доставку медіаконтенту користувачам по всьому світу. [17] Це покращує користувацький досвід, зменшує навантаження на сервер та забезпечує масштабованість при зростанні кількості користувачів і контенту. Крім того, Cloudinary підтримує безпечне зберігання медіафайлів та надає інструменти для керування доступом до них.

2.4 Особливості програмної реалізації «EasyIngrecipes»

Вебдодаток реалізовано з використанням фреймворку Next.js, що забезпечує серверний рендеринг та покращує продуктивність. Інтерфейс побудований на основі React, що дозволяє створювати динамічні компоненти та повторно використовувати їх. Для стилізації інтерфейсу застосовано модульні CSS-стилі, що забезпечують ізоляцію стилів кожного компонента та запобігають конфліктам і перекриттям класів. Також використано Tailwind CSS для швидкої побудови адаптивного дизайну з використанням утилітарних класів. Такий підхід дозволяє значно пришвидшити розробку та забезпечити узгоджений вигляд інтерфейсу на різних пристроях. Це дозволяє підтримувати чисту та масштабовану структуру коду.

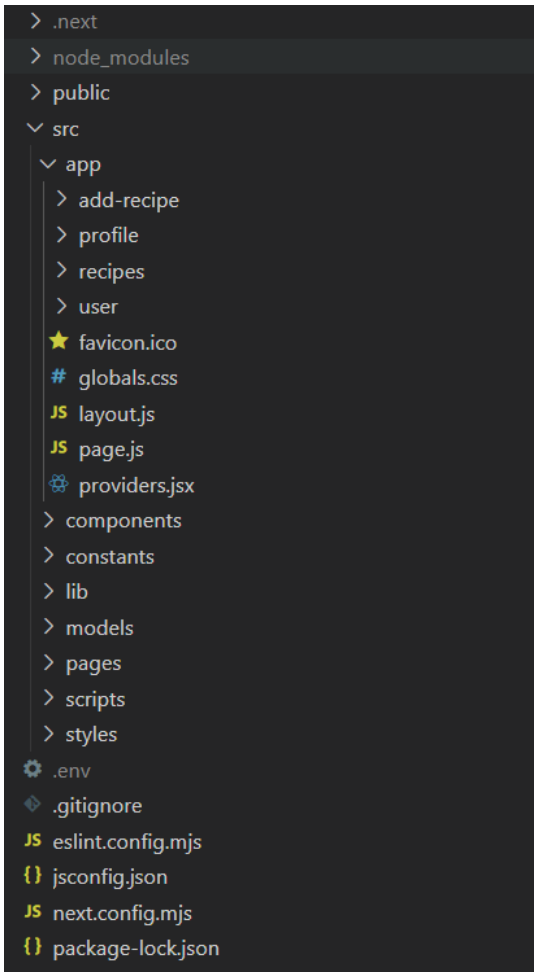


Рисунок 2.3 – Структура проекту

На рисунку 2.3 представлено структуру проєкту вебдодатку. Вона охоплює основні папки та файли, що забезпечують як клієнтську, так і серверну частину розробки. Така організація дозволяє легко орієнтуватися у функціоналі проєкту, розмежовує відповідальність між компонентами та сприяє масштабованості застосунку.

Папка app містить усі сторінки вебдодатку, включаючи головну сторінку, сторінку з окремим рецептом, сторінку профілю користувача, профілю користувача, додавання рецепту. Тут також знаходиться файл layout.js, який відповідає за загальну структуру сторінки, який має фіксовані блоки шапки, футера, та провайдерів (як-от контекст авторизації). Головна сторінка page.js є точкою входу, де відображаються основні компоненти: пошук, рецепти, інформація про платформу.

```
import { Geist, Geist_Mono } from "next/font/google";
import "./globals.css";
import Header from "../components/Header/Header";
import Footer from "../components/Footer/Footer";
import Providers from "./providers";

const geistSans = Geist({ ···
});

const geistMono = Geist_Mono({ ···
});

export const metadata = { ···
};

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body
        className={`${geistSans.variable} ${geistMono.variable} antialiased`}
      >
        <Providers>
          <Header />
          {children}
          <Footer />
        </Providers>
      </body>
    </html>
  );
}
```

Рисунок 2.4 – Реалізація файлу layout.js

У файлі layout.jsx що на рисунку 2.4 імпортуються шрифти з Google Fonts (Geist, Geist_Mono) та застосовуються до тіла документа через відповідні класи, що забезпечує сучасний та гармонійний зовнішній вигляд тексту. Глобальні стилі підключено через файл globals.css, завдяки чому встановлюється єдина стилістика для всього проєкту. До структури сторінки додаються компоненти Header та Footer, які забезпечують єдиний шаблон з постійним заголовком і нижнім колонтитулом на кожній сторінці сайту. Крім того, компонент Providers використовує SessionProvider з пакету next-auth/react, що дозволяє зберігати та підтримувати сесію автентифікації користувачів протягом усього їх перебування на сайті.

```
import { connectToDatabase } from "../lib/mongodb";
import Recipe from "../models/Recipe";
import HeroSection from "../components/HomePage/HeroSection/HeroSection";
import AboutUs from "@/components/HomePage/AboutUsSection/AboutUs";
import PopularRecipeSection from "@/components/HomePage/PopularRecipeSection/Po

// Fetch popular recipes server-side
async function getPopularRecipes() {
  try {
    await connectToDatabase();

    // Find top rated recipes
    const recipes = await Recipe.find().sort({ rating: -1 }).limit(4).lean();

    return {
      recipes: JSON.parse(JSON.stringify(recipes)),
      error: null,
    };
  } catch (error) {···
  }
}

export default async function Home() {
  const popularRecipesData = await getPopularRecipes();

  return (
    <>
      <HeroSection />
      <PopularRecipeSection initialData={popularRecipesData} />
      <AboutUs />
    </>
  );
}
```

Рисунок 2.5 – Реалізація файлу page.js

У цьому файлі page.js що на рисунку 2.5 здійснюється серверний запит до бази даних MongoDB за допомогою функції connectToDatabase, після чого за допомогою моделі Recipe відбувається вибірка чотирьох найкращих за рейтингом рецептів. Отримані дані передаються у компонент PopularRecipeSection через проп initialData, що дозволяє відобразити популярні рецепти на головній сторінці. Окрім цього, структура сторінки містить компоненти HeroSection, який відповідає за вітальну секцію сайту, та AboutUs, який надає інформацію про платформу. Такий підхід дозволяє поєднати SSR (server-side rendering) із компонентною архітектурою для забезпечення швидкого завантаження та SEO-оптимізації сторінки.

Папка components містить окремі частини інтерфейсу, які використовуються на різних сторінках додатку. Серед них – такі елементи, як RecipeCard (картка рецепта), Header (верхня панель навігації), AuthModal (модальне вікно авторизації), RegisterForm (форма реєстрації) , AddRecipeModal (форма для публікації рецепту) та інші. Також в папці є багаторазові компоненти інтерфейсу, такі як Button. Компонент Button – багаторазова кнопка з підтримкою різних стилів, яку можна адаптувати під конкретні дії або контексти в інтерфейсі (наприклад, кнопка входу, додавання рецепта). Ці компоненти реалізовані як незалежні модулі з чітко визначеною відповідальністю, що спрощує підтримку і повторне використання коду. Крім того, у цій папці розміщено файли модульних стилів, які забезпечують ізольовану стилізацію кожного компонента, що знижує ризик конфліктів між CSS- класами та підвищує гнучкість оформлення інтерфейсу.

```jsx
import React from "react";
import classNames from "classnames";
import styles from "./Button.module.scss"; // Стилі для кнопки

const Button = ({
  variant = "primary",
  size = "normall",
  className,
  children,
  ...props
}) => {
  return (
    <button
      className={classNames(
        styles.button,
        styles[variant],
        {
          [styles[size]]: size,
        },
        className
      )}
      {...props}
    >
      {children}
    </button>
  );
};

export default Button;
```

Рисунок 2.5 – Реалізація компоненти Button.jsx

```
.button {
  border-radius: 15px;
  cursor: pointer;
  text-align: center;
  width: auto;
  white-space: nowrap;
  padding: 6px 12px;
  color: black;
  &:disabled {
    cursor: not-allowed;
    background: #f9fafb;
    color: #d2d6db;
    border: 1px solid #d2d6db;
  }
  &:active {
    border: none;
  }
}
.primary {···
}
.secondary {···
}
.third {···
}
.bigSize {
  width: 264px;
}
.addIteam {···
}
```

Рисунок 2.6 – Реалізація стилів компонети Button.jsx

Папка pages/api містить API-роути, які обробляють серверні запити, наприклад, авторизацію, реєстрацію користувачів, збереження рецептів, завантаження зображень тощо. Ці файли є обгортками до логіки роботи з базою даних MongoDB, перевірки прав доступу, обробки форм тощо. Таким чином забезпечується повноцінна підтримка клієнт-серверної архітектури.

У наведеному нижче рисунку 2.7 представлено код, де реалізовано API-обробник для обробки запиту POST при реєстрації нового користувача. Спочатку відбувається підключення до бази даних через функцію connectToDatabase, після чого перевіряється, чи не існує вже користувача з таким email. Якщо користувача не знайдено, пароль хешується за допомогою бібліотеки bcryptjs для забезпечення безпеки, і створюється новий запис користувача у базі з полями

username, email, bio, avatar і захешованим password. У випадку успішного створення, API повертає статус 201 і мінімальну інформацію про нового користувача. Якщо запит надійшов методом, відмінним від POST, сервер повертає статус 400 з відповідним повідомленням про те, що метод не підтримується.

```
export default async function handler(req, res) {
  if (req.method === "POST") {
    const { name, email, password, avatar, bio } = req.body;

    // Підключення до бази даних
    await connectToDatabase();

    // Перевірка, чи користувач вже існує
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res
        .status(400)
        .json({ message: "Користувач з таким email вже існує." });
    }

    // Хешування пароля
    const hashedPassword = await bcrypt.hash(password, 10);

    // Створення нового користувача
    const newUser = new User({
      username: name,
      email,
      password: hashedPassword,
      bio,
      avatar,
    });
    await newUser.save();

    // Відправка відповіді
    res.status(201).json({
      message: "Користувач успішно зареєстрований",
      user: {
        id: newUser._id,
        email: newUser.email,
        username: newUser.username,
      },
    });
```

Рисунок 2.7 – register.js

У папці scripts зберігаються допоміжні скрипти, що використовуються для первинного заповнення бази даних рецептами, користувачами або інгредієнтами. Крім того, тут містяться функції для взаємодії з Cloudinary – сервісом для

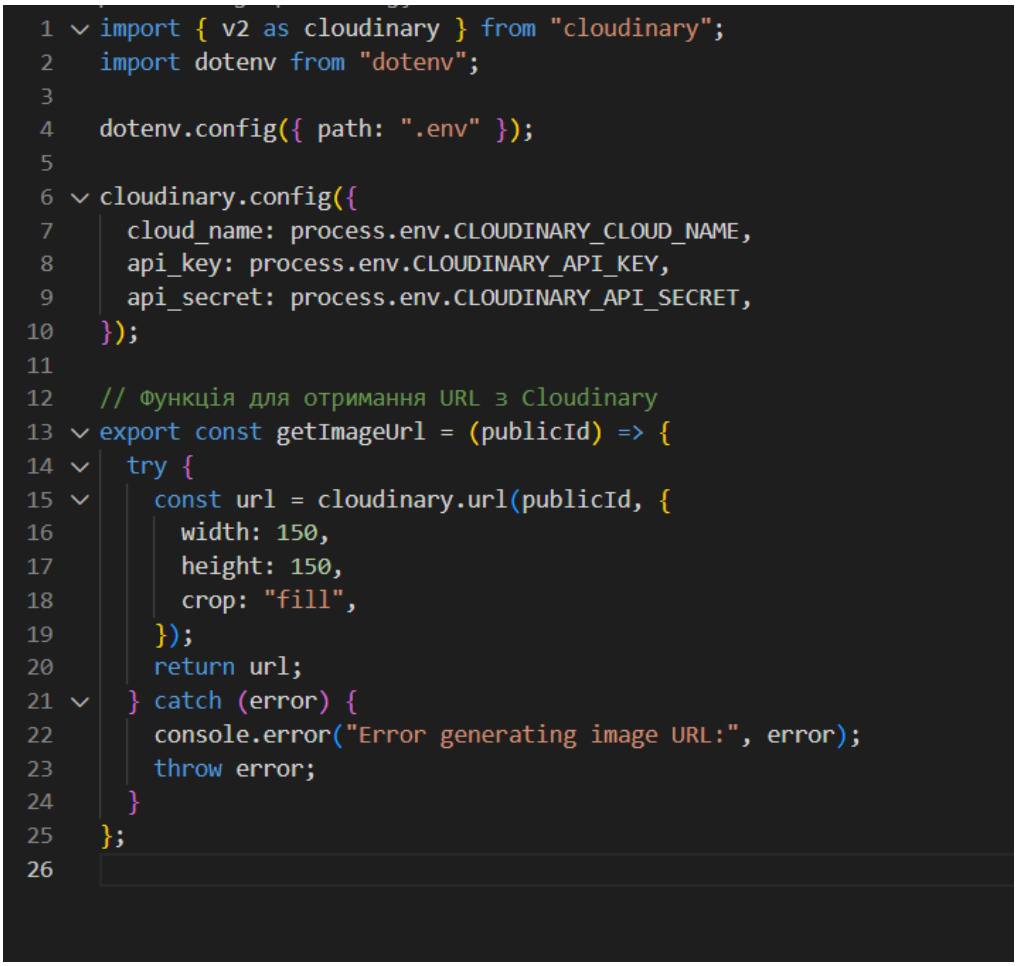
збереження зображень у хмарі, зокрема для завантаження фото страв та автарів користувачів.

На рисунку 2.8 представлено фунцію, що реалізує завантаження зображень до хмарного сховища Cloudinary. Функція uploadToCloudinary приймає файл, формує об’єкт FormData з потрібними параметрами та надсилає POST-запит до API Cloudinary. У разі успіху вона повертає безпечний URL зображення.

```
export const uploadToCloudinary = async (file) => {
  const formData = new FormData();
  formData.append("file", file);
  formData.append("upload_preset", "user_avatar");
  formData.append("folder", "users/avatars");

  const response = await fetch(
    "https://api.cloudinary.com/v1_1/dz1kvzidt/image/upload",
    {
      method: "POST",
      body: formData,
    }
  );

  if (!response.ok) {
    throw new Error("Помилка завантаження зображення");
  }

  const data = await response.json();
  return data.secure_url;
};
```

Рисунок 2.8 – Реалізація функції uploadToCloudinary.js, що завантажує зображення до Clodinary

Окремо у файлі imageUploadConfig.js, що реалізований на рисунку 2.9 налаштовується підключення до Cloudinary через змінні середовища, а також реалізовано функцію getImageUrl, що генерує URL зображення за його publicId з заданими параметрами. Це забезпечує ефективну роботу з зображеннями на клієнтській та серверністороні.



```
import { v2 as cloudinary } from "cloudinary";
import dotenv from "dotenv";

dotenv.config({ path: ".env" });

cloudinary.config({
  cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
  api_key: process.env.CLOUDINARY_API_KEY,
  api_secret: process.env.CLOUDINARY_API_SECRET,
});

// Функція для отримання URL з Cloudinary
export const getImageUrl = (publicId) => {
  try {
    const url = cloudinary.url(publicId, {
      width: 150,
      height: 150,
      crop: "fill",
    });
    return url;
  } catch (error) {
    console.error("Error generating image URL:", error);
    throw error;
  }
};
```

Рисунок 2.9 – Реалізація функції функцію getImageUrl

База даних MongoDB використовується для зберігання інформації про рецепти, користувачів, інгредієнти, рейтинги. Для взаємодії з базою даних використовується ORM-бібліотека Mongoose, що спрощує опис моделей і запитів до MongoDB. Завдяки Mongoose забезпечується валідація даних, зв’язки між моделями та зручна структура для подальшої обробки даних у вебдодатку.

У наведеному нижче рисунку 2.10 представлено код, реалізовано функцію підключення до бази даних MongoDB з використанням бібліотеки mongoose. Спочатку перевіряється наявність змінної середовища MONGODB_URI, яка містить рядок підключення до бази даних. Якщо змінна не вказана, викидається помилка. Далі реалізовано механізм кешування з’єднання, щоб уникнути повторних підключень під час роботи застосунку, особливо в середовищі розробки або при serverless-архітектурі (наприклад, у Next.js). Якщо з’єднання вже існує (cached.conn), воно повторно використовується. Інакше створюється

нове з’єднання (cached.promise) і зберігається в кеш. Також у глобальну область видимості (global) додається об’єкт mongoose, щоб кеш зберігався між перезавантаженнями модулів. Це рішення забезпечує ефективне, надійне та безпечне підключення до MongoDB без зайвих викликів mongoose.connect().

```
import mongoose from "mongoose";

const MONGODB_URI = process.env.MONGODB_URI;

if (!MONGODB_URI) {
  throw new Error("Будь ласка, додайте MONGODB_URI у .env");
}

// Кеш для уникнення повторних підключень
let cached = global.mongoose || { conn: null, promise: null };

export async function connectToDatabase() {
  if (cached.conn) {
    return cached.conn;
  }

  if (!cached.promise) {
    cached.promise = mongoose
      .connect(MONGODB_URI, {
        bufferCommands: false,
      })
      .then((mongoose) => mongoose);
  }

  cached.conn = await cached.promise;
  return cached.conn;
}

// Збереження кешу в глобальну область видимості
if (!global.mongoose) {
  global.mongoose = cached;
}
```

Рисунок 2.10 – Реалізація функції підключення до бази даних MongoDB

```
useEffect(() => {
  const fetchRecipes = async () => {
    try {
      setLoading(true);

      // Construct query params from filters
      const queryParams = new URLSearchParams();
      Object.entries(filters).forEach(([key, value]) => {
        if (value) {
          queryParams.append(key, value);
        }
      });

      const response = await fetch(
        `/api/filter-recipes?${queryParams.toString()}`
      );

      if (!response.ok) {
        throw new Error("Помилка при отриманні рецептів");
      }

      const data = await response.json();
      setRecipes(data);
      setError(null);
    } catch (err) {
      console.error("Error fetching recipes:", err);
      setError("Не вдалося завантажити рецепти. Спробуйте пізніше.");
    } finally {
      setLoading(false);
    }
  };
```

Рисунок 2.11 – Реалізацію фільтрації рецептів за допомогою useEffect

У фрагменті коду, що на риснку 2.11 наведено реалізацію фільтрації рецептів за допомогою React-хука useEffect. Цей хук відслідковує зміни об'єкта filters і при кожній зміні виконує асинхронний запит до серверного API для отримання відфільтрованих даних. Параметри фільтрів формуються у вигляді URL-параметрів, які додаються до запиту. Після отримання відповіді з сервера дані конвертуються у формат JSON і зберігаються у стані компонента, що забезпечує оновлення інтерфейсу відповідно до вибраних фільтрів. Також реалізовано обробку помилок і індикацію завантаження, що підвищує зручність користування вебзастосунком.

```
export default async function handler(req, res) {
  // Викликаємо CORS
  runCors(req, res, async () => {
    try {
      await connectToDatabase(); // Підключення до БД

      const {···
      } = req.query;

      // Об'єкт фільтрації
      const filter = {
        ...(mealType && { type: mealType }),
        ...(difficulty && { difficulty }),
        ...(diet && { diet: { $in: diet.split(",") } }),
        ...(cuisine && { cuisine }),
        ...(cookingTime && {···
        }),
        ...(includeIngredients && {···
        }),
        ...(excludeIngredients && {···
        }),
        ...(searchTerm && {···
        }),
      };
```

Рисунок 2.12 – Реалізація API-обробнику /api/filter-recipes

На риснку 2.12 продемонстровано об'єкт filter в API-обробнику /api/filter-recipes, який формує умови фільтрації для отримання рецептів з бази даних. У фільтр динамічно додаються параметри запиту: тип страви, складність, дієта, кухня, час приготування, інгредієнти та ключові слова.

2.5 Організація тестування та налагодження програмного засобу «EasyIngrecipes»

Під час тестування були застосовані як ручні, так і автоматизовані методи. Основна увага приділялася функціональності входу та реєстрації користувача, додаванню та пошуку рецептів, фільтрації і роботі з хмарним сховищем Cloudinary. Перевірка відбувалася у середовищах Chrome, Firefox і Safari, а також на мобільних пристроях.

Для модульного тестування окремих функцій (наприклад, валідації форм, взаємодії з API або завантаження зображень) було підготовлено базові юніт-тести з використанням бібліотеки Jest. Зокрема, перевірялися такі функції, як uploadToCloudinary, відповідь сервера на POST /api/register, а також логіка хешування пароля з використанням bcryptjs.

У процесі тестування були виявлені такі недоліки:

- некоректне відображення кнопки на мобільних пристроях при певних розмірах екрану – усунуто шляхом адаптивного стилювання;
- відсутність перевірки на довжину пароля під час реєстрації – додано перевірку у frontend-формі;
- повідомлення про помилку при завантаженні не завжди було видимим – реалізовано окремий компонент сповіщення;
- уповільнена реакція при масовому підвантаженні рецептів – реалізовано lazy-loading та попередню перевірку кешованих запитів.

Загалом результати тестування підтвердили функціональну готовність застосунку, а виявлені недоліки були оперативно виправлені.

2.6 Рекомендації по використанню та впровадженню програмного засобу

Програмним продуктом можна скористатися за наявності сучасного веббраузера, як Google Chrome, Mozilla Firefox чи Microsoft Edge, перейшовши за посиланням. Також необхідним є наявність стабільного інтернет-з’єднання. Інтерфейс продукту є адаптивним, тому зручно переглядати, як з комп’ютеру, так і мобільного пристрою. Більше можна дізнатися у Додатку Б.

ВИСНОВКИ

У результаті виконаної бакалаврської роботи було розроблено вебзастосунок для пошуку кулінарних рецептів за інгредієнтами, що відповідає поставленим завданням: реалізовано функціонал пошуку, фільтрації, реєстрації та авторизації користувачів, додавання та перегляду рецептів.

У процесі розробки було використано сучасний стек технологій: React для створення інтерфейсу, Node.js і MongoDB для бекенда, а також Cloudinary для зберігання зображень у хмарі. Для зручності обробки даних і моделювання структури бази даних застосовано бібліотеку Mongoose.

Інтерфейс сайту побудовано на принципах повторного використання компонентів, що підвищує масштабованість і зручність супроводу проєкту. Зокрема, створено багаторазові компоненти кнопок, карток рецептів, модальних вікон тощо.

Для перевірки працездатності функціоналу було проведено тестування вручну, а також розроблено базові модульні тести. Під час тестування виявлено окремі недоліки у відображенні інтерфейсу та валідації форм, які були усунені.

Серед проблем, які потребують подальшого вдосконалення, можна відзначити обмежене покриття тестами, відсутність підтримки багатомовності та оптимізації продуктивності при великій кількості рецептів. У майбутньому доцільно реалізувати:

- розширене автоматизоване тестування компонентів та API;
- персоналізовані рекомендації на основі історії користувача;
- систему тегів та інтелектуального пошуку.

Отримані результати підтверджують актуальність обраної теми та демонструють практичну реалізацію інструменту, що може бути корисним для широкої аудиторії користувачів. Програмний засіб має потенціал для подальшого розвитку як у рамках навчального, так і комерційного проєкту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Переваги Next.JS: аналізуємо з точки зору розробника та замовника | Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/front-end-bez-nextjs-yak-sportkar-bez-dviguna-analizujemo-perevagi-ta-nedoliki (дата звернення: 21.05.2025).
2. DAN IT Education. Розробка з боку Front end – що це таке і чим відрізняється від Back end?. *DAN IT Education*. URL: https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaietsja-vid-back-end/ (дата звернення: 21.05.2025).
3. DAN IT Education. 5 найпопулярніших фреймворків для веброзробки. *DAN IT Education*. URL: https://dan-it.com.ua/uk/blog/5-najpopuljarnishih-frejmvorkiv-dlja-vebrozrobki/. (дата звернення: 22.05.2025).
4. Що таке Node.js: визначення, особливості та застосування | Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/vse-chto-nuzhno-znat-o-nodejs (дата звернення: 22.05.2025).
5. Що таке Next.JS: сучасна технологія для веб-розробки | Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/rozbirajemo-freymvork-nextjs-viznachennya-priznachennya-ta-perevagi (дата звернення: 22.05.2025).
6. Найкращі фреймворки Node.js: розбираємо їх особливості | Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/luchshie-freymvorki-nodejs (дата звернення: 22.05.2025).
7. Фреймворки для Python: 12 найкращих для веб-розробки | Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/python-frameworks (дата звернення: 22.05.2025).

8. REDSTONE. Інтеграція хмарних сервісів у ваші веб-додатки: можливості та переваги. *REDSTONE*. URL: https://redstone.agency/blog/intehratsiia-khmarnykh-servisiv-u-vashi-veb-dodatky-mozhlyvosti-ta-perevahy/ (дата звернення: 23.05.2025).
9. SQL чи NoSQL - ось в чому питання - Альтернативна Наука Віталія Сема. *Альтернативна Наука Віталія Сема*. URL: https://alternativescience.net/programming/242-sql-chy-nosql-os-v-chomu-pytannya/ (дата звернення: 22.05.2025).
10. SPA в програмуванні: розбираємося з одностранічніками. *FoxmindEd*. URL: https://foxminded.ua/spa-u-prohramuvanni/ (дата звернення: 21.05.2025).
11. API • Глосарій • Corefy. *Corefy*. URL: https://corefy.com/uk/glossary/api (дата звернення: 21.05.2025).
12. Андрій Денисенко. Вступ до REST API – RESTful вебсервіси. *robot_dreams - онлайн-курси для фахівців у сфері big data, machine learning, data science | Робот Дрімс*. URL: https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi (дата звернення: 21.05.2025).
13. GraphQL що це за мова і як маніпулює даними для API. *FoxmindEd*. URL: https://foxminded.ua/graphql-shcho-tse/ (дата звернення: 23.05.2025).
14. brainlab. Що таке React JS? Як почати вивчати Реакт?. *brainlab*. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-react-js-yak-pochaty-vyvchaty-reakt (дата звернення: 23.05.2025).
15. Що таке Figma і навіщо вона потрібна?. *DAN IT Education*. URL: https://dan-it.com.ua/uk/blog/chto-takoe-figma-i-zachem-ona-nuzhna/ (дата звернення: 23.05.2025).
16. Contributors to Wikimedia projects. Cloudinary - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Cloudinary (date of access: 23.05.2025).
17. Cloudinary. Про продукт, інтеграцію. *Автоматизируй свой бизнес: продукты,интеграции,интеграторы*.URL: https://hellip.com/ua/product/cloudinary.html (дата звернення: 23.05.2025).

ДОДАТКИ

ДОДАТОК А

Технічне завдання

1. Вступ

«EasyIngrecipes» – вебзастосунок для пошуку кулінарних рецептів за інгредієнтами. Розробка належить до галузі інформаційних технологій, зокрема вебпрограмування та сервісів для організації побуту. Вебзастосунок призначений для широкого кола користувачів, які цікавляться приготуванням їжі та прагнуть ефективно використовувати наявні продукти. Об’єктом використання програми є кулінарні портали, побутові вебресурси, мобільні чи десктопні пристрої, що мають доступ до Інтернету.

2. Підстави для розробки

Складність у швидкому пошуку рецептів відповідно до потреб користувачів створила необхідність у створення вебсайту, який матиме можливість пошуку за інгредієнтами та розширених функцій фільтрації. Розробка ведеться згідно з навчальним планом та завданням на кваліфікаційну роботу, затвердженим кафедрою комп’ютерних наук і інформаційних технологій Волинського національного університету.

3. Призначення розробки

Вебдодаток призначений для забезпечення користувачів зручним інструментом для швидкого пошуку кулінарних рецептів за наявними інгредієнтами. Основне функціональне призначення полягає у можливості фільтрації результатів за різними критеріями, такими як час приготування, складність виконання, калорійність та інші параметри. Крім того, вебдодаток

дозволяє створювати та зберігати власні рецепти, отримувати інформацію про харчову цінність страв. Однією з важливих складових є підтримка інтерактивної взаємодії між користувачами – можливість залишати оцінки та коментарі до рецептів, а також переглядати профілі інших учасників платформи.

4. Вимоги до програми

4.1 Вимоги до функціональних характеристик

Сайт повинен забезпечувати наступні функціональні харектеристики:

- авторизація/реєстрація користувачів;
- пошук рецептів за інгредієнтами;
- додавання, редагування та видалення рецептів;
- сортування та фільтрація за параметрами;
- перегляд нутріціологічної інформації;
- можливість оцінки, коментування, збереження рецептів;
- перегляд інформації про автора рецепта;
- адаптивна верстка;
- lazy loading і пагінація для списку рецептів.

4.2 Вимоги до надійності

Вимоги, що забезпечують стабільну, безпечну та зручну роботу системи, належать такі:

- захист авторизованих дій токенами;
- обробка помилок на фронтенді та бекенді;
- перевірка коректності вхідних даних.

4.3 Умови експлуатації

Вебзастосунок функціонує в сучасних браузерах, таких як Google Chrome, Mozilla Firefox, Microsoft Edge, що забезпечує широку доступність і зручність використання. Програма не потребує локальної установки, оскільки працює у форматі вебдодатку.

4.4 Вимоги до інформаційної і програмної сумісності

Вебдодаток забезпечує інформаційну та програмну сумісність шляхом використання єдиного формату обміну даними – JSON, що дозволяє ефективно обробляти запити між клієнтом і сервером. Для розробки використовується стек технологій, до складу якого входять Next.js (на базі React із використанням API routes), база даних MongoDB, а також TailwindCSS для стилізації. Основною мовою програмування є JavaScript (ES6+) із розміткою JSX. У реалізації також застосовуються додаткові бібліотеки: React Hook Form для роботи з формами, Axios для HTTP-запитів, Mongoose для взаємодії з MongoDB.

4.5 Вимоги до маркування і упаковки

Назва платформи: EasyIngrecipes. Програмний продукт супроводжується README-файлом.

4.6 Вимоги до транспортування і збереження

Зберігання: у вигляді git-репозиторію на GitHub;

Транспортування: через zip-архів або хмарне сховище.

5. Стадії і етапи розробки

- аналітичний етап: збір вимог, вивчення аналогів;
- проєктування: макети (Figma), структурування компонентів;
- розробка: Frontend (Next.js, React) та Backend (API-роути, MongoDB);
- тестування: ручне та автоматизоване (unit + UI).

ДОДАТОК Б

Керівництво користувачу

Для користування вебсайту «EasyIngrecipes», користувачу необхідно відкрити сайт у браузері та перейти за посиланням на головну сторінку вебсайту.

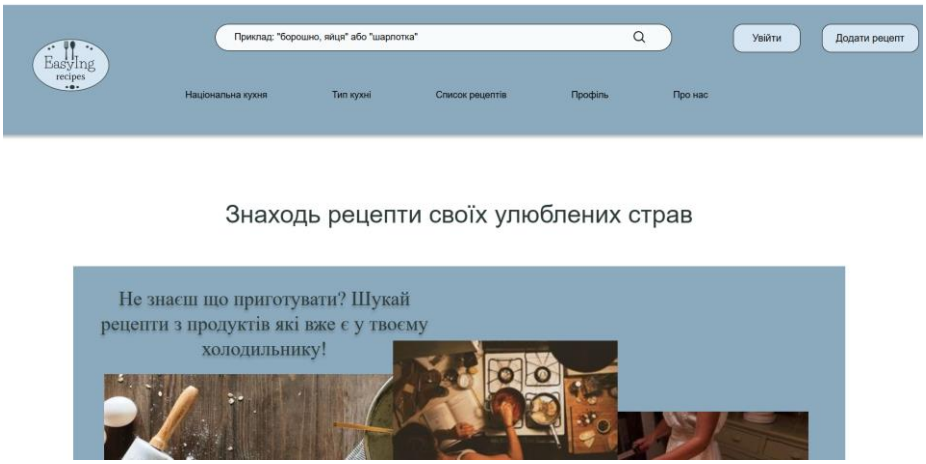


Рисунок Б.1 – Головна сторінка

Хедер головної сторінки містить, навігацію, поле пошуку рецептів за назвою чи інгредієнтами, кнопки авторизації та додання рецепту. Користувач може додати свій рецепт на сайт натиснувши кнопку «Додати рецепт», але для цього йому спочатку необхідно авторизуватися або зареєструватися.

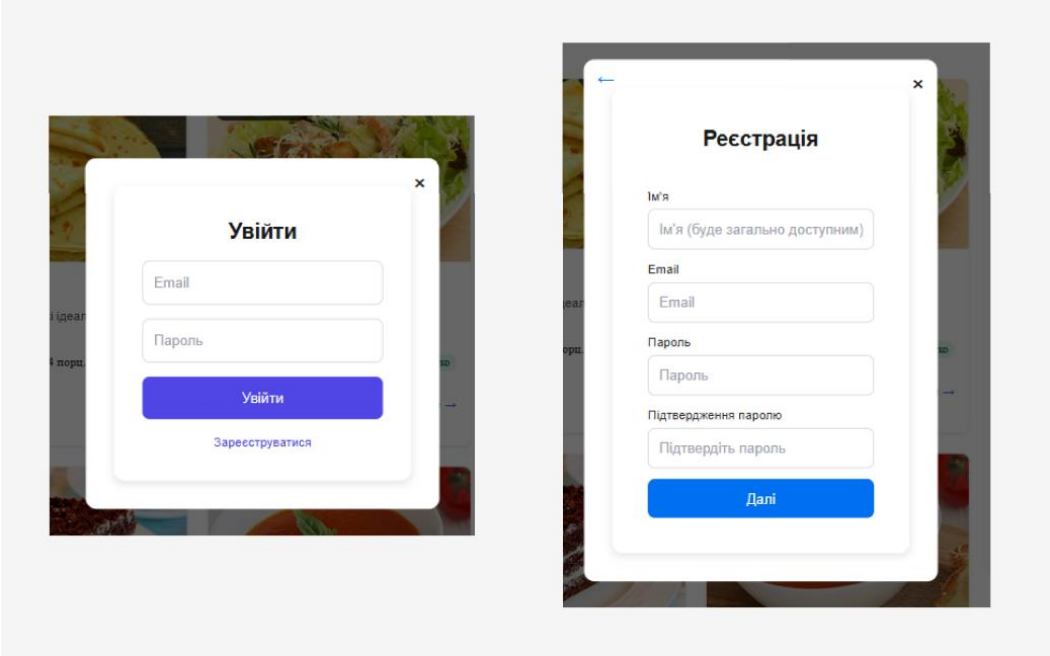


Рисунок Б.2 – Режими авторизації та реєстрації

Після авторизації/реєстрації користувач потрапляє на свій профіль, має можливість його відредагувати. Після авторизації користувач має можливість опубліковувати рецепти, створювати приватні колекції з своїх неопублікованих рецептів. З рецептів опублікованих іншими користувачами мати змогу створювати колекції, ставити оцінки і відгуки.

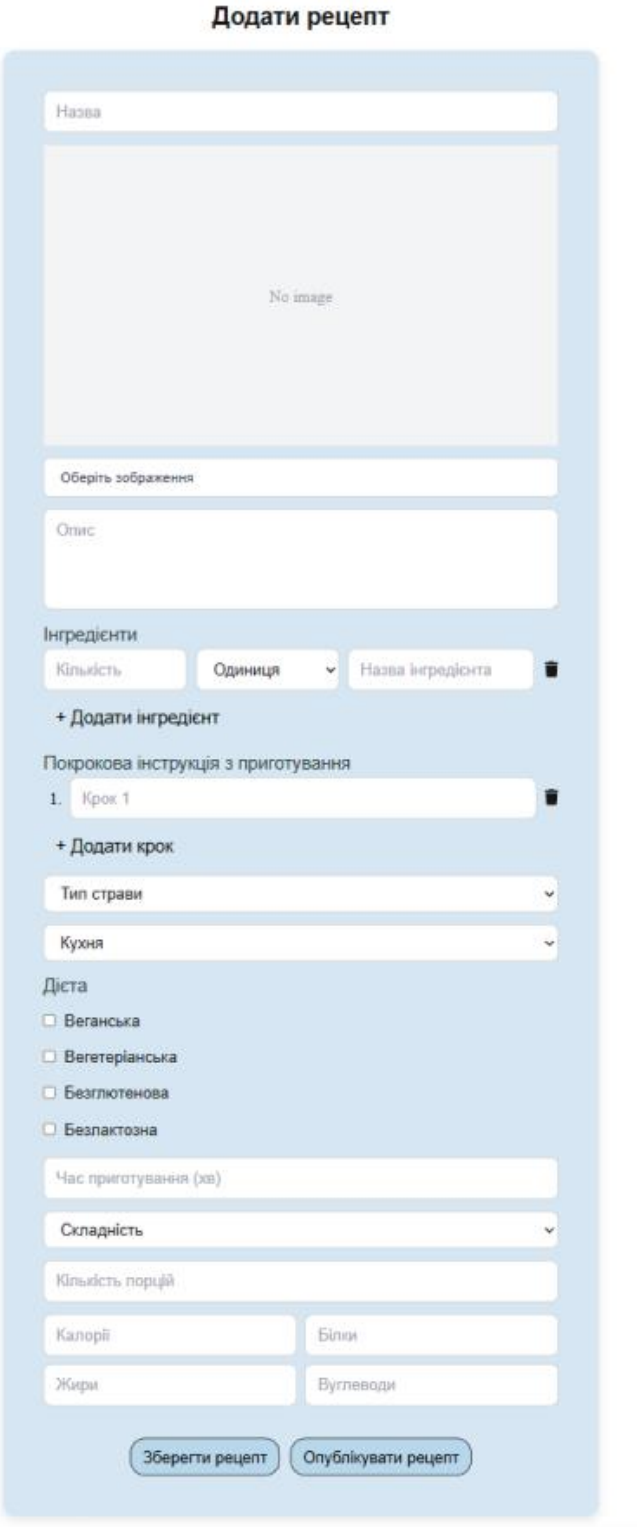


Рисунок Б.3 – Сторінка створення рецепту

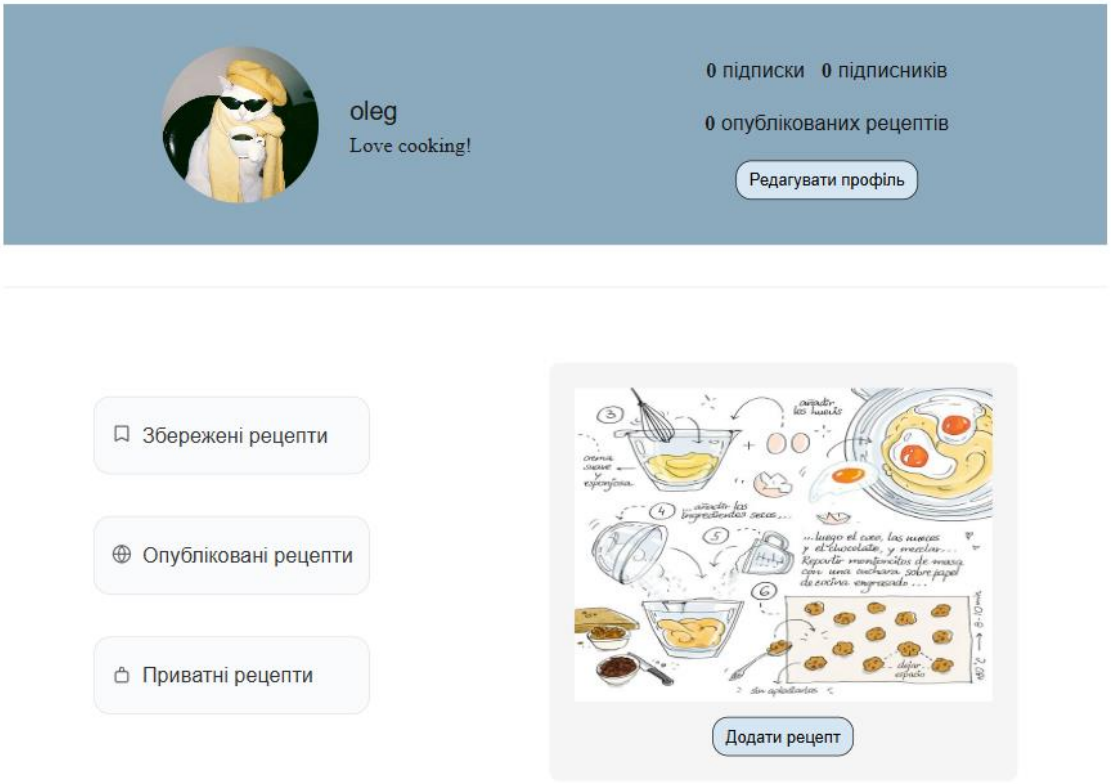


Рисунок Б.4 – Профіль авторизованого користувача

Головна сторінка має блок з чотирьох найпопулярніших рецептів і кнопки «Перегляну всі рецепти» при натисканні відповідно переходимо на сторінку зі всіма рецептами, яка містить форму фільтрації, сортування рецептів та можливості пошуку рецепту за інгредієнтами і без певних інгредієнтів.

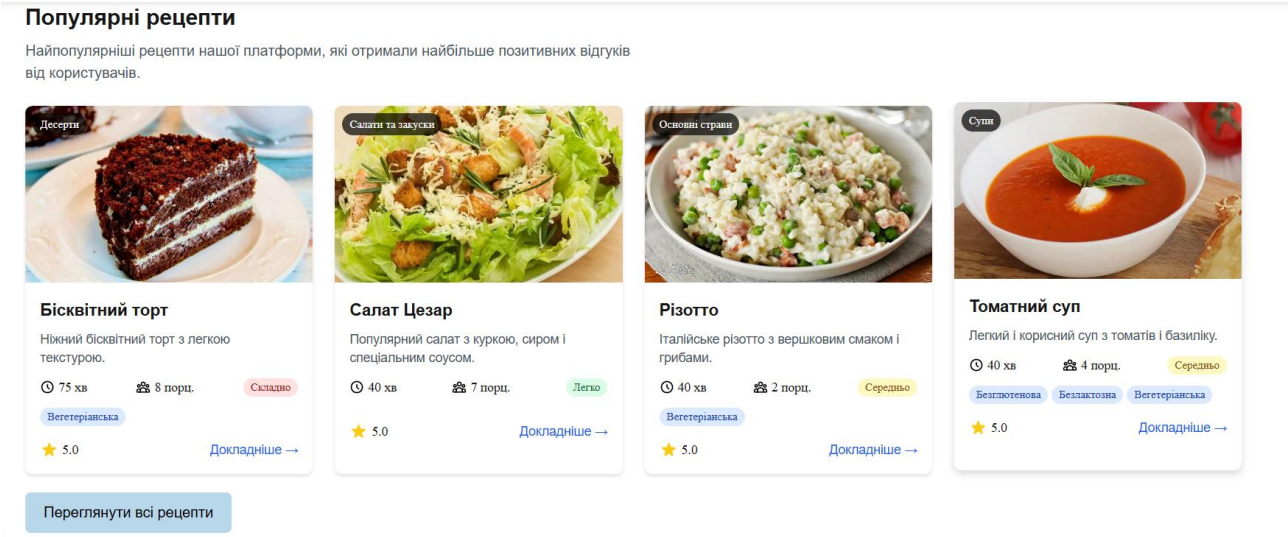


Рисунок Б.5 – Блок з популярними рецептами

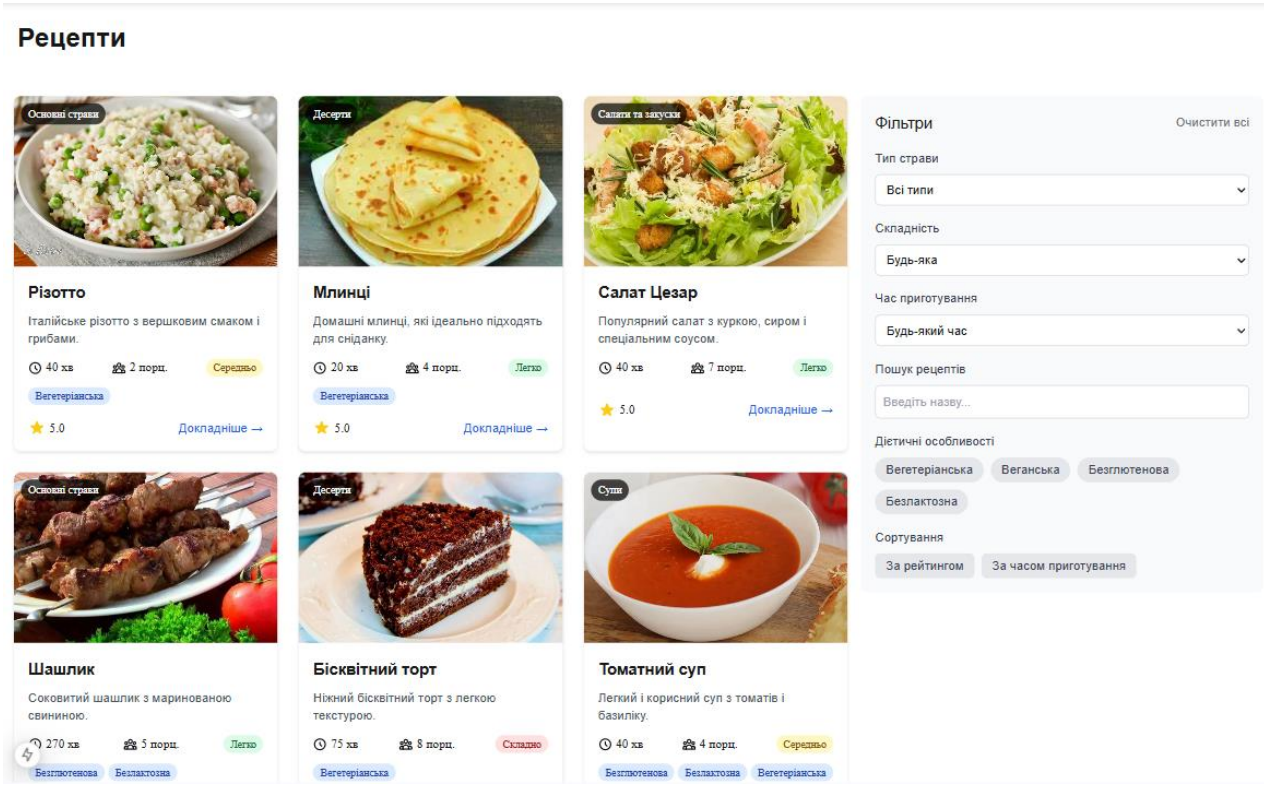


Рисунок Б.6 – Сторінка зі всіма рецептами та фільтрами і сортуванням

При натисканні на картку рецепта доступний детальний перегляд рецепту.

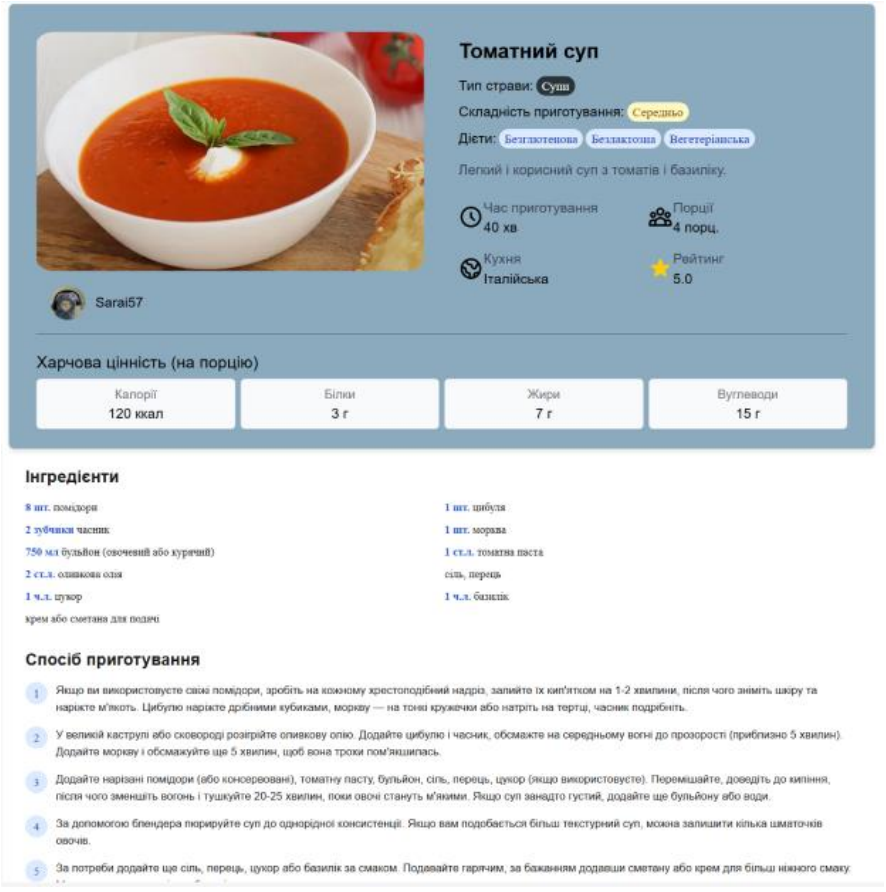


Рисунок Б.7 – Детальна сторінка рецепту

Окрім компонентів що на рисунку 3, сторінка рецепту має:

- поле в якому можна поділитися фото, результату приготування рецепту;
- перегляд та можливість залишити відгук.

На детальній сторінці рецепту при натисненні на фото чи ім’я користувача можна перейти на його профіль.

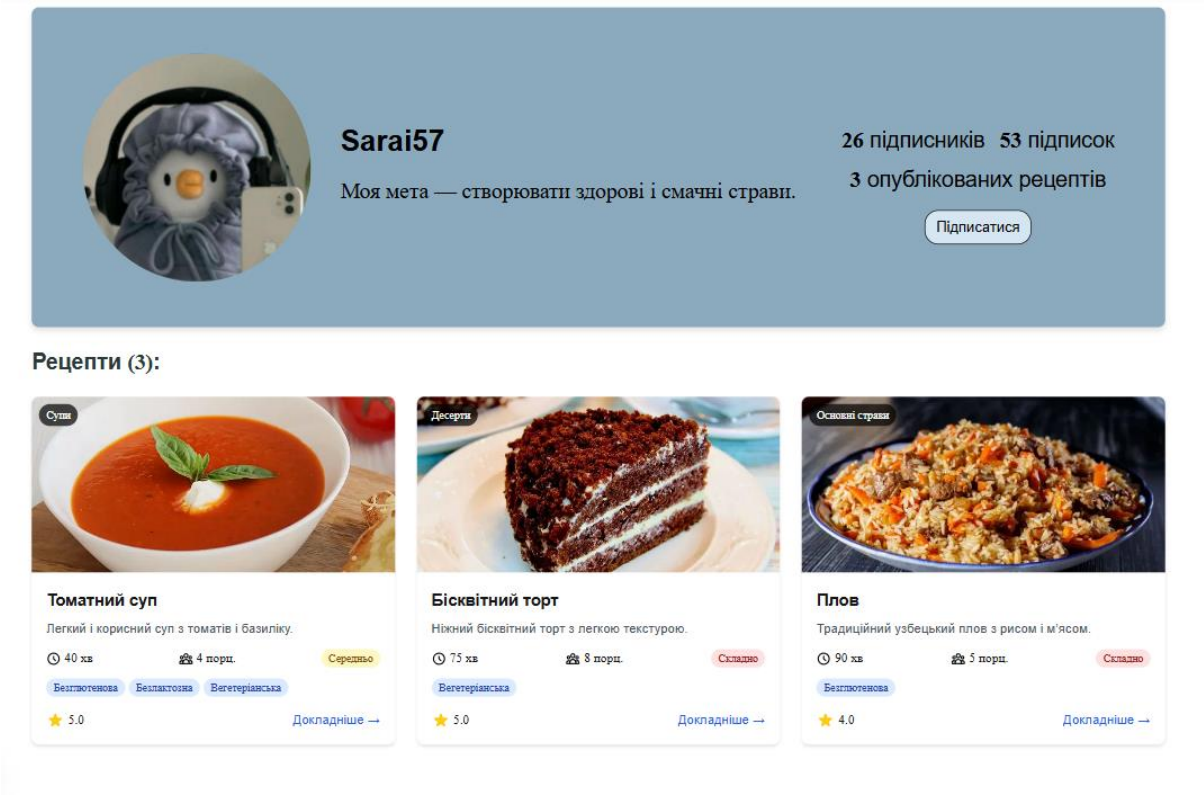


Рисунок Б.8 – Профіль користувача

Вебсайт є адаптивним, що забезпечує комфортну роботу на різних пристроях. Інші версії сайту має той самий функціонал, що й десктопна: пошук рецептів за інгредієнтами, фільтри, а також перегляд інформації про рецепти. Інтерфейс оптимізований для екранів різних розмірів, із зручним розташуванням елементів навігації та функцією скролу.

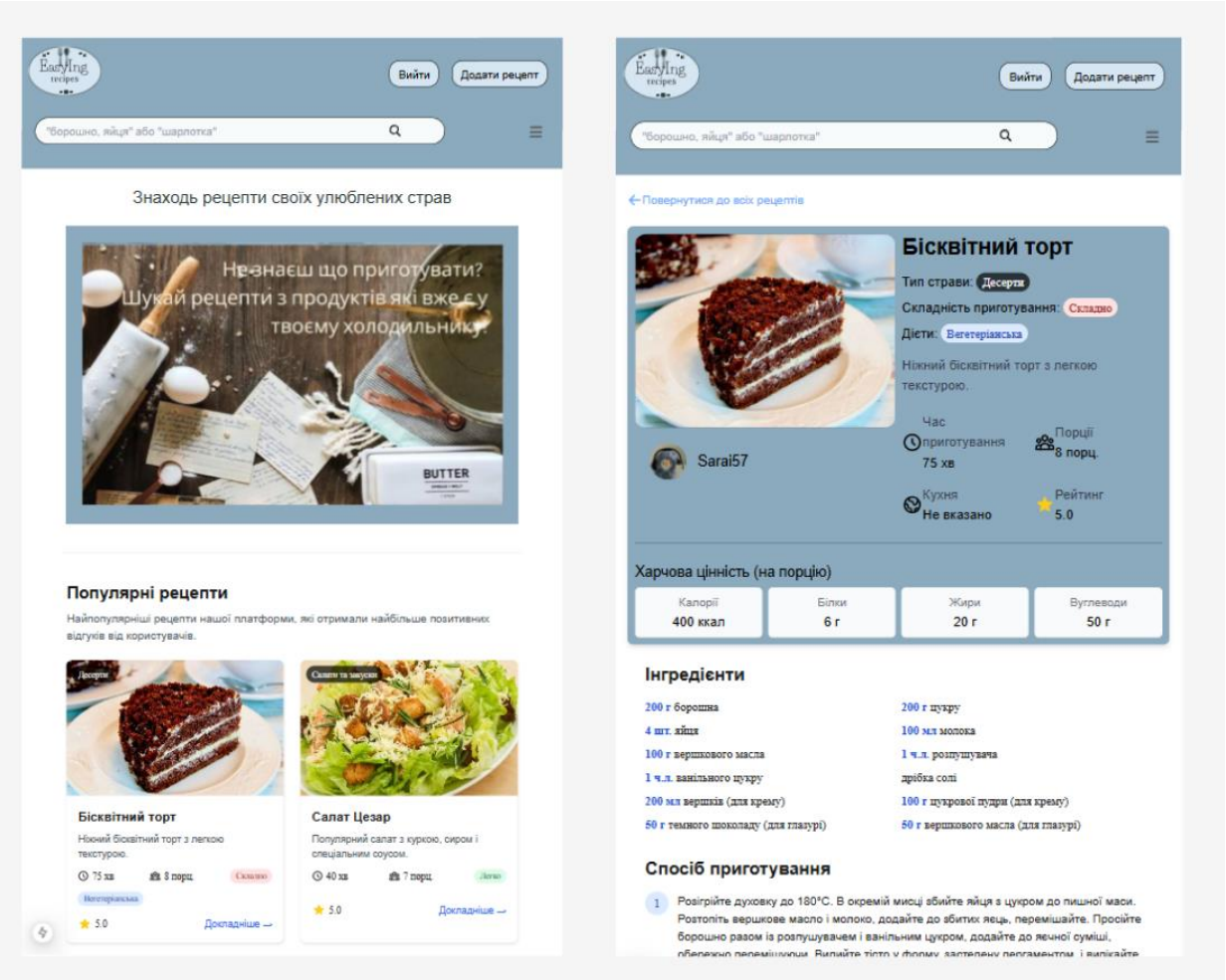


Рисунок Б.9 – Вигляд сторінок для планшету

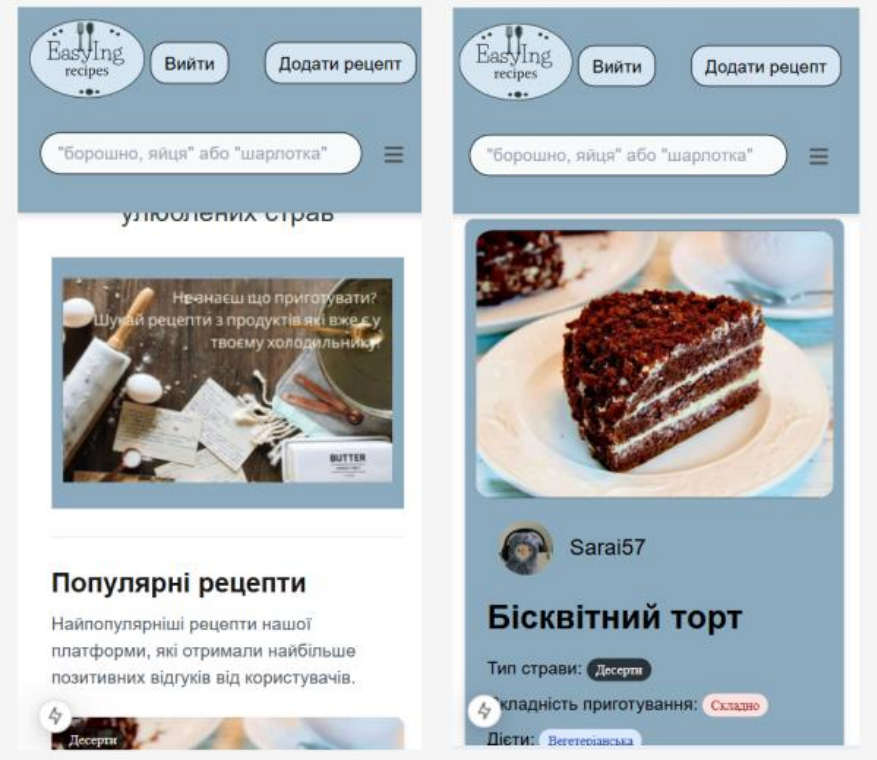


Рисунок Б.6 – Вигляд сторінок у мобільній версії

АНОТАЦІЯ

Маркевич К. О. Розробка вебсайту для пошуку кулінарних рецептів за інгредієнтами. Рукопис

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 Комп'ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2025 р.

Актуальність теми полягає у необхідності забезпечення користувачів зручним сервісом для пошуку та організації рецептів, що враховує наявні інгредієнти та особисті вподобання. Такий вебзастосунок сприяє економії часу та ресурсів під час приготування страв, а також підвищує якість харчування завдяки зручному доступу до різноманітних кулінарних рішень.

Метою роботи є розробка вебсайту для пошуку та фільтрації рецептів за інгредієнтами та іншими параметрами, з можливістю перегляду детальної інформації про страви, користувачів та їх відгуки.

Реалізацію здійснено за допомогою фреймворку Next.js, що забезпечує серверний і клієнтський рендеринг, високу продуктивність та SEO-оптимізацію. Компонентну структуру інтерфейсу побудовано з використанням бібліотеки React. Для зберігання й обробки даних застосовано базу даних MongoDB у поєднанні з ORM-бібліотекою Mongoose. Завантаження зображень реалізовано з використанням хмарного сервісу Cloudinary. Функціонал вебсайту включає реєстрацію та автентифікацію користувачів, пошук рецептів за заданими інгредієнтами, додавання власних рецептів, систему рейтингу та відгуків.

У роботі описано архітектуру вебзастосунку, структуру бази даних, принципи взаємодії між клієнтською та серверною частинами, а також реалізацію основних компонентів інтерфейсу. Наведено приклади коду та проаналізовано можливі шляхи подальшого розвитку.

Ключові слова: вебсайт, кулінарні рецепти, фільтрація, пошук за інгредієнтами, інтерфейс користувача, персоналізація, хмарне сховище.