МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Web-застосунок персоналізованих рекомендацій кулінарних рецептів**

Виконав: студент 4 курсу, групи 6.1211-пзс-
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
Є. В. Патей
(ініціали та прізвище)
Керівник доцент А.І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Тетяна КРИТСЬКА
"_01_" березня 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Патею Євгенію Владиславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Web-застосунок персоналізованих рекомендацій кулінарних рецептів

керівник роботи Безверхий Анатолій Ігорович, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від від 27.12.2024 р. № 2029-с

2. Строк подання студентом кваліфікаційної роботи 02.06.2025
3. Вихідні дані бакалаврської роботи
   - комплект нормативних документів ;
   - технічне завдання до роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   - огляд та збір літератури стосовно теми кваліфікаційної роботи;
   - огляд та аналіз існуючих рішень та аналогів;
   - дослідження проблеми порівняння цін та пропозицій з різних джерел;
   - створення програмного продукту та його опис;
   - перелік вимог для роботи програми;
   - розробка висновків та пропозицій.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	05.05-06.05.25	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	06.05-07.05.25	виконано
3	Аналіз існуючих аналогів	07.05-08.05.25	виконано
4	Розробка структури проекту	08.05-09.05.25	виконано
5	Узгодження подальших дій з науковим керівником	09.05-10.05.25	виконано
6	Налаштування інтеграції вільного API	10.05-11.05.25	виконано
7	Розробка алгоритмів для представлення веб-сторінки	11.05-15.05.25	виконано
8	Розробка фронтенду сторінки	15.05-16.05.25	виконано
9	Розробка функції пошуку рецептів з відображенням результатів	16.05-17.05.25	виконано
10	Розробка фронтенду та бекенду сторінки з рецептами	17.05-18.05.25	виконано
11	Оновлення дизайну веб застосунку	18.05-20.05.25	виконано
12	Покращення взаємодії з вільнім API	20.05-22.05.25	виконано
13	Реалізація підбору рецептів за вподобанням користувача	22.05-01.06.25	виконано
14	Остаточна перевірка працездатності всіх складових	01.06-08.06.25	виконано
15	Створення презентації	08.06-09.06.25	виконано
16	Оформлення звіту	09.06-14.06.25	виконано

Студент _____ Є.В. Патей
(підпис) (прізвище та ініціали)

Керівник роботи _____ А.І. Безверхий
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок — 81,

Рисунків — 14,

Джерел — 15.

Патей Є.В. Web-застосунок персоналізованих рекомендацій кулінарних рецептів : кваліфікаційна робота бакалавра спеціальності 121 «Програмне забезпечення систем» / наук. керівник А. І. Безверхий. Запоріжжя : ЗНУ, 2025. 81 с.

У сучасному світі швидкий доступ до корисної інформації про харчування та рецепти стає все більш затребуваним. Персоналізовані рекомендації кулінарних рецептів допомагають користувачам ефективно планувати харчування з урахуванням їхніх вподобань та дієтичних потреб.

Метою проекту є розробка веб-застосунку, який надає персоналізовані рекомендації кулінарних рецептів з урахуванням фільтрів за типом страви, кухнею, дієтою та поживними характеристиками. Основними завданнями є реалізація інтерфейсу для пошуку рецептів, система фільтрації та відображення результатів із можливістю завантаження додаткових рецептів.

В ході роботи проведено аналіз існуючих рішень у сфері рекомендацій кулінарних рецептів, розроблено структуру та дизайн застосунку, реалізовано функціонал пошуку та фільтрації з використанням сучасних веб-технологій. Веб-застосунок забезпечує зручний та інтуїтивно зрозумілий інтерфейс, що сприяє покращенню користувацького досвіду.

Ключові слова: *персоналізовані рекомендації, кулінарні рецепти, веб-застосунок, фільтрація, пошук, інтерфейс користувача*.

ABSTRACT

Pages — 81,

Figures — 14,

Sources — 15.

Patey E.V. Web application for personalised recommendations of culinary recipes: bachelor's thesis in the field of study 121 “Software Systems” / scientific supervisor A.I. Bezverkhyi. Zaporizhzhia: ZNU, 2025. 81 p.

In today's world, quick access to useful information about nutrition and recipes is becoming increasingly in demand. Personalised recommendations for recipes help users to plan their meals effectively, taking into account their preferences and dietary needs.

The aim of the project is to develop a web application that provides personalised recommendations for recipes, taking into account filters such as dish type, cuisine, diet and nutritional characteristics. The main tasks are to implement an interface for searching recipes, a filtering system and displaying results with the ability to download additional recipes.

During the course of the work, an analysis of existing solutions in the field of culinary recipe recommendations was carried out, the structure and design of the application was developed, and search and filtering functionality was implemented using modern web technologies. The web application provides a convenient and intuitive interface that improves the user experience.

Keywords: *personalised recommendations, recipes, web application, filtering, search, user interface*.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 11
1.1 Визначення аналітичного сервісу 11
1.2 Огляд літературних джерел 13
1.3 Аналіз програмних продуктів-аналогів 14
1.4 Постановка завдання 19
2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ 22
2.1 Аналіз сучасних систем управління базами даних 22
2.2 Аналіз сучасних технологій для створення серверної частини 24
2.3 Аналіз вимог та очікування користувачів сервісу з кулінарних рецептів 26
2.4 Огляд інших технологій 29
2.5 Висновки до розділу 2 31
3 РОЗРОБКА АНАЛІТИЧНОГО СЕРВІСУ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ КУЛІНАРНИХ РЕЦЕПТІВ 33
3.1 Опис предметної області 33
3.2 Вимоги до системи 34
3.3 Архітектура системи 35
3.3.1 Основні складові системи 36
3.3.2 Огляд бази даних 37
3.3.3 Огляд серверної частини 40
3.3.4 Огляд клієнтської частини 41
3.4 Розробка бази даних 44
3.5 Програмна реалізація клєнтської частини 46
3.5.1 Розробка авторизації та керування інтерфейсом користувача 46
3.5.2 Розробка клієнтського модуля відображення рецепту та додавання

до обраного 51
3.5.3 Розробка клієнтської логіки пошуку рецептів..... 55
3.5.4 Розробка клієнтської логіки авторизації та керування інтерфейсом користувача..... 58
3.6 Програмна реалізація серверної частини..... 65
3.6.1 Розробка серверної логіки авторизації користувачів 65
3.6.2 Розробка серверної логіки авторизації користувачів 68
3.6.2 Розробка серверного проксі для роботи з API Spoonacular 71
3.7 Розробка дизайну застосунку..... 72
3.8 Апробація веб-застосунку 76
3.9 Висновки до розділу 3 77
ВИСНОВКИ..... 79
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ..... 80

ВСТУП

Актуальність теми

Сучасні технології значно полегшують процес приготування їжі, надаючи доступ до безлічі рецептів, однак вибір потрібного рецепта може бути складним, особливо коли користувач має певні дієтичні обмеження, особисті вподобання або бажає приготувати страву з конкретних інгредієнтів. Кулінарні веб-сервіси набувають все більшої популярності, оскільки дозволяють швидко знаходити рецепти за особистими вподобаннями, дієтичними обмеженнями або наявними інгредієнтами. Проте більшість існуючих рішень є загальними і не враховують індивідуальні потреби користувачів, що створює попит на персоналізовані системи рекомендацій.

Персоналізовані рекомендації в кулінарній сфері сприяють не лише економії часу, а й покращенню харчових звичок. За допомогою сучасних веб-технологій можна створити інструмент, що адаптується до уподобань користувача, пропонуючи найбільш відповідні рецепти. Такий підхід підвищує зручність користування та якість взаємодії з системою.

Застосування інформаційних технологій у поєднанні з API кулінарних баз даних дає можливість реалізувати функціональний, адаптивний та ефективний веб-застосунок для кожного користувача.

Мета дослідження

Розробити веб-застосунок, який здійснює персоналізований пошук та рекомендацію кулінарних рецептів, з урахуванням уподобань користувача та фільтрів (тип страви, кухня, дієта, калорійність тощо).

Завдання дослідження

- Проаналізувати існуючі кулінарні сервіси та технології рекомендацій.
- Розробити структуру веб-застосунку для рекомендації рецептів.
- Реалізувати інтерфейс з пошуком та фільтрами.
- Інтегрувати API для отримання рецептів у реальному часі.

- Реалізувати систему пагінації та динамічного завантаження рецептів.
- Забезпечити адаптивність і зручність використання інтерфейсу.

Об’єкт дослідження

Процес розробки веб-застосунку для персоналізованого підбору кулінарних рецептів.

Предмет дослідження

Методи, технології та інструменти створення веб-застосунків із функціями персоналізованого пошуку і рекомендацій.

Методи дослідження

Теоретичні: аналіз існуючих кулінарних платформ, підходів до реалізації рекомендаційних систем.

Практичні: розробка прототипу веб-застосунку з використанням HTML, CSS, JavaScript, Node.js, Express, Spoonacular API.

Практичне значення одержаних результатів

Створений веб-застосунок дозволяє користувачам ефективно знаходити рецепти, що відповідають їхнім вподобанням, покращуючи кулінарні навички, спрощуючи процес приготування їжі та підвищуючи якість харчування. Така система може бути основою для подальшого розвитку — наприклад, із можливістю збереження обраних страв, створення меню на тиждень або відстеження харчової цінності страв.

Глосарій

Веб-застосунок — програмне забезпечення, яке працює через веб-браузер, забезпечуючи користувачам доступ до функцій і сервісів без необхідності встановлення на комп’ютер.

Кулінарний рецепт — структурована інформація, що містить інгредієнти, кількість, кроки приготування страви, а також додаткові дані, як-от калорійність, час приготування тощо.

Персоналізована рекомендація — автоматично згенерована порада або

підбір варіантів (у цьому випадку рецептів), яка враховує особисті вподобання, потреби або історію вибору користувача.

Інтерфейс користувача (UI) — набір візуальних елементів, з якими взаємодіє користувач (кнопки, поля вводу, меню, результати пошуку тощо).

Бекенд — серверна частина веб-застосунку, що відповідає за обробку запитів користувача, взаємодію з API, обробку даних і передачу результатів на фронтенд.

Фронтенд — клієнтська частина веб-застосунку, яка відповідає за відображення інформації, збирання введених даних та взаємодію користувача із системою.

API (Application Programming Interface) — інтерфейс для взаємодії між застосунками, який дозволяє отримувати або надсилати дані з зовнішніх сервісів. У проекті використовується Spoonacular API для отримання рецептів.

Пошук — функціональність веб-застосунку, що дозволяє користувачеві вводити запит (наприклад, назву інгредієнта чи страви) та отримувати відповідні рецепти.

Користувач — особа, яка взаємодіє з веб-застосунком з метою пошуку, перегляду або збереження кулінарних рецептів.

Адаптивний дизайн — підхід до розробки інтерфейсу, що забезпечує коректне відображення веб-застосунку на пристроях з різними розмірами екранів (смартфони, планшети, комп’ютери).

JavaScript — мова програмування, яка використовується для реалізації інтерактивної поведінки на стороні клієнта у веб-застосунках.

Node.js — середовище виконання JavaScript на сервері, яке використовується для створення бекенд-частини застосунку.

Express.js — веб-фреймворк для Node.js, що спрощує створення серверних частин застосунків і обробку HTTP-запитів.

Spoonacular API — зовнішній інтерфейс, який надає доступ до великої бази кулінарних рецептів, інформації про інгредієнти, харчову цінність тощо.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення аналітичного сервісу

Під аналітичним сервісом розуміють систему, яка виконує збирання, узагальнення, обчислення та подання інформації з метою полегшення процесу прийняття рішень користувачем. У сучасних умовах, коли даних стає дедалі більше, такі сервіси дозволяють користувачам швидко орієнтуватися в потоках інформації та знаходити найдоцільніші варіанти дій.

Аналітична діяльність передбачає збирання, систематизацію, обчислення та осмислення інформації з метою виявлення залежностей, тенденцій і прийняття обґрунтованих рішень. У сучасному цифровому середовищі вона широко застосовується для підвищення ефективності роботи з великим обсягом даних.

Аналітичний веб-сервіс — це програмна система, орієнтована на виявлення найбільш релевантних рішень для користувача шляхом опрацювання значних масивів інформації. Система кулінарних рекомендацій та пошуку рецептів забезпечує інтелектуальний підбір страв на основі заданих користувачем параметрів, як-от інгредієнти, дієтичні обмеження, тип кухні тощо.

Функціональність сервісу базується на автоматизованому зборі даних із зовнішніх джерел (API, відкриті бази кулінарних знань), їх подальшій фільтрації, обробці та підготовці до аналізу (див. рис. 1). Ключовим етапом є перетворення необроблених даних у структуровану форму, придатну для збереження у сховищі та швидкої обробки запитів користувача.

Уся інформація після обробки зберігається у базі даних, що забезпечує високу продуктивність сервісу під час виконання пошуку та рекомендацій. Система дозволяє моментально витягати потрібну інформацію, що критично важливо для забезпечення персоналізованого

досвіду користування.

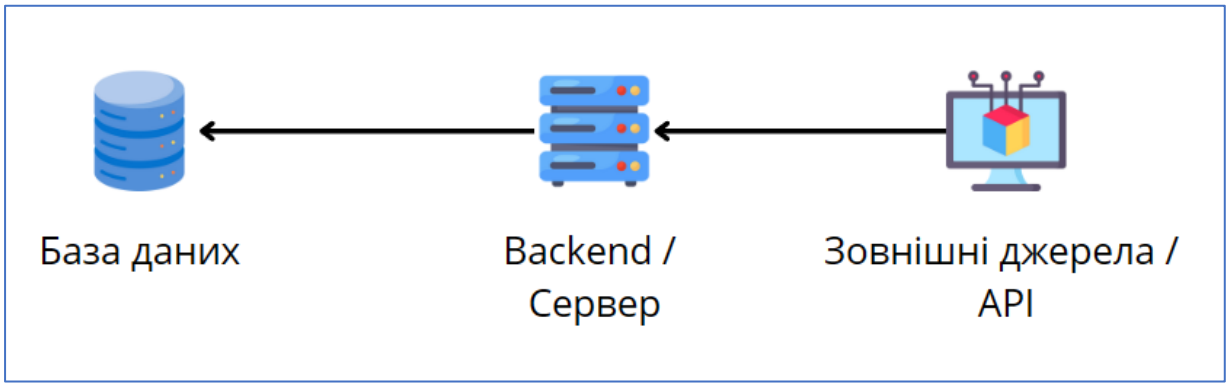


Рисунок *1 – Загальна схема взаємодії серверної частини з джерелами даних*

Для кінцевого користувача взаємодія з сервісом реалізована через інтуїтивно зрозумілий веб-інтерфейс (див. рис. 2), що дозволяє задавати запити, переглядати результати пошуку, застосовувати фільтри та отримувати персоналізовані кулінарні рекомендації.

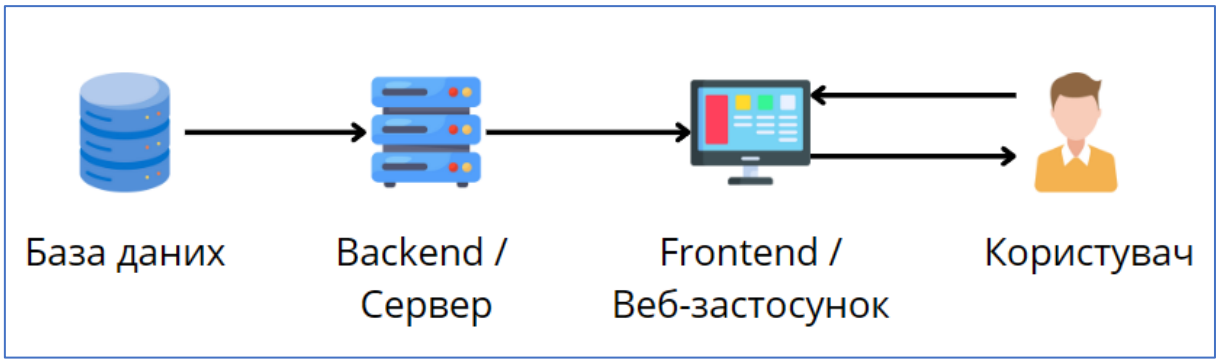


Рисунок *2 – Принцип роботи клієнтської частини веб-застосунку*

Аналітичний інструмент виступає ефективним рішенням для користувачів, які прагнуть швидко знайти відповідні рецепти згідно зі своїми смаками, потребами та доступними інгредієнтами. Його

застосування дозволяє не лише зекономити час на пошук, а й запропонувати більш релевантні варіанти приготування страв.

1.2 Огляд літературних джерел

Під час розробки вебзастосунку для персоналізованого підбору кулінарних рецептів було проведено детальний аналіз літературних джерел, які охоплюють як загальні аспекти веброзробки, так і специфічні технології, що використовувались у проекті. Це дозволило обрати найоптимальніші технічні рішення для реалізації функціоналу авторизації, взаємодії з API, а також збереження даних користувача.

В першу чергу, важливим етапом стала підготовка клієнтської частини, яка базується на JavaScript та DOM-структурі. Було використано документацію MDN, що містить приклади застосування подій, роботи з формами, та асинхронних запитів через fetch [2], [12]. Також широко використовувались практичні приклади та рекомендації з порталу W3Schools [1].

Для реалізації серверної частини з обробкою запитів та авторизацією, було розглянуто документацію PHP [4], [5], зокрема, можливості використання cURL для взаємодії з зовнішніми API та методи захисту паролів через функцію password_hash [11]. У зв’язку з цим, велика увага приділялась реалізації безпечної реєстрації та входу користувача.

Окремо було досліджено документацію до API Spoonacular, який використовується у проекті для отримання кулінарних рецептів [3]. У рамках забезпечення безпеки доступу до API ключа, було реалізовано власний проксі-сервер, про що йдеться у практичних матеріалах з RESTful-архітектури [15].

З метою покращення інтерфейсу користувача, були розглянуті основні принципи UX/UI-дизайну, зокрема рекомендації Nielsen Norman Group [13], а також адаптивні компоненти Bootstrap для побудови модальних вікон [8].

Для збереження даних облікових записів та обраних рецептів

використовувалась база даних MySQL, з детальним вивченням її можливостей через офіційне керівництво [6].

Також були розглянуті матеріали щодо побудови клієнт-серверної архітектури та розгортання проектів на середовищі через XAMPP [14].

Отже, літературні джерела, використані в процесі роботи, надали необхідні теоретичні та практичні знання для ефективної реалізації проекту. Вони охоплюють як загальні підходи до веброзробки, так і конкретні аспекти роботи з API, безпекою, базами даних та користувацьким інтерфейсом.

1.3 Аналіз програмних продуктів-аналогів

Різноманітність рецептів та кулінарних рекомендацій є дуже поширеною темою яка зацікавить кожного. Існує чимало веб-ресурсів і мобільних застосунків, які надають рецептурний контент, дозволяють зберігати улюблені страви або формувати меню. Серед них є як прості сайти з кулінарними статтями, так і складні системи з елементами штучного інтелекту, що здатні пропонувати персоналізовані страви. Розглянемо декілька найбільш відомих аналогів.

Eat This Much – це онлайн-платформа і мобільний застосунок для створення персоналізованих планів харчування. Користувач задає кількість калорій, тип дієти (наприклад, кето, веганська, палео) і платформа формує повний раціон на день або тиждень (див. Рис.3).

Особливості:

- Сильний акцент на харчову цінність і дієтичні плани;
- Автоматичне складання меню;
- Можливість налаштування бюджету на день;
- Генерація списку покупок;
- Менш гнучка база рецептів, обмежена специфікою дієт.

Eat This Much орієнтований більше на дієтичне харчування, ніж на загальну кулінарну зручність. Проте механізм персонального планування їжі можна вважати ефективним аналогом для реалізації схожої функції.

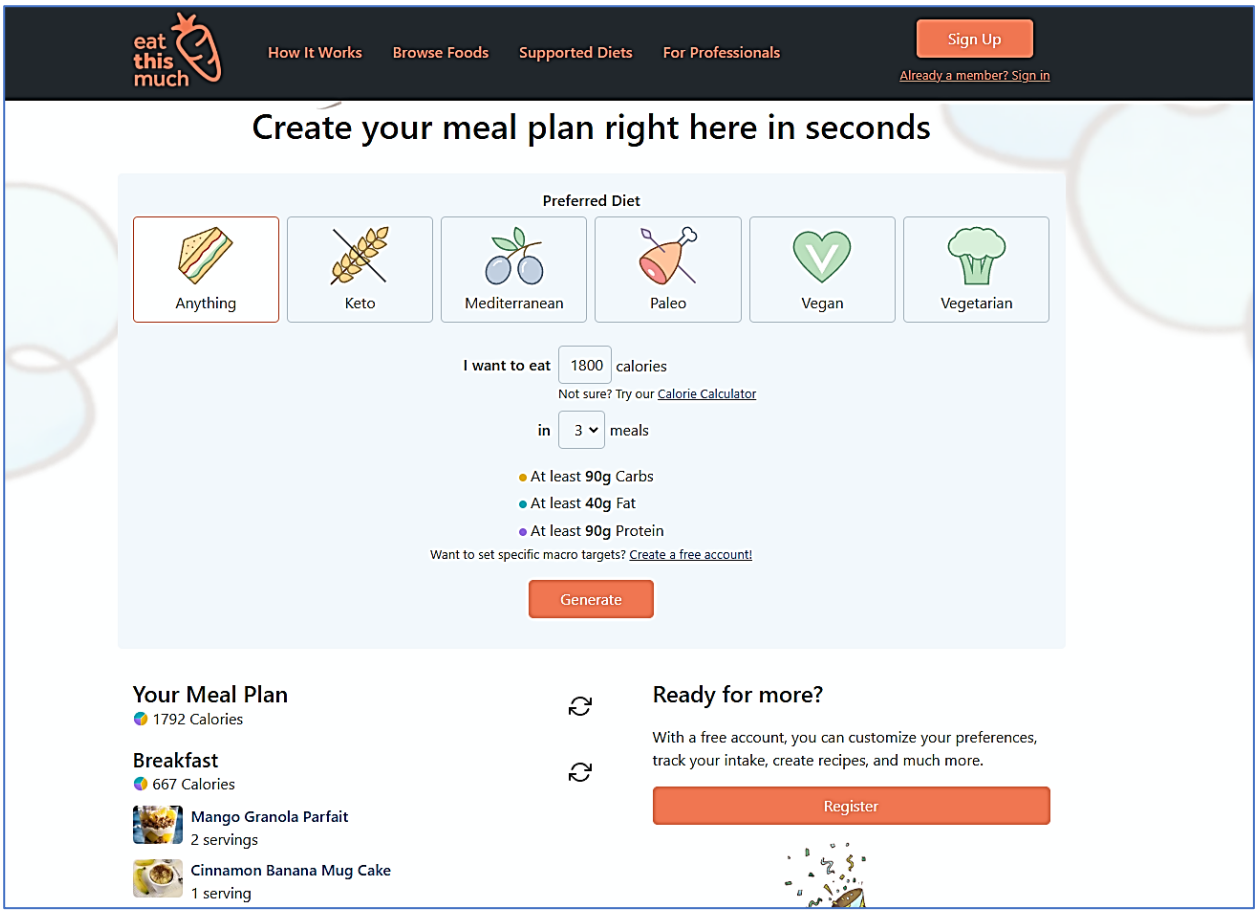


Рисунок *3 – Сторінка з планування харчування в Eat This Much*

Tasty - мультимедійний кулінарний застосунок від BuzzFeed, орієнтований на молодіжну аудиторію. Він пропонує рецепти з короткими та яскравими відеоінструкціями (див. Рис.4).

Особливості:

- Сильна візуальна складова — рецепти супроводжуються якісними відео;
- Можливість створити «кухарську книгу» зі збережених страв;
- Обмежене налаштування персональних рекомендацій;

- Відсутність глибокого аналізу наявних інгредієнтів;
- Вибір рецептів за категоріями (вегетаріанські, святкові, швидкі тощо).

Хоч сервіс і має зручний інтерфейс, він не дає можливості враховувати поточні продукти вдома чи формувати динамічні меню.

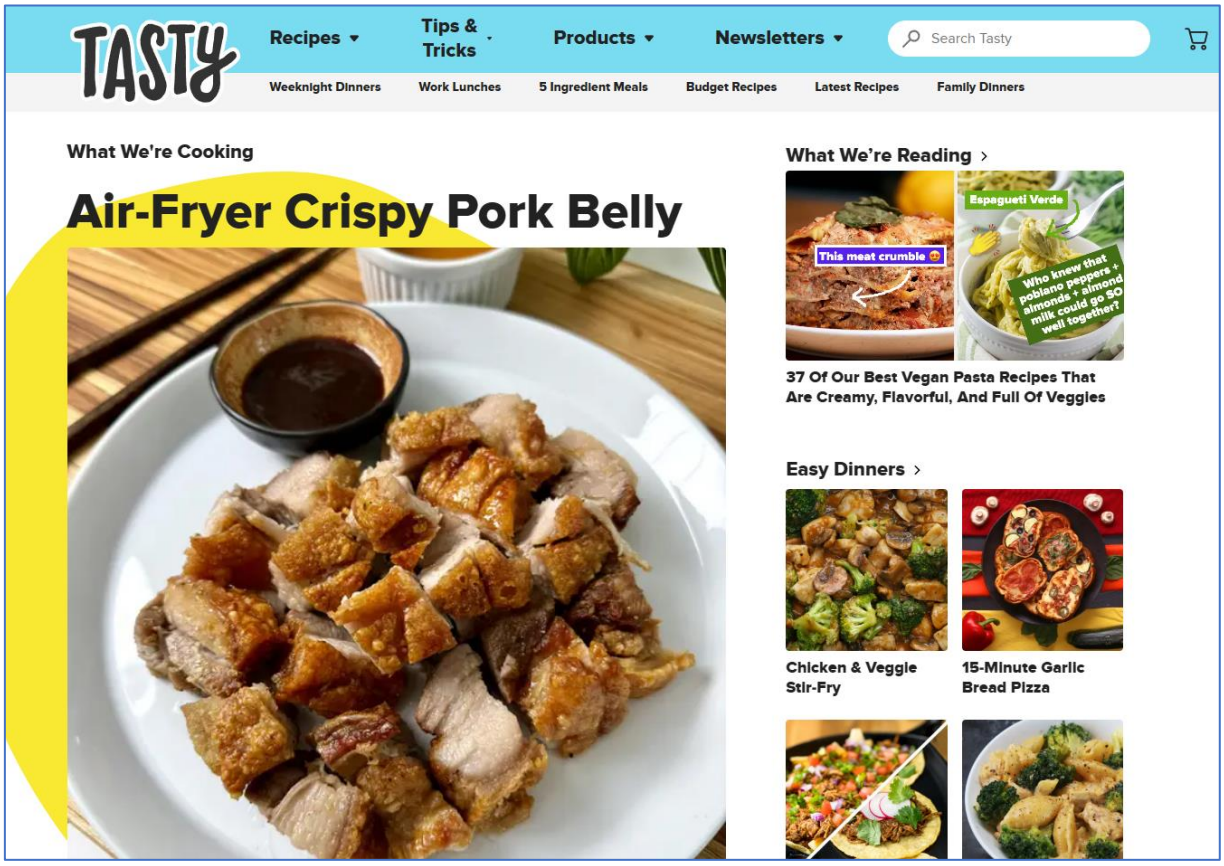


Рисунок *4 – Інтерфейс веб-застосунку Tasty*

SuperCook - один із небагатьох сервісів, який пропонує рецепти на основі тих продуктів, що вже є в холодильнику користувача. Користувач вводить або вибирає інгредієнти — система пропонує рецепти з урахуванням лише наявних продуктів або з мінімальною кількістю додаткових (див. Рис.5).

Переваги:

- Вибір рецептів з продуктів що користувач зараз має;
- Фільтрація за типом кухні, складністю, часом приготування;
- Можливість додати продукти в обране;
- Проста візуальна подача та мінімалістичний дизайн;
- Відсутність реальної персоналізації за смаками чи дієтичними особливостями.

SuperCook близький до мети проекту, але не має адаптивної системи рекомендацій і не дозволяє створювати персональні меню на основі звичок користувача.

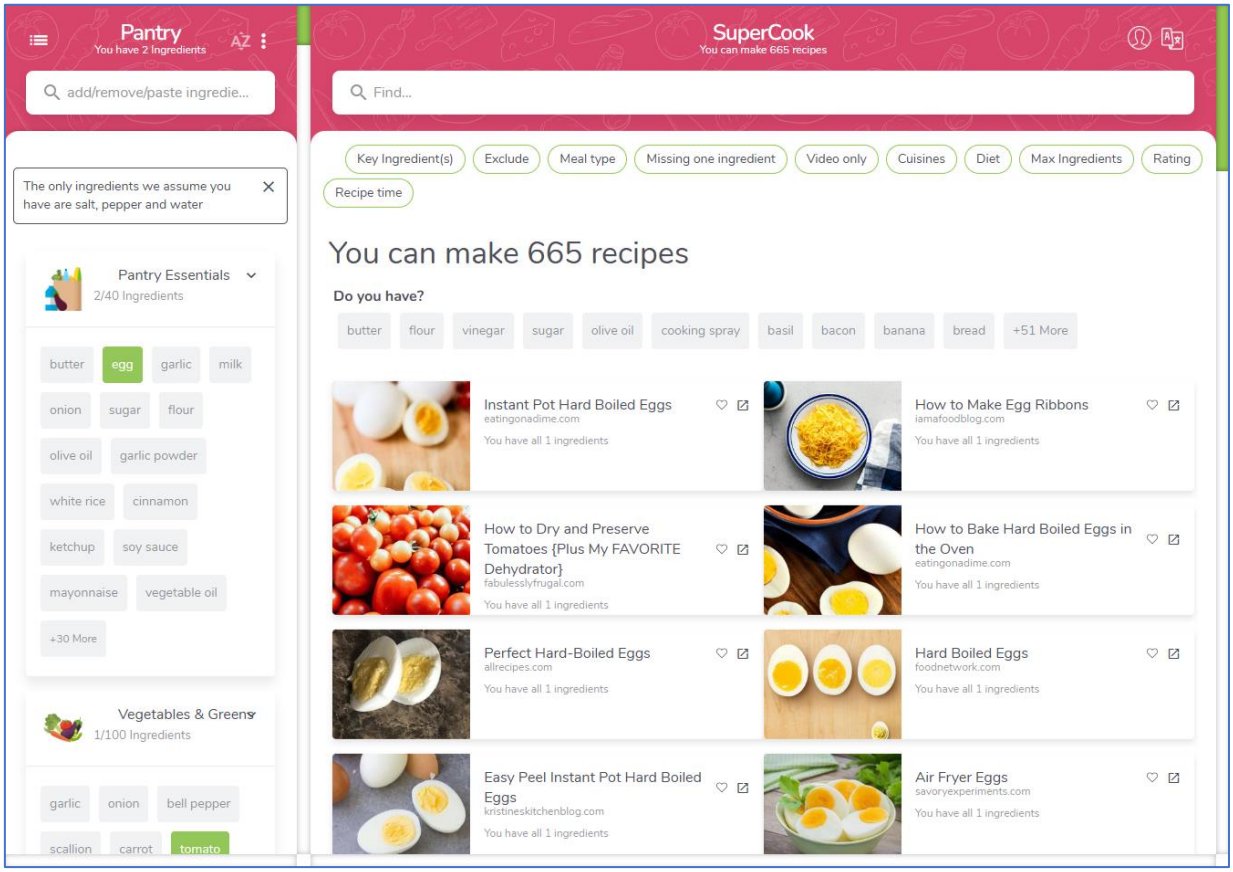


Рисунок *5 – Інтерфейс SuperCook: підбір за наявними інгредієнтами*

Проведений аналіз показав, що всі розглянуті сервіси мають певні переваги — чи то зручний інтерфейс, чи персоналізовані рекомендації, чи генерацію меню. Проте жоден із них не реалізує всі ці функції одночасно,

особливо якщо враховувати локалізацію, інтеграцію з продуктовими магазинами або наявністю інгредієнтів у реальному часі.

Функціональні можливості кулінарного веб-застосунку, які слід виділити:

- **Пошук рецептів.** Користувачі мають змогу здійснювати пошук кулінарних рецептів за ключовими словами, назвою страви, типом кухні (італійська, американська, азіатська тощо), категорією (сніданки, обіди, десерти) або харчовими цінностями.
- **Фільтрація та сортування.** Система дозволяє фільтрувати результати за часом приготування, складністю, калорійністю, дієтичними обмеженнями (веганство, безглютенова дієта тощо).
- **Індивідуальні рекомендації.** На основі вподобань користувача, історії його взаємодії, алергій та збережених інгредієнтів система генерує персоналізовану стрічку рекомендованих страв.
- **Рецепти з наявних інгредієнтів.** Користувач може вказати продукти, які є в нього вдома, і система запропонує список страв, які можна приготувати з цих інгредієнтів, або з мінімальною кількістю додаткових покупок.
- **Збереження рецептів.** Користувачі можуть додавати рецепти у “Бажане”, щоб при нагоді скористатися потрібним рецептом у сервісі.
- **Реєстрація та особистий кабінет.** Система дозволяє створити обліковий запис, в якому зберігаються вподобання, історія приготувань, налаштування харчування, обмеження та дані для генерації рекомендацій.
- **Зворотний зв'язок.** Користувачі можуть залишати оцінки та коментарі, ділитися враженнями до рецептів.

Інтерфейс веб-застосунку наведено на рисунку 6.

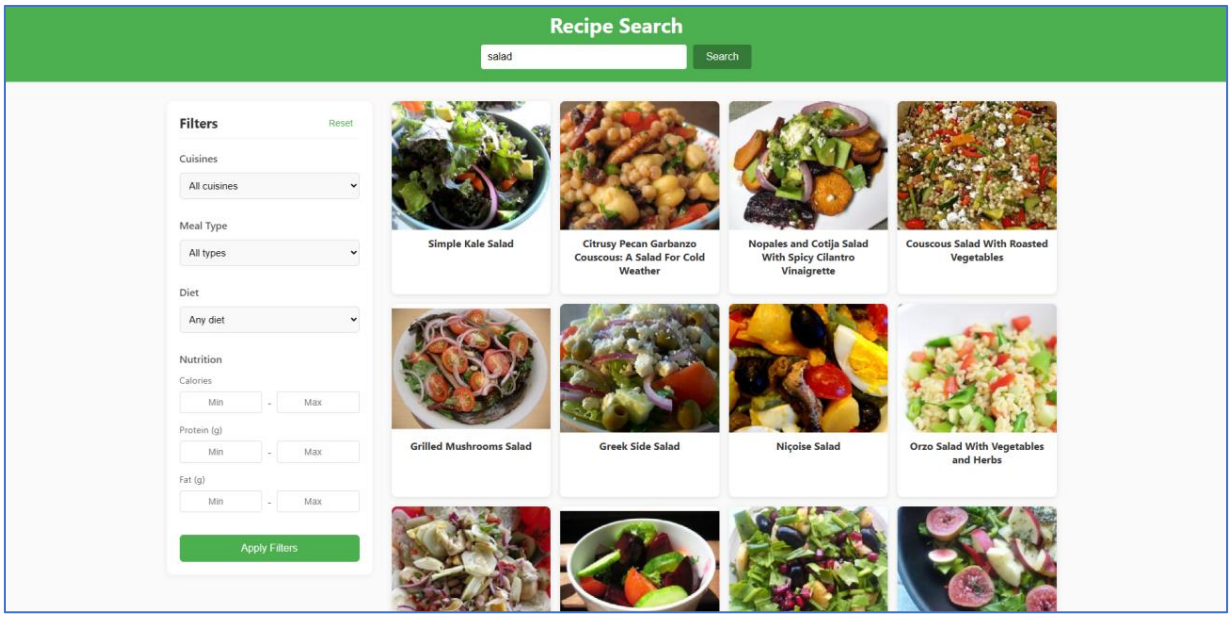


Рисунок *6 – Сторінка з пошуком та фільтру кулінарних рецептів*

Таким чином, розробка веб-застосунку для персоналізованих кулінарних рекомендацій, який поєднує аналіз наявних продуктів, смаки користувача, дієтичні обмеження та гнучкий інтерфейс — є актуальною і може зайняти конкурентну нішу.

1.4 Постановка завдання

Метою кваліфікаційної роботи є розробка веб-застосунку для пошуку та персоналізованих рекомендацій кулінарних рецептів. Завдяки використанню сучасних веб-технологій та алгоритмів обробки даних, система має допомогти користувачам ефективно планувати свій раціон, знаходити нові страви відповідно до власних уподобань, а також оптимізувати використання продуктів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Вивчити існуючі рішення у сфері кулінарних сервісів і визначити їх переваги та недоліки.
2. Розробити структуру бази даних для зберігання інформації про рецепти,

інгредієнти, категорії, уподобання користувача та інші метадані.

- Обрати відповідну СУБД;
- Сформувати структуру таблиць та визначити ключові зв’язки між ними;
- Забезпечити підтримку масштабування та швидкий доступ до даних.

3. Розробити механізм збору або отримання даних про рецепти:

- Обрати API для завантаження рецептів;
- Реалізувати механізм регулярного оновлення та обробки інформації;

4. Реалізувати серверну частину (бекенд) веб-застосунку:

- Реалізувати обробку запитів на пошук, фільтрацію, персоналізовані рекомендації;
- Налаштувати авторизацію та автентифікацію користувачів.

5. Розробити клієнтську частину (фронтенд) застосунку:

- Створити інтуїтивно зрозумілий та адаптивний інтерфейс користувача;
- Зверстати основні сторінки: головну, пошуку, перегляду рецепту, особистого кабінету;
- Зв’язати інтерфейс із серверною частиною.

6. Реалізувати функціональність персоналізованих рекомендацій:

- Розробити механізм аналізу уподобань користувача;
- Враховувати історію переглядів, збережені рецепти, дієтичні обмеження;
- Побудувати систему, що пропонує рецепти на основі наявних інгредієнтів.

Поставлене завдання передбачає створення зручного, гнучкого та інформативного веб-застосунку, який дозволить користувачам швидко

знаходити рецепти відповідно до власних смакових уподобань, доступних інгредієнтів та дієтичних обмежень. Такий сервіс має підвищити якість життя користувачів, зекономити час на пошук рецептів і планування харчування, а також популяризувати здорове та різноманітне харчування.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз сучасних систем управління базами даних

Організація та збереження інформації є ключовим аспектом у функціонуванні будь-якого програмного продукту. Щоб забезпечити ефективну роботу з великими обсягами даних, необхідно обрати відповідне програмне рішення для керування базою даних. Такі системи дозволяють впорядковувати інформацію, працювати з нею у зручному вигляді, а також забезпечувати швидкий доступ до неї за потреби.

Системи управління базами даних (СУБД) реалізують набір інструментів для додавання, оновлення, вилучення та обробки структурованих або напівструктурованих даних. СУБД поділяються на різні типи залежно від того, яку логічну модель даних вони підтримують. Найбільш поширеними є реляційні СУБД, які ґрунтуються на табличному поданні даних. Також активно розвиваються документоорієнтовані, графові та інші типи СУБД, що застосовуються у специфічних сферах — наприклад, для роботи з неструктурованими або взаємопов’язаними даними.

При визначенні найдоцільнішого інструменту для проекту важливо враховувати такі характеристики:

- **Можливості системи.** СУБД має забезпечувати повний набір функцій для реалізації всіх необхідних сценаріїв: реєстрація користувачів, обробка запитів, збереження вподобаних елементів тощо.
- **Швидкодія.** Важливим аспектом є час обробки запитів та загальна продуктивність при роботі з різним обсягом даних. Це критично при розробці систем із багатьма одночасними користувачами.
- **Можливість масштабування.** З огляду на потенційне зростання обсягу даних та навантаження, СУБД повинна підтримувати горизонтальне або вертикальне масштабування без втрати ефективності.

- **Стійкість до збоїв.** Необхідна висока стабільність у роботі та мінімізація ризиків втрати або пошкодження даних.
- **Вартість володіння.** У тому числі враховується наявність безкоштовної ліцензії, витрати на технічну підтримку, а також необхідність закупівлі додаткового програмного забезпечення.
- **Доступність документації та підтримки.** Активна спільнота розробників, наявність навчальних матеріалів та рішень типових проблем значно полегшують процес інтеграції СУБД у проект.

Порівняння популярних СУБД:

- MySQL. Це одна з найвідоміших відкритих реляційних СУБД. Вона демонструє високу продуктивність, зручність у використанні та сумісність з більшістю сучасних платформ. Її функціональність дозволяє реалізовувати як прості, так і складні інформаційні структури.
- PostgreSQL. Це високопродуктивна система з відкритим кодом, яка підтримує роботу з великими обсягами даних, транзакціями, а також забезпечує можливості для розширення функцій. Система також підтримує просторові дані та складні запити.
- MongoDB. Ця документоорієнтована СУБД належить до типу NoSQL. Вона дозволяє працювати з неструктурованими або слабо структурованими даними, забезпечуючи гнучкість та зручність при моделюванні динамічних структур. Підтримка масштабування по горизонталі та висока доступність — її головні переваги.
- Microsoft SQL Server. Комерційна розробка від Microsoft, яка відзначається широким набором функцій, серед яких — підтримка складних транзакцій, інтеграція з хмарними рішеннями та аналітичні інструменти. Часто використовується у корпоративному середовищі.
- Oracle Database. Ця платформа є прикладом потужного і надійного рішення для підприємств. Вона підтримує складну архітектуру, масштабування, резервне копіювання та інструменти для аналітики,

що робить її ефективною у роботі з критично важливими даними.

Висновок

Для збереження користувацьких даних, обліку вподобаних рецептів та забезпечення стабільної роботи сервісу було прийнято рішення використовувати MySQL як основну систему керування базою даних. Такий вибір зумовлений кількома практичними міркуваннями, що відповідають поставленим вимогам до інформаційної системи. Такий вибір пояснюється наступними чинниками:

- **Простота інтеграції.** MySQL без проблем взаємодіє з PHP, який використовується для серверної логіки проекту.
- **Швидкість.** СУБД забезпечує високу продуктивність при роботі зі структурованими запитами до користувацьких даних та вподобаних рецептів.
- **Достатність функціоналу.** У проекті використовуються типові операції — зберігання користувачів, автентифікація, збереження рецептів у вибране — які повністю підтримуються MySQL.
- **Нульова вартість.** Система є відкритою і не потребує оплати за використання, що відповідає вимогам доступного розгортання на локальному сервері (наприклад, XAMPP).
- **Наявність великої кількості ресурсів.** У випадку виникнення проблем легко знайти рішення або поради в інтернеті, завдяки широкій спільноті розробників.

З урахуванням усіх зазначених переваг, MySQL виявився найбільш придатною платформою для реалізації завдань проекту, поєднуючи простоту в роботі, ефективність і широку підтримку.

2.2 Аналіз сучасних технологій для створення серверної частини

Для повноцінного функціонування веб-застосунку необхідно не лише створити зручний інтерфейс для користувача, а й розробити серверну

частину, яка буде опрацьовувати запити, взаємодіяти з базою даних та забезпечувати безперебійну логіку роботи системи.

У моєму проекті серверна складова виконує кілька ключових задач:

- **Передача даних між фронтендом і базою**. Сервер повинен забезпечити прийом і обробку запитів, пов’язаних із збереженням облікових даних, зчитуванням обраних рецептів, фільтрацією за заданими критеріями тощо.
- **Зв’язок з базою даних**. Надійна комунікація з СУБД є обов’язковою, зокрема для авторизації, реєстрації, додавання та зчитування улюблених рецептів.
- **Захист інформації та контроль доступу**. Серверна частина відповідає за реалізацію перевірки сесій користувачів, шифрування паролів і обмеження доступу до особистих даних.

Після аналізу різних варіантів, моїм рішенням стало використання **Node.js** для реалізації серверної частини. Це середовище виконання JavaScript-коду на стороні сервера, яке надає низку переваг, зокрема:

- **Висока швидкість обробки запитів**. Завдяки неблокуючій архітектурі Node.js дозволяє обслуговувати декілька запитів одночасно без затримок, що є критичним для інтерактивних веб-застосунків.
- **Уніфікація мови програмування**. Оскільки клієнтська частина мого проекту також створена на JavaScript, це дозволило уникнути перемикання між різними мовами на різних рівнях розробки, що суттєво пришвидшило роботу над кодом.
- **Асинхронна модель**. Перевага асинхронності в тому, що сервер не «зависає» на одному запиті, а продовжує працювати з іншими, навіть якщо певна операція займає час, наприклад, взаємодія з базою або зовнішнім API.
- **Гнучка екосистема**. Node.js підтримує велику кількість модулів, які

легко підключаються до проекту — наприклад, модулі для шифрування, обробки запитів, логування чи роботи з базами даних.

Для організації логіки маршрутизації, обробки запитів і створення API я використав фреймворк **Express.js**. Він дозволяє швидко та структуровано реалізувати серверну частину додатку. Завдяки простому синтаксису та великій кількості готових рішень, Express ідеально підходить для студентських проектів, де важливо швидко протестувати функціонал і адаптувати код під змінні вимоги.

До переваг Express можна віднести:

- Зручне створення маршрутів (endpoint'ів) для API;
- Просту інтеграцію з middleware для обробки запитів;
- Можливість розділяти логіку на модулі, що спрощує підтримку проекту.

Під час розробки реалізовано окремий серверний файл server.js, який відповідає за маршрути API, обробку запитів на додавання в «Улюблене», отримання персоналізованих рекомендацій і автентифікацію користувача. З'єднання з базою MySQL здійснюється через Node.js.

2.3 Аналіз вимог та очікування користувачів сервісу з кулінарних рецептів

При створенні інформаційного веб-сервісу важливо враховувати реальні потреби потенційних користувачів. Головною метою даного проекту є розробка платформи для пошуку кулінарних рецептів, яка не тільки надає доступ до великої бази страв, але й персоналізує досвід кожного користувача.

Основні очікування користувачів

Аналіз поведінки типових користувачів дозволив сформулювати базові очікування:

- Швидкий та зручний пошук рецептів. Користувачі хочуть мати можливість знайти відповідний рецепт за кілька кліків, використовуючи фільтри (за кухнею, типом страви, дієтою, інгредієнтами тощо).
- Персоналізовані рекомендації. Багато користувачів очікують, що сервіс буде «пам’ятати» їхні вподобання і пропонувати саме ті рецепти, які найбільш відповідають їхнім смаковим або дієтичним уподобанням.
- Можливість зберігати улюблені рецепти. Функція «Улюблене» є важливою для тих, хто регулярно готує і хоче мати швидкий доступ до перевірених страв.
- Безпека та простота реєстрації/входу. Сучасні користувачі звикли до зручних форм реєстрації, а також хочуть бути впевненими у безпеці своїх особистих даних.
- Адаптивність і зрозумілий інтерфейс. Незалежно від типу пристрою (комп’ютер, планшет, смартфон), інтерфейс має залишатися інтуїтивним і приємним у використанні.

Запропоновані рішення для задоволення потреб користувачів

Для задоволення виявлених вимог та очікувань користувачів планується реалізувати низку функціональних та технічних рішень, що забезпечать комфортну та ефективну взаємодію з сервісом.

1. **Гнучкий пошук рецептів.** Передбачено впровадження системи фільтрації, що дозволятиме користувачеві обирати рецепти за різними критеріями: тип страви, кухня, харчові вподобання, дієтичні обмеження, калорійність, кількість білків, жирів тощо. Це дозволить значно звузити пошук і знайти найбільш релевантні варіанти.

2. **Механізм персоналізації.** Сервіс має враховувати індивідуальні переваги користувача на основі збережених рецептів, історії вибору та взаємодій із платформою. На основі цих даних буде формуватись добірка рекомендацій, яка відповідатиме смаковим уподобанням конкретного користувача.
3. **Можливість зберігати улюблені рецепти.** Для покращення зручності повторного використання рецептів, передбачено функціонал додавання страв до обраного списку. Користувач зможе переглядати, сортувати та видаляти рецепти зі свого особистого списку.
4. **Простий механізм реєстрації та входу.** Щоб користувачі могли зберігати свої дані та отримувати персоналізований досвід, планується створення модального вікна для входу та реєстрації з інтуїтивним інтерфейсом. Вся особиста інформація має зберігатися у зашифрованому вигляді з дотриманням принципів безпеки.
5. **Адаптивний інтерфейс.** Інтерфейс користувача повинен бути адаптованим під різні типи пристроїв — від комп’ютерів до мобільних телефонів. Це дозволить користуватись сервісом у будь-яких умовах — вдома, у дорозі чи в магазині.
6. **Зрозуміла навігація та візуальна структура.** Особлива увага приділятиметься логіці розташування елементів, структурі сторінок та загальному дизайну, щоб користувач легко орієнтувався у сервісі незалежно від рівня досвіду користування веб-додатками.
7. **Безпечне зберігання даних.** При роботі з особистими даними користувачів та їх активністю важливо дотримуватись стандартів безпеки: шифрування паролів, захист від несанкціонованого доступу та належна організація взаємодії з базою даних.

2.4 Огляд інших технологій

У процесі проектування програмного забезпечення використовувалися не лише інструменти, безпосередньо пов’язані з програмуванням, а й ті, що допомогли організувати роботу, створити логічну структуру та візуально представити ключові ідеї й архітектурні рішення.

Lucidchart — це хмарний сервіс для створення діаграм, блок-схем, організаційних структур, логічних карт та інших візуальних матеріалів. Даний інструмент активно застосовувався для формалізації структури бази даних, планування взаємодії компонентів та створення загального бачення архітектури системи. Візуалізація складних процесів значно спростила подальшу реалізацію і допомогла уникнути помилок на етапі кодування (див. Рис.7).

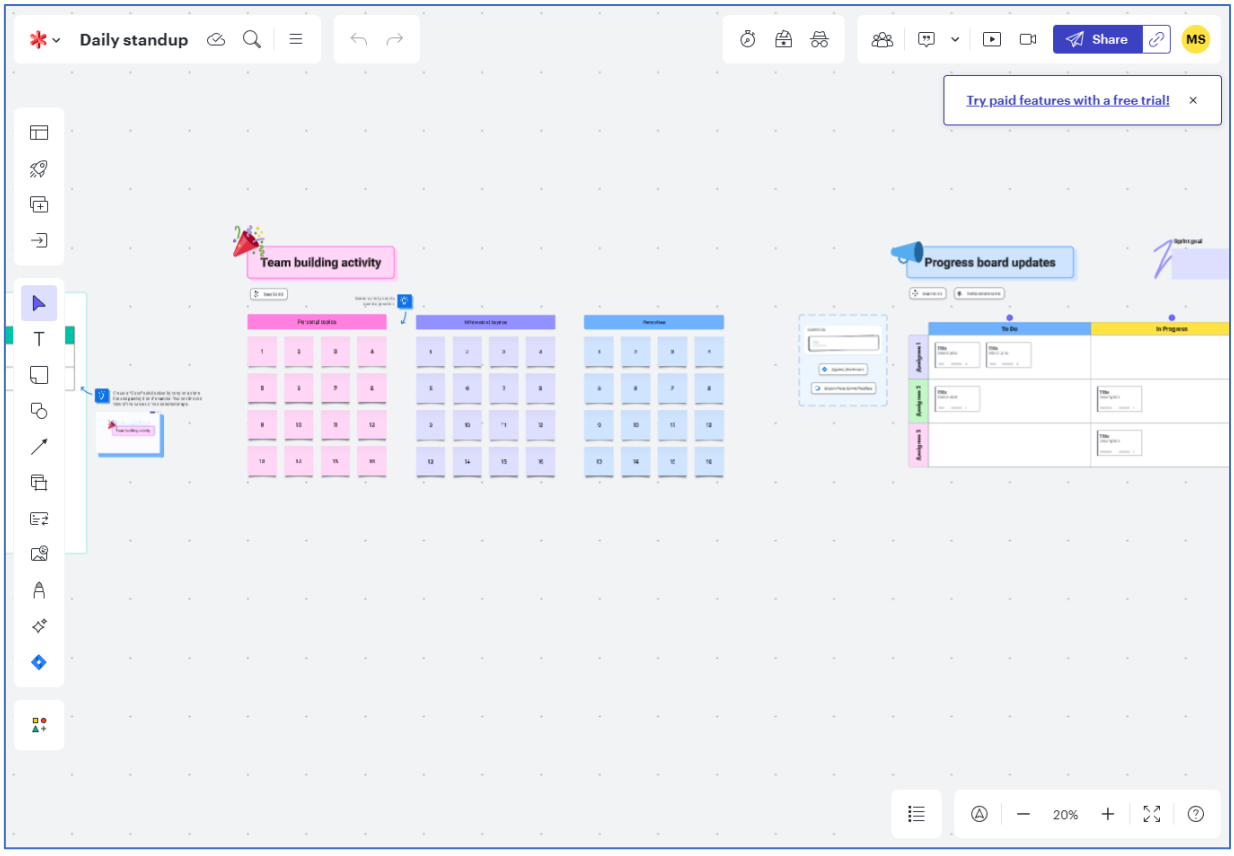


Рисунок *7 – Інтерфейс застосунку Lucidchart.*

Trello — система управління завданнями, що дозволила правильно організувати етапи розробки. Інструмент зручний для розподілу завдань, встановлення дедлайнів та фіксації прогресу. Особливо корисним було використання чеклістів та тегів, які забезпечили чітке розуміння того, на якому етапі знаходиться кожна частина проекту. Це дало змогу працювати послідовно та ефективно планувати час (див. Рис.8).

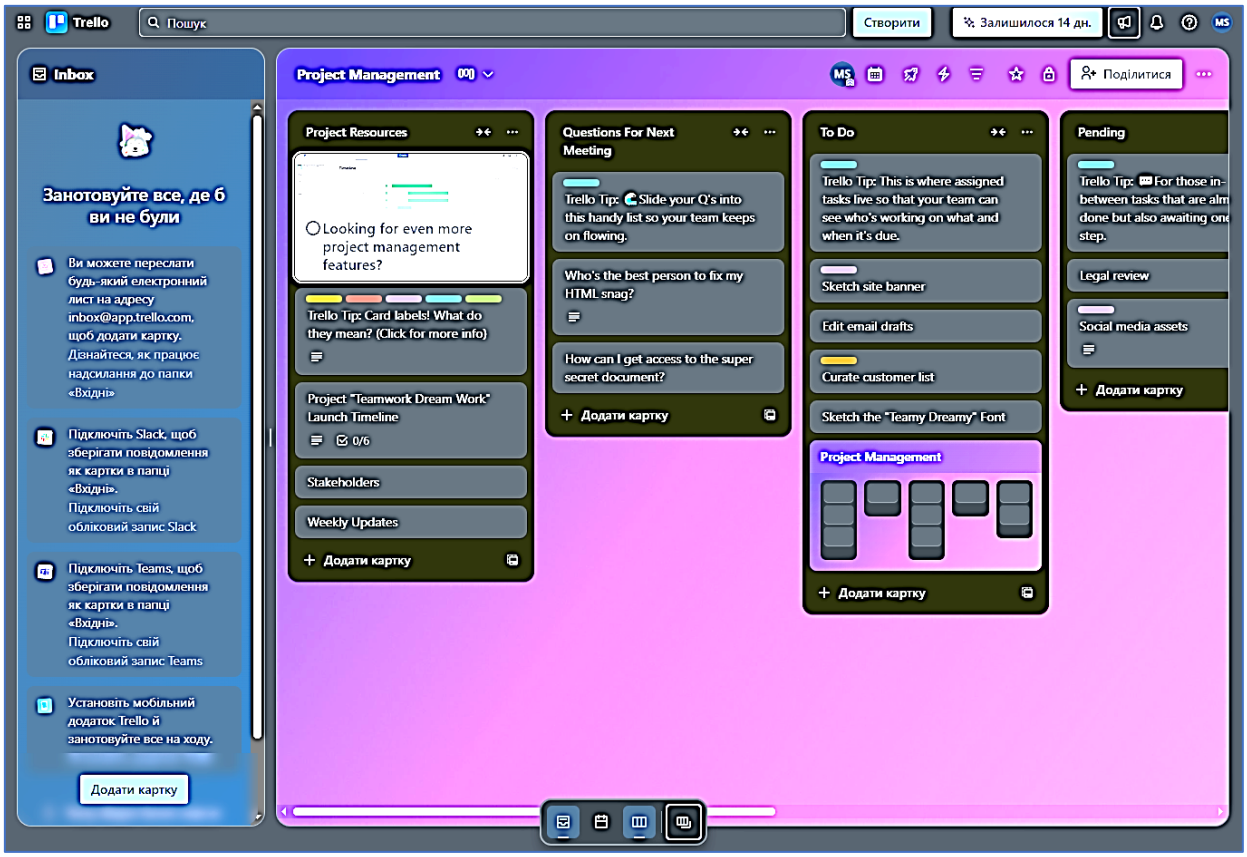


Рисунок *8 – Інтерфейс застосунку Trello.*

Figma — платформа для UI/UX-дизайну, що використовувалась для створення макетів інтерфейсу вебзастосунку. Інструмент дав змогу реалізувати інтерактивні прототипи, перевірити працездатність окремих елементів інтерфейсу ще до написання HTML-розмітки. Серед головних

переваг — можливість командної роботи в режимі реального часу та широка підтримка векторної графіки. Figma стала ключовим засобом для опрацювання дизайну, завдяки чому вдалося створити зручний та візуально привабливий інтерфейс (див. Рис.9).

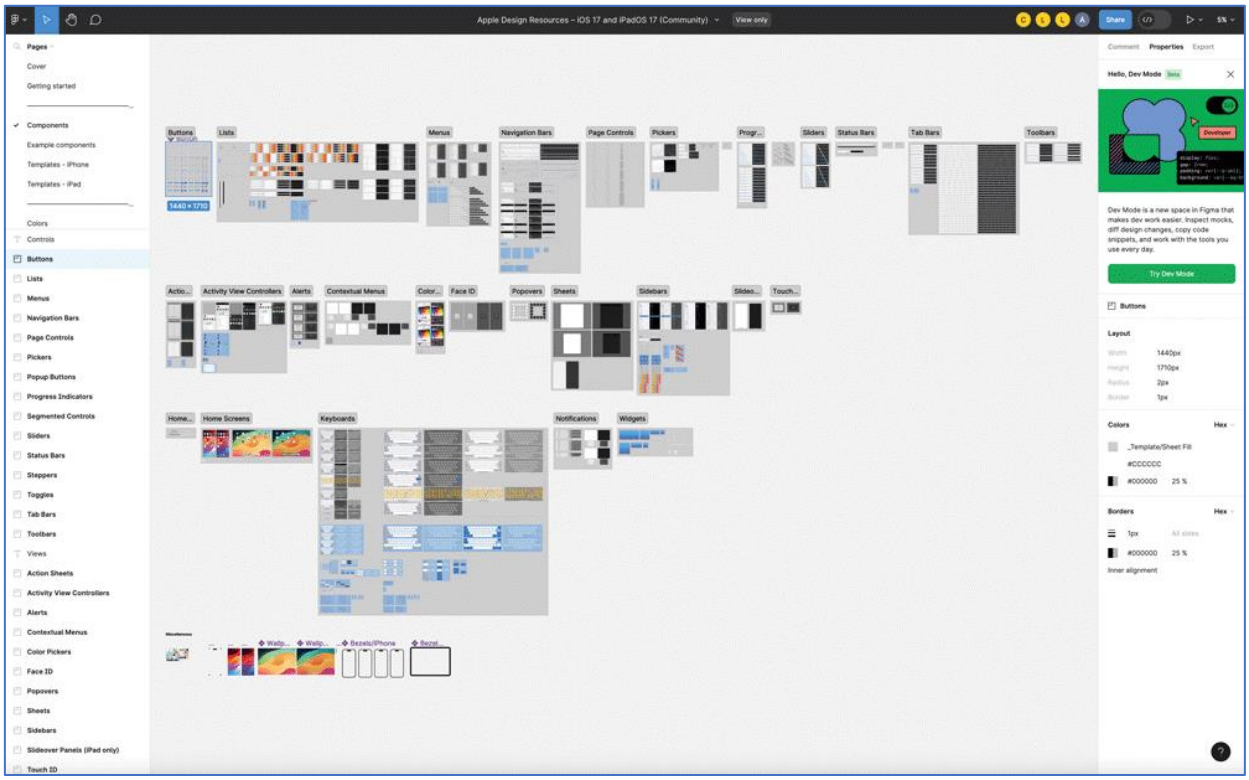


Рисунок 9 – Інтерфейс застосунку Figma

Використання допоміжних технологій дозволило краще спланувати процес розробки, уникнути хаотичної реалізації і досягти більш структурованого та логічного підходу до побудови системи. Ці інструменти сприяли ефективній комунікації, візуалізації задумів і загальному підвищенню якості програмного продукту.

2.5 Висновки до розділу 2

1. Проведено детальний аналіз сучасних систем управління базами даних, серед яких було розглянуто такі популярні рішення як MySQL,

PostgreSQL, SQLite та MongoDB, що дозволило обґрунтовано визначити найдоцільнішу технологію для реалізації вебсервісу.

2. Порівняння функціональних можливостей СУБД продемонструвало, що MySQL оптимально підходить для проекту завдяки своїй простоті у використанні, підтримці реляційної моделі, великій спільноті та ефективності при роботі з середніми обсягами даних.
3. Вивчення сучасних підходів до створення серверної частини показало доцільність використання Node.js у поєднанні з Express як основи для бекенду завдяки асинхронній обробці запитів, високій продуктивності та можливості використовувати одну мову для фронтенду і серверної частини.
4. Проведено аналіз ключових вимог користувачів кулінарного сервісу, серед яких визначальними стали швидкий пошук рецептів, персоналізовані рекомендації, фільтрація за дієтичними уподобаннями, наявність «вибраного» та зручний інтерфейс для мобільних і десктопних пристроїв.
5. У процесі вивчення очікувань цільової аудиторії сформульовано чіткі функціональні вимоги до системи, що забезпечують її практичність, інтуїтивну взаємодію з користувачем та можливість збереження індивідуального досвіду.
6. Окрему увагу приділено огляду допоміжних інструментів, зокрема Lucidchart, Trello та Figma, які не є частиною програмного коду, але значно посприяли плануванню, візуалізації архітектури та створенню дизайну інтерфейсу користувача.
7. Результати аналітичного етапу проекту дозволили сформувати чітку технічну основу, на якій базувалась подальша розробка сервісу, що дало змогу ефективно організувати роботу, уникнути типових помилок та досягти високого рівня якості реалізації.

3 РОЗРОБКА АНАЛІТИЧНОГО СЕРВІСУ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ КУЛІНАРНИХ РЕЦЕПТІВ

3.1 Опис предметної області

Предметна область даного проекту належить до сфери кулінарії та здорового способу життя. Основна ідея полягає у створенні вебсервісу, який допомагає користувачам знаходити, зберігати та персоналізовано підбирати рецепти відповідно до їхніх смаків, харчових уподобань, дієтичних обмежень або наявних інгредієнтів.

У сучасному суспільстві, де все більше людей прагнуть готувати вдома, контролювати своє харчування та експериментувати з новими стравами, існує зростаюча потреба в зручних інструментах для пошуку кулінарних ідей. Особливо важливою є можливість швидко знайти рецепти, які відповідають конкретним запитам користувача — наприклад, безглютенові страви, вегетаріанське меню або низькокалорійні обіди.

Предметна область включає в себе такі аспекти, як тематичний пошук кулінарних рецептів, категоризація за типом страви (сніданки, десерти, супи тощо), за кухнею світу, а також можливість додавання рецептів до «обраного», щоб мати до них швидкий доступ у майбутньому. Крім того, важливою частиною є персоналізація — аналіз вподобань користувача та формування рекомендацій на основі його дій та обраних рецептів.

Головна мета цього сервісу — забезпечити простий та ефективний інструмент для користувачів, який допоможе знаходити рецепти за кілька кліків, адаптувати пропозиції під конкретні потреби та зробити процес приготування їжі більш зручним, організованим і натхненним. У результаті, вебплатформа не лише вирішує побутові задачі користувачів, але й сприяє формуванню корисних звичок, самовираженню через кулінарію та економії часу.

Таким чином, проект охоплює предметну область, що на перетині

гастрономії, IT і цифрової допомоги у повсякденному житті, і відповідає актуальним запитам сучасного користувача.

3.2 Вимоги до системи

Розробка будь-якої програмної системи передбачає формулювання чітких вимог, які охоплюють як функціональні, так і нефункціональні характеристики. Вимоги до системи є надзвичайно важливими, оскільки вони формують основу для проектування, реалізації, тестування та подальшого впровадження програмного продукту. У контексті створення вебсервісу з кулінарними рецептами, вимоги спрямовані на забезпечення зручного, безпечного та персоналізованого досвіду для кінцевого користувача.

Функціональні вимоги:

- Надання можливості користувачу шукати рецепти за різними параметрами: назва, тип страви, кухня, дієтичні обмеження, складники.
- Реалізація особистого кабінету користувача з можливістю авторизації та реєстрації.
- Забезпечення можливості зберігати рецепти в «вибране» для подальшого доступу.
- Формування персоналізованих рекомендацій на основі обраних у «вибране» рецептів.
- Надання користувачу можливості переглядати детальну інформацію про рецепт: інгредієнти, інструкції, харчові властивості, калорійність.
- Можливість фільтрації та сортування результатів пошуку за обраними критеріями (наприклад, калорійність, час приготування).
- Реалізація модального вікна для авторизації/реєстрації без перенаправлення користувача з головної сторінки.
- Підключення до зовнішнього API (наприклад, Spoonacular) для отримання актуальних рецептів через власний backend-проксі.

Нефункціональні вимоги:

- Висока швидкодія сервісу: сторінки повинні завантажуватись без затримок, а запити до бази даних і API — оброблятись ефективно.
- Безпека: реалізація шифрування паролів, захист персональних даних користувачів, обмеження доступу до функцій авторизованим користувачам.
- Надійність: система повинна працювати стабільно при одночасному використанні багатьма користувачами.
- Масштабованість: структура проекту має дозволяти легке розширення функціональності в майбутньому.
- Інтуїтивно зрозумілий та приємний інтерфейс, який не потребує додаткового навчання користувача.
- Кросбраузерність та адаптивність: система повинна коректно працювати у сучасних браузерах і на різних пристроях (ПК, планшети, смартфони).
- Логічне розділення backend і frontend частин, а також збереження конфіденційної інформації на сервері (API-ключі, дані користувача).

Вище перелічені вимоги формують загальну основу системи, забезпечують чітке бачення її функціональності та очікуваних характеристик, а також гарантують відповідність проекту потребам користувача. Урахування цих вимог дозволяє побудувати якісний, зручний і надійний кулінарний вебсервіс, здатний ефективно вирішувати поставлені задачі.

3.3 Архітектура системи

Система веб-застосунку з кулінарними рецептами реалізована за архітектурою клієнт-сервер. Такий підхід дозволяє ефективно розділити відповідальність між клієнтською частиною, яка відповідає за взаємодію з користувачем, і серверною частиною, яка обробляє запити, взаємодіє з базою даних та виконує логіку сервісу.

У клієнт-серверній моделі користувач (через браузер) виступає як

клієнт, який надсилає запити до сервера. Сервер, у свою чергу, виконує обробку запитів, звертається до бази даних або зовнішнього API та надсилає у відповідь необхідні дані у структурованому вигляді .

3.3.1 Основні складові системи

- **Клієнтська частина (фронтенд)**

  Це інтерфейс, з яким безпосередньо взаємодіє користувач. Вона реалізована за допомогою HTML, CSS та JavaScript. Фронтенд відповідає за відображення сторінок, обробку подій, виведення результатів пошуку, відкриття модальних вікон для входу/реєстрації та виведення персоналізованих рекомендацій. Також він надсилає запити до серверної частини для отримання або запису даних.

- **Серверна частина (бекенд)**

  Серверна логіка реалізована за допомогою PHP та Node.js. PHP відповідає за обробку запитів, збереження даних у базу даних, взаємодію з API (через проксі), автентифікацію користувачів, управління сесіями. Node.js використовується для обробки асинхронних запитів та забезпечення взаємодії з клієнтською частиною у реальному часі, якщо необхідно.

- **База даних (БД)**

  В якості СУБД використовується MySQL. У базі даних зберігається інформація про зареєстрованих користувачів, їх обране (вибрані рецепти), а також службові дані, необхідні для авторизації, сесій і персоналізації.

Додаткові особливості архітектури:

- Для безпечної роботи з API зовнішніх сервісів використовується проксі-файл на сервері, який приховує API-ключ і не допускає його потрапляння у клієнтський код.
- Архітектура дозволяє масштабування проекту шляхом додавання нових сервісів, сторінок або API-маршрутів без потреби в суттєвих змінах

основної структури.

- Модульний підхід у реалізації frontend-логіки дозволяє легко оновлювати інтерфейс або додавати нові функції, зберігаючи стабільність інших компонентів.

Загалом, архітектура системи забезпечує ефективну взаємодію між її компонентами, дозволяє забезпечити належний рівень продуктивності, безпеки та розширюваності, що робить її придатною для реального використання та подальшого розвитку.

3.3.2 Огляд бази даних

База даних у проекті кулінарного веб-сервісу відповідає за зберігання інформації про користувачів, їхню активність, взаємодію з рецептами, а також за надання функціональності персоналізації та комунікації в системі. Схема бази даних (див. Рис. 10) побудована з урахуванням принципів нормалізації та забезпечує гнучке зберігання інформації, яка використовується як на стороні бекенду, так і у фронтенді.

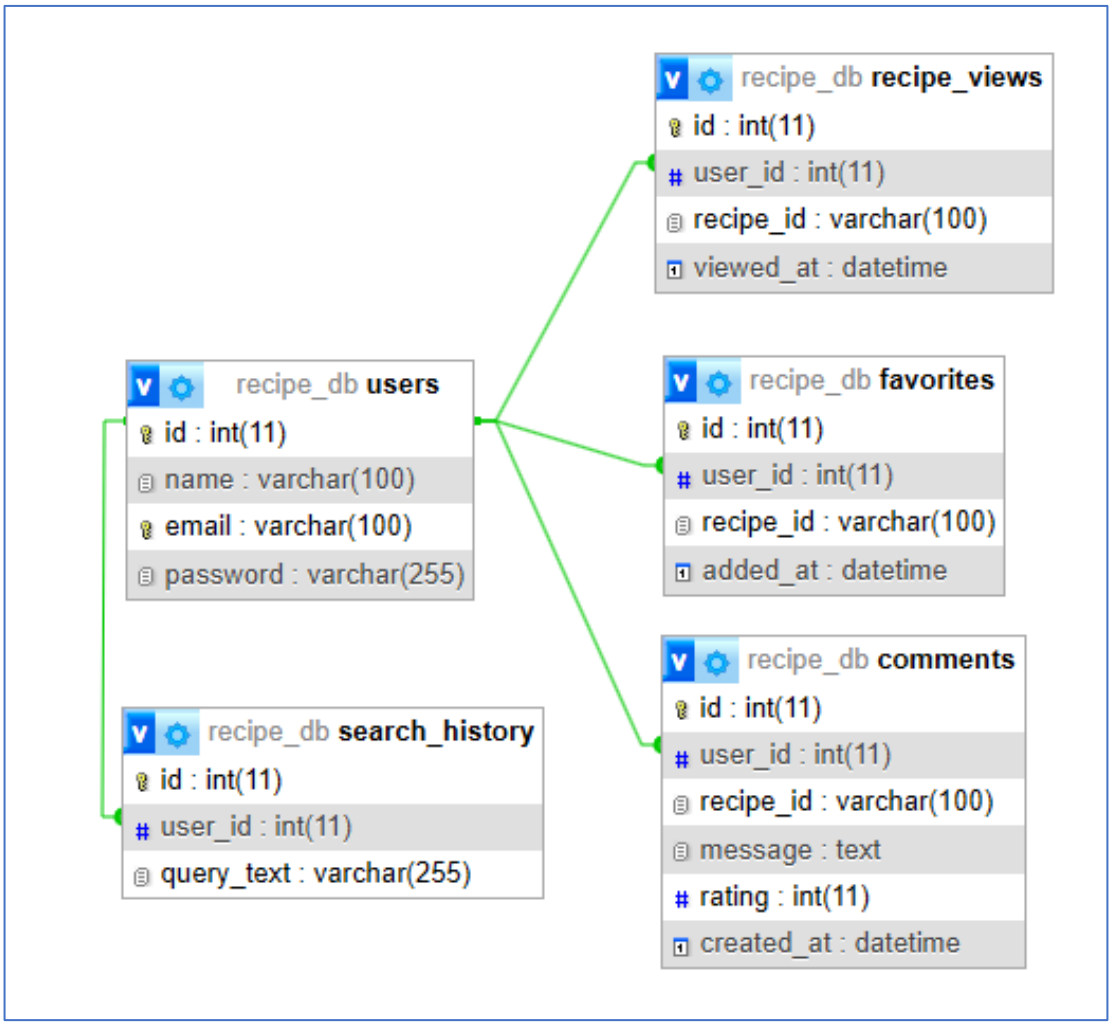


Рисунок *10 – Схема бази даних*

Основні таблиці:

1) **users** — таблиця з інформацією про зареєстрованих користувачів. Стовпці:
    - id — унікальний ідентифікатор користувача, первинний ключ.
    - name — ім’я користувача.
    - email — електронна пошта (унікальна).
    - password — захешований пароль для авторизації.
2) **favorites** — таблиця, яка зберігає рецепти, додані користувачем до «Обраного».

    Стовпці:
    - id — унікальний ідентифікатор запису.

- user_id — зовнішній ключ до таблиці users.
- recipe_id — унікальний ідентифікатор рецепту з зовнішнього API.
- added_at — дата додавання до обраного.

3) **search_history** — таблиця з історією пошукових запитів користувача. Стовпці:
   - id — унікальний ідентифікатор запису.
   - user_id — зовнішній ключ до таблиці users.
   - query_text — текст пошукового запиту.
4) **recipe_views** — таблиця, що фіксує історію переглядів рецептів. Стовпці:
   - id — унікальний ідентифікатор перегляду.
   - user_id — зовнішній ключ до таблиці users.
   - recipe_id — ідентифікатор рецепту (може бути локальний або з API).
   - viewed_at — дата перегляду рецепту.
5) **comments** — таблиця для зберігання коментарів користувачів до рецептів.

   Стовпці:
   - id — унікальний ідентифікатор коментаря.
   - user_id — зовнішній ключ до таблиці users.
   - recipe_id — ідентифікатор рецепту, до якого залишено коментар.
   - message — текст повідомлення.
   - rating — оцінка рецепту користувачем (наприклад, від 1 до 5).
   - created_at — дата і час створення коментаря.

Завдяки цій структурі система підтримує авторизацію, персоналізовану стрічку рекомендацій, функцію «Улюблене», зберігання історії дій користувача та взаємодію з іншими користувачами через коментарі. Усі таблиці пов’язані через зовнішні ключі, що забезпечує цілісність та логічну узгодженість збережених даних.

3.3.3 Огляд серверної частини

Система веб-застосунку з кулінарними рецептами реалізована за архітектурою клієнт

Серверна частина проекту реалізована за класичною клієнт-серверною архітектурою з елементами логічного поділу на окремі модулі. Основна мета серверної частини — обробка запитів від клієнта, взаємодія з базою даних, забезпечення автентифікації користувача, а також захищений доступ до стороннього вільного API Spoonacular.

Сервер складається з кількох PHP-файлів, кожен з яких відповідає за окремий функціонал. Завдяки цьому структура коду стала більш організованою та зручною для підтримки. Деякі частини також можуть бути перенесені до мікросервісів у майбутньому.

Основні модулі серверної частини:

- **auth.php**

  Відповідає за авторизацію та реєстрацію користувачів. Забезпечує перевірку логіна та пароля, хешування паролів при реєстрації та створення нових користувачів у базі даних. Також реалізує перевірку сесії для визначення, чи користувач авторизований.

- **favorites.php**

  Реалізує функціональність "Обраного". Дає змогу додавати рецепти до списку улюблених, переглядати цей список та видаляти з нього елементи. Всі дані пов’язані з конкретним користувачем і зберігаються в базі даних.

- **recommendations.php**

  Обробляє логіку створення персоналізованих рекомендацій на основі обраних рецептів користувача. Аналізує дані в базі та формує запит до API через проксі й повертає користувачеві відповідні рецепти.

- **proxy.php**

  Служить для безпечної роботи із зовнішнім API (Spoonacular). Його головне завдання — приховати API-ключ на сервері та перенаправляти

запити від клієнта до API. Таким чином користувач не має прямого доступу до ключа, що підвищує безпеку.

Висновок

Серверна частина проекту структурована у вигляді набору розділених PHP-файлів, кожен з яких виконує окреме завдання. Такий підхід дозволяє легко розширювати функціонал та підтримувати систему в майбутньому. Модульна структура дозволяє ізолювати окремі логіки (авторизація, запити до API, обробка рекомендацій), що спрощує підтримку і розвиток проекту.

3.3.4 Огляд клієнтської частини

Клієнтська частина веб-застосунку створена за допомогою вебтехнологій — HTML, CSS та JavaScript. Вона відповідає за взаємодію з користувачем: отримання запитів, відображення результатів, обробку подій (наприклад, додавання в обране, перегляд рецепту, написання коментарів) та комунікацію з серверною частиною.

Архітектура реалізована у вигляді окремих HTML-сторінок, які пов’язані зі своїми відповідними JavaScript- та CSS-файлами. Вся логіка побудована у функціональному стилі та структурована для зручної підтримки.

Структура клієнтської частини:

1) **home.html + home.js + home-style.css**

- Основна сторінка, з якої користувач здійснює пошук рецептів.
- Містить динамічний підбір рецептів.
- Рецепти відображаються у вигляді карток з описом.
- Якщо користувач авторизований, з’являється кнопка "Обране" та персональні рекомендації — окремий розділ, сформований на основі вибраних рецептів користувача.
- Вікно для реєстрації та авторизації користувачів.
- Адаптивний і зручний інтерфейс.

2) **recipe.html + recipe.js + recipe-style.css**

- Сторінка детального перегляду рецепта.

- Дані рецепта отримуються за ID, який передається через URL.
- Відображається назва, зображення, інгредієнти, інструкції, калорійність тощо.
- Доступна кнопка "Додати в обране", яка надсилає запит на сервер (використовується fetch до PHP-файлу).
- Є можливість залишати коментар: форма для вводу тексту, вибору оцінки та кнопка надсилання.
- Коментарі та оцінювання рецепту користувачів завантажуються при відкритті сторінки, відображаються внизу.
- Адаптивний і зручний інтерфейс.

3) search.html + search.js + search-style.css

- Сторінка з полем для пошуку рецептів.
- Фільтр пошуку за параметрами рецептів (кухня, дієта тощо).
- Рецепти відображаються у вигляді зручних карточок з описом.
- Відображення кількості знайдених рецептів.
- Адаптивний і зручний інтерфейс.

JavaScript-файли та логіка

1) search.js

- Функції обробки пошуку та фільтрації рецептів за заданими параметрами.
- Динамічне створення карток рецептів.

2) home.js

- Перевірка авторизації користувача.
- Обробка модального вікна входу/реєстрації.
- Реєстрація користувача.
- Кнопка “Вибране” та вихід.
- Динамічне оновлення розділів з картками рецептів.

3) recipe.js

- Виконує запит до API (через проксі) для отримання даних рецепта.

- Відображення деталей рецепта.
- Додавання до “Вибраного”.
- Розділ з коментарями та оцінкам користувачів.

Робота з API та проксі

Оскільки зовнішній API (Spoonacular) потребує API-ключа, запити не надсилаються напряму з браузера. Замість цього використовується **proxy.php**, який:

- отримує параметри від JavaScript;
- формує правильний запит до API (вставляючи API-ключ);
- повертає дані назад у браузер у форматі JSON.

Це дозволяє приховати ключ і забезпечити безпечний доступ до зовнішніх сервісів.

Робота з обраним, історією та коментарями

Вся додаткова інформація, пов’язана з активністю користувача, зберігається в базі даних. JavaScript-частина надсилає запити на PHP-файли, які відповідають за CRUD-операції:

- **Обране**: додавання та перегляд збережених рецептів.
- **Історія пошуку**: фіксується кожен запит, який надсилається з головної сторінки.
- **Перегляди рецептів**: фіксується кожен перехід на сторінку рецепта.
- **Коментарі**: надсилаються через форму, отримуються з БД і виводяться внизу сторінки.

Робота з інтерфейсом

Інтерфейс оновлюється динамічно:

- Після кожної дії (додати в обране, надіслати коментар тощо) DOM перебудовується.
- Якщо користувач не авторизований, деякі елементи приховуються або неактивні.
- Використовується адаптивна верстка, що забезпечує зручність на

мобільних пристроях.

Клієнтська частина реалізована з чітким поділом функцій між сторінками та файлами. Незважаючи на відсутність сучасних фреймворків, структура дозволяє ефективно підтримувати і розвивати проект. Вся логіка користувацької взаємодії, динамічного оновлення інтерфейсу та зв'язку з серверною частиною реалізована за допомогою JavaScript у зручному і зрозумілому вигляді.

3.4 Розробка бази даних

Під час створення серверної частини веб-платформи для роботи з рецептами, як систему управління базами даних було обрано MySQL. Мовою керування даними виступає SQL, оскільки вона є стандартом для створення та маніпулювання реляційними базами.

Для зберігання даних користувачів, обраних рецептів, історії пошуків, переглядів рецептів та коментарів була сформована структура бази даних під назвою recipe_db. Вона включає п'ять основних таблиць, кожна з яких виконує свою функцію.

Лістинг *1 — Створення таблиці користувачів*

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL
);
```

Лістинг *2 — Створення таблиці збережених рецептів*

```
CREATE TABLE favorites (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  user_id INT NOT NULL,
  recipe_id VARCHAR(50) NOT NULL,
  added_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Лістинг *3 — Створення таблиці історії пошуків*

```
CREATE TABLE search_history (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  query_text TEXT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Лістинг *4 — Створення таблиці переглядів рецептів*

```
CREATE TABLE recipe_views (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  recipe_id VARCHAR(50) NOT NULL,
  viewed_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Лістинг *5 — Створення таблиці коментарів*

```
CREATE TABLE comments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  recipe_id VARCHAR(50) NOT NULL,
  message TEXT NOT NULL,
  rating TINYINT CHECK (rating >= 1 AND rating <= 5),
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
```

```
);
```

На лістингах 1-5 представлено повну реалізацію таблиць, що забезпечують функціонування авторизації, збереження вподобаних рецептів, історії взаємодії з контентом, а також додавання коментарів і оцінок. Ця структура дозволяє гнучко взаємодіяти з даними користувача на веб-платформі та реалізовувати персоналізовані функції.

3.5 Програмна реалізація клієнтської частини

3.5.1 Розробка авторизації та керування інтерфейсом користувача

У цьому модулі реалізована логіка роботи з авторизацією користувача на стороні клієнта, а також керування відображенням елементів інтерфейсу в залежності від авторизації. Код відповідає за взаємодію з модальним вікном входу/реєстрації та збереження імені користувача.

Лістинг *6 — Ініціалізація інтерфейсу після завантаження сторінки*

```
document.addEventListener('DOMContentLoaded', () => {
  setupAuthUI();
  setupLoginModal();
});
```

На лістингу 6 запускається налаштування UI та модального вікна після повного завантаження DOM.

Лістинг *7 — Відображення елементів в залежності від авторизації*

```
function setupAuthUI() {
  const username = localStorage.getItem('username');
  const userGreeting =
document.getElementById('userGreeting');
  const loginBtn = document.getElementById('loginBtn');
  const logoutBtn =
```

```
document.getElementById('logoutBtn');
      const profileBtn =
document.getElementById('profileBtn');
      const favoritesBtn =
document.getElementById('favoritesBtn');

      if (username) {
        userGreeting.style.display = 'inline';
        userGreeting.textContent = `Hello, ${username}`;
        loginBtn.style.display = 'none';
        logoutBtn.style.display = 'inline-block';
        profileBtn.style.display = 'inline-block';
        favoritesBtn.style.display = 'inline-block';

        logoutBtn.addEventListener('click', () => {
          localStorage.removeItem('username');
          window.location.reload();
        });

        favoritesBtn.addEventListener('click', () => {
          window.location.href = 'favorites.html';
        });
      } else {
        userGreeting.style.display = 'none';
        loginBtn.style.display = 'inline-block';
        logoutBtn.style.display = 'none';
        profileBtn.style.display = 'none';
        favoritesBtn.style.display = 'none';
      }
    }
```

На лістингу 7 визначається які кнопки показувати або приховувати в

залежності від наявності користувача в localStorage.

Лістинг *8 – Налаштування модального вікна входу/реєстрації*

```
function setupLoginModal() {
  const loginBtn = document.getElementById('loginBtn');
  const loginModal = document.getElementById('loginModal');
  const closeBtn = document.querySelector('.close');
  const switchToRegister = document.getElementById('switchToRegister');
  const switchToLogin = document.getElementById('switchToLogin');
  const loginForm = document.getElementById('loginForm');
  const registerForm = document.getElementById('registerForm');

  if (!loginBtn || !loginModal) return;

  loginBtn.addEventListener('click', () => {
    loginModal.style.display = 'block';
    loginForm.style.display = 'block';
    registerForm.style.display = 'none';
  });

  closeBtn.addEventListener('click', () => {
    loginModal.style.display = 'none';
  });

  window.addEventListener('click', (event) => {
    if (event.target === loginModal) {
      loginModal.style.display = 'none';
```

```
  }
});

switchToRegister.addEventListener('click', () => {
  loginForm.style.display = 'none';
  registerForm.style.display = 'block';
});

switchToLogin.addEventListener('click', () => {
  registerForm.style.display = 'none';
  loginForm.style.display = 'block';
});
```

На лістингу 8 блок відповідає за відкриття, закриття модального вікна та перемикання між формами входу і реєстрації.

Лістинг *9 – Вхід користувача*

```
loginForm.addEventListener('submit', async (e) => {
  e.preventDefault();
  const email = loginForm.email.value;
  const password = loginForm.password.value;

  const res = await fetch('/recipe-search/auth.php',
{
    method: 'POST',
    headers: { 'Content-Type': 'application/x-www-
form-urlencoded' },
    body:
`action=login&email=${encodeURIComponent(email)}&password=$
{encodeURIComponent(password)}`
  });
```

```
    const data = await res.json();
    if (data.success) {
      localStorage.setItem('username', data.username);
      loginModal.style.display = 'none';
      window.location.reload();
    } else {
      alert(data.message || 'Login failed');
    }
  });
```

На лістингу 9 при надсиланні форми входу, запит надсилається на сервер до `auth.php`, після чого у випадку успіху ім’я користувача зберігається.

Лістинг 10 — *Реєстрація нового користувача*

```
  registerForm.addEventListener('submit', async (e) =>
{
    e.preventDefault();
    const name = registerForm.name.value;
    const email = registerForm.email.value;
    const password = registerForm.password.value;

    const res = await fetch('/recipe-search/auth.php',
{
      method: 'POST',
      headers: { 'Content-Type': 'application/x-www-
form-urlencoded' },
      body:
`action=register&name=${encodeURIComponent(name)}&email=${e
ncodeURIComponent(email)}&password=${encodeURIComponent(pas
sword)}`
    });
```

```
      const data = await res.json();
      if (data.success) {
        localStorage.setItem('username', data.username);
        loginModal.style.display = 'none';
        window.location.reload();
      } else {
        alert(data.message || 'Registration failed');
      }
    });
}
```

На лістингу 10 форма реєстрації надсилає дані на сервер. У разі успішної реєстрації користувач одразу авторизується.

Цей JavaScript-файл забезпечує динамічне керування інтерфейсом головної сторінки в залежності від авторизації користувача. Завдяки інтеграції з серверним файлом auth.php, здійснюється обробка входу та реєстрації, а результат зберігається у localStorage для подальшого використання. Наступним можна надсилати auth.php або будь-який інший файл серверної частини.

3.5.2 Розробка клієнтського модуля відображення рецепту та додавання до обраного

Даний модуль відповідає за завантаження повної інформації про вибраний користувачем рецепт, її динамічне відображення на сторінці, а також за можливість додати рецепт до списку обраного. Ключовим елементом є взаємодія з серверним проксі /proxy.php, який забезпечує безпечне звернення до Spoonacular API, та серверним PHP-модулем favorite.php, що керує обраними рецептами у базі даних.

Лістинг *11 — Отримання параметрів URL та ініціалізація сторінки*

рецепту

```
    const urlParams = new
URLSearchParams(window.location.search);
    const recipeId = urlParams.get('id');

    window.addEventListener('DOMContentLoaded', () => {
      if (!recipeId) {
        alert('Recipe not found');
        return;
      }

      loadRecipe();
      setupFavoriteButton();
      setupBackButton();
    });
```

На лістингу 11 ініціалізується сторінка рецепту після її завантаження. Він витягує ідентифікатор рецепту з параметрів URL і викликає основні функції: завантаження даних рецепту, налаштування кнопки додавання до обраного та кнопки повернення.

Лістинг *12 — Завантаження даних рецепту через серверний проксі*

```
    async function loadRecipe() {
      const recipeUrl = `/recipe-
search/proxy.php?path=recipes/${recipeId}/information&inclu
deNutrition=true`;

      try {
        const response = await fetch(recipeUrl);
        const recipe = await response.json();
        renderRecipeDetails(recipe);
      } catch (error) {
```

```
        console.error('Error fetching recipe data:',
error);
        alert('Failed to load the recipe. Please try again
later.');
      }
    }
```

На лістингу 12 здійснюється запит до внутрішнього проксі, який перенаправляє звернення до зовнішнього API. Отримані дані передаються у функцію renderRecipeDetails для виводу на сторінці.

Лістинг *13 — Вивід інформації про рецепт на сторінці*

```
function renderRecipeDetails(recipe) {
  const title = document.createElement('h1');
  title.textContent = recipe.title;

  const img = document.createElement('img');
  img.src = recipe.image;

  const infoBlock = document.createElement('div');
  if (recipe.cuisines.length > 0) {
    const cuisinesEl = document.createElement('p');
    cuisinesEl.innerHTML = `<strong>Cuisines:</strong>
${recipe.cuisines.join(', ')}`;
    infoBlock.appendChild(cuisinesEl);
  }
  recipeDetailsContainer.appendChild(title);
  recipeDetailsContainer.appendChild(img);
  recipeDetailsContainer.appendChild(infoBlock);
  }
```

На лістингу 13 функція динамічно формує вміст сторінки рецепту: назву,

зображення, тип кухні, харчову цінність, список інгредієнтів та інструкції приготування.

Лістинг *14 — Надсилання рецепту до обраного*

```
function setupFavoriteButton() {
  favBtn.addEventListener('click', async () => {
    const formData = new FormData();
    formData.append('action', 'add_favorite');
    formData.append('recipe_id', recipeId);

    const res = await fetch('/recipe-
search/public/favorite.php', {
      method: 'POST',
      body: formData,
      credentials: 'include',
    });

    const data = await res.json();
    if (data.success) {
      alert('Рецепт добавлен в избранное!');
    } else {
      alert('Ошибка: ' + data.message);
    }
  });
}
```

На лістингу 14 користувач може додати рецепт до списку обраного. При натисканні на кнопку формується POST-запит до серверного файлу favorite.php, що записує обраний рецепт до бази даних.

Цей модуль забезпечує користувачу повноцінний перегляд обраного рецепту з інструкціями, можливістю збереження в обране, а також використовує серверний проксі для безпечного доступу до зовнішнього API.

3.5.3 Розробка клієнтської логіки пошуку рецептів

Даний модуль реалізує функціональність пошуку рецептів на клієнтській стороні з урахуванням фільтрів та пагінації. Основною особливістю є використання `proxy.php` на сервері для безпечного звернення до зовнішнього API (наприклад, Spoonacular), що дозволяє приховати API-ключ від клієнта. Також реалізовано формування карток рецептів, фільтрація результатів за заданими параметрами.

Лістинг *15 — Основні змінні та налаштування фільтрів*

```
    // DOM elements
    const searchButton =
document.getElementById('searchButton');
    const filterButton =
document.getElementById('applyFiltersButton');
    const searchInput =
document.getElementById('searchInput');
    const resultsContainer =
document.getElementById('resultsContainer');
    const loadMoreWrapper =
document.getElementById('loadMoreWrapper');

    // Filters
    const cuisineFilter =
document.getElementById('cuisineFilter');
    const mealTypeFilter =
document.getElementById('mealTypeFilter');
    const dietFilter =
document.getElementById('dietFilter');
    const minCaloriesInput =
document.getElementById('minCalories');
    const maxCaloriesInput =
document.getElementById('maxCalories');
```

```
    const minProteinInput =
document.getElementById('minProtein');
    const maxProteinInput =
document.getElementById('maxProtein');
    const minFatInput = document.getElementById('minFat');
    const maxFatInput = document.getElementById('maxFat');
```

На лістингу 15 ініціалізуються всі необхідні елементи DOM для фільтрації пошукового запиту користувача, що дозволяє тонко налаштовувати результати.

Лістинг *16 – Основна функція пошуку та формування URL до proxy.php*

```
    async function searchRecipes(query, isLoadMore = false)
{
      const offset = currentPage * 20;

      const cuisine = cuisineFilter.value;
      const mealType = mealTypeFilter.value;
      const diet = dietFilter.value;

      const minCalories = minCaloriesInput.value || 0;
      const maxCalories = maxCaloriesInput.value || 800;
      const minProtein = minProteinInput.value || 0;
      const maxProtein = maxProteinInput.value || 100;
      const minFat = minFatInput.value || 0;
      const maxFat = maxFatInput.value || 100;

      let url =
`../proxy.php?path=recipes/complexSearch&query=${encodeURIC
omponent(query)}&number=20&offset=${offset}`;

      if (cuisine) url +=
```

```
`&cuisine=${encodeURIComponent(cuisine)}`;
      if (mealType) url +=
`&type=${encodeURIComponent(mealType)}`;
      if (diet) url += `&diet=${encodeURIComponent(diet)}`;

      url +=
`&minCalories=${minCalories}&maxCalories=${maxCalories}`;
      url +=
`&minProtein=${minProtein}&maxProtein=${maxProtein}`;
      url += `&minFat=${minFat}&maxFat=${maxFat}`;
```

На лістингу 16 формується повний URL запиту до proxy.php, додаючи всі вибрані фільтри користувача. Вона дозволяє безпечно та динамічно надсилати параметризовані запити до API Spoonacular без розкриття ключа на фронтенді.

Лістинг 17 — *Обробка відповіді API та рендер карток*

```
data.results.forEach(recipe => {
  const card = createRecipeCard(recipe);
  resultsContainer.appendChild(card);
});
```

На лістингу 17 блок відповідає за відображення отриманих результатів у вигляді карток рецептів. Для кожного рецепту викликається функція createRecipeCard, яка створює HTML-елемент з картинкою та назвою.

Лістинг *18 — Ініціалізація пошуку за параметром URL при завантаженні*

```
window.addEventListener('DOMContentLoaded', () => {
  const params = new
URLSearchParams(window.location.search);
```

```
    const queryFromURL = params.get('query');

    if (queryFromURL) {
      searchInput.value = queryFromURL;
      currentQuery = queryFromURL;
      currentPage = 0;
      searchRecipes(queryFromURL, false);
    }
  });
```

На лістингу 18 код дозволяє автоматично запустити пошук. Це зручно для переходів між сторінками або закладок з попередньо заданим запитом.

3.5.4 Розробка клієнтської логіки авторизації та керування інтерфейсом користувача

У цьому модулі реалізована логіка відкриття та закриття модального вікна авторизації/реєстрації, відправка даних форми на сервер, обробка відповідей, а також керування видимістю елементів інтерфейсу в залежності від стану авторизації користувача.

Лістинг *19 — Ініціалізація елементів та відкриття/закриття модального вікна*

```
document.addEventListener("DOMContentLoaded", () => {
  const loginBtn = document.getElementById("loginBtn");
  const logoutBtn =
document.getElementById("logoutBtn");
  const profileBtn =
document.getElementById("profileBtn");
  const favoritesBtn =
document.getElementById("favoritesBtn");
  const userGreeting =
```

```
document.getElementById("userGreeting");

      const modal = document.getElementById("authModal");
      const closeModal =
document.getElementById("closeModal");
      const toggleMode =
document.getElementById("toggleMode");
      const modalTitle =
document.getElementById("modalTitle");
      const authForm = document.getElementById("authForm");
      const nameField =
document.getElementById("nameField");
      const emailField =
document.getElementById("emailField");
      const passwordField =
document.getElementById("passwordField");
      const submitBtn =
document.getElementById("submitBtn");
      const authMessage =
document.getElementById("authMessage");

      let isRegister = false;

      // Open window
      loginBtn.onclick = () => {
        modal.style.display = "flex";
        authMessage.textContent = '';
        authForm.reset();
      };

      // Close window
      closeModal.onclick = () => {
        modal.style.display = "none";
```

```
    authForm.reset();
    authMessage.textContent = '';
  };

  // Close on click outside the form
  window.onclick = (e) => {
    if (e.target === modal) {
      modal.style.display = "none";
      authForm.reset();
      authMessage.textContent = '';
    }
  };
```

На лістингу 19 код відповідає за ініціалізацію змінних, відкриття та закриття модального вікна авторизації, а також за очищення форми і повідомлень при закритті.

Лістинг 20 — *Переключення між режимами «Вхід» і «Реєстрація»*

```
  // Переключення режимів
  toggleMode.onclick = (e) => {
    e.preventDefault();
    isRegister = !isRegister;
    modalTitle.textContent = isRegister ? "Register" :
"Login";
    nameField.style.display = isRegister ? "block" :
"none";
    submitBtn.textContent = isRegister ? "Register" :
"Login";
    toggleMode.textContent = isRegister ? "Login" :
"Register";
    authMessage.textContent = '';
    authForm.reset();
```

```
};
```

На лістингу 20 дозволяється користувачу перемикатися між формою входу та реєстрації, змінюючи заголовок, кнопку та видимість поля для імені.

Лістинг *21 — Відправка форми авторизації/реєстрації та обробка відповіді*

```
authForm.onsubmit = async (e) => {
  e.preventDefault();
  authMessage.textContent = '';

  if (isRegister && nameField.value.trim() === '') {
    authMessage.textContent = 'Please enter your
name';
    return;
  }

  const formData = new FormData();
  formData.append("action", isRegister ? "register" :
"login");
  formData.append("name", nameField.value.trim());
  formData.append("email", emailField.value.trim());
  formData.append("password", passwordField.value);

  try {
    const res = await fetch(AUTH_URL, {
      method: "POST",
      body: formData
    });

    if (!res.ok) {
      authMessage.textContent = `Server error:
```

```
${res.status}`;
            return;
          }

          const data = await res.json();

          if (data.success) {
            modal.style.display = "none";
            userGreeting.textContent = `Hello,
${data.name}`;
            userGreeting.style.display = "inline";
            logoutBtn.style.display = "inline";
            profileBtn.style.display = "inline";
            loginBtn.style.display = "none";

            if (favoritesBtn) {
              favoritesBtn.style.display = "inline-block";
              favoritesBtn.onclick = () => {
                window.location.href = "favorites.html";
              };
            }
          } else {
            authMessage.textContent = data.message;
          }
        } catch (error) {
          authMessage.textContent = 'Error connecting to
server: ' + error.message;
          console.error(error);
        }
      };
```

На лістингу 21 код відправляє на сервер форму для входу або реєстрації,

обробляє відповідь у форматі JSON та оновлює інтерфейс залежно від результату. У разі успіху відображає привітання, ховає кнопку входу і показує кнопки виходу, профілю та вибране.

Лістинг *22 — Вихід із облікового запису*

```
logoutBtn.onclick = async () => {
  try {
    await fetch(AUTH_URL, {
      method: "POST",
      body: new URLSearchParams({ action: "logout" })
    });
    window.location.reload();
  } catch (error) {
    console.error('Logout error:', error);
  }
};
```

На лістингу 22 обробник кнопки виходу виконує запит на сервер для розлогінення користувача та перезавантажує сторінку.

Лістинг 23 — *Перевірка статусу авторизації при завантаженні сторінки*

```
fetch(AUTH_URL, {
  method: "POST",
  body: new URLSearchParams({ action: "check" })
})
.then(res => res.json())
.then(data => {
  if (data.loggedIn) {
    userGreeting.textContent = `Hello, ${data.name}`;
    userGreeting.style.display = "inline";
    logoutBtn.style.display = "inline";
```

```
            profileBtn.style.display = "inline";
            loginBtn.style.display = "none";

            if (favoritesBtn) {
              favoritesBtn.style.display = "inline-block";
              favoritesBtn.onclick = () => {
                window.location.href = "favorites.html";
              };
            }
          } else {
            userGreeting.style.display = "none";
            logoutBtn.style.display = "none";
            profileBtn.style.display = "none";
            favoritesBtn.style.display = "none";
            loginBtn.style.display = "inline";
          }
        })
        .catch(err => console.error('Check login status
error:', err));
```

На лістингу 23 код забезпечує перевірку стану сесії на сервері при завантаженні сторінки та відповідне оновлення елементів інтерфейсу.

Результатом виконання цього модуля є повноцінна реалізація функціоналу авторизації користувачів на клієнтській частині сайту. Завдяки взаємодії з сервером через асинхронні запити, модуль забезпечує швидке відкриття та закриття модального вікна для входу та реєстрації, коректну обробку введених даних, а також динамічне оновлення інтерфейсу користувача залежно від статусу авторизації. Це дозволяє надійно і зручно керувати сесією користувача, відображати відповідні кнопки і привітання, а також надавати доступ до персоналізованих сторінок, таких як профіль чи вибране. Впровадження перевірки стану входу при завантаженні сторінки гарантує, що користувачі отримують актуальний інтерфейс без необхідності

додаткових дій. Таким чином, модуль підвищує зручність використання ресурсу і забезпечує безперебійну роботу системи аутентифікації.

3.6 Програмна реалізація серверної частини

3.6.1 Розробка серверної логіки авторизації користувачів

У цьому модулі реалізована логіка обробки запитів на реєстрацію, вхід, вихід із системи, а також перевірка статусу авторизації користувача. Сервер приймає POST-запити з параметром `action` та виконує відповідні дії, працюючи з базою даних MySQL через PDO. Використовується захешування паролів для безпечного зберігання, а також сесії для підтримки стану авторизації.

Лістинг 24 — *Обробка реєстрації нового користувача*

```
if ($action === 'register') {
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $passwordHash = password_hash($_POST['password'],
PASSWORD_DEFAULT);

    // Перевірка унікальності email
    $stmt = $pdo->prepare("SELECT id FROM users WHERE
email = ?");
    $stmt->execute([$email]);
    if ($stmt->fetch()) {
        echo json_encode(['success' => false, 'message'
=> 'Email already registered']);
        exit;
    }

    // Додавання нового користувача в БД
```

```
        $stmt = $pdo->prepare("INSERT INTO users (name,
email, password) VALUES (?, ?, ?)");
        $stmt->execute([$name, $email, $passwordHash]);
        $userId = $pdo->lastInsertId();

        // Ініціалізація сесії користувача
        $_SESSION['user_id'] = $userId;
        $_SESSION['user'] = ['name' => $name, 'email' =>
$email];

        echo json_encode(['success' => true, 'name' =>
$name]);
        exit;
```

На лістингу 24 код перевіряє, чи не зареєстрована вже електронна пошта, а при відсутності конфліктів — додає нового користувача та створює сесію.

Лістинг 25 – *Авторизація користувача за email та паролем*

```
    elseif ($action === 'login') {
        $email = trim($_POST['email']);
        $password = $_POST['password'];

        $stmt = $pdo->prepare("SELECT id, name, password
FROM users WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();

        if ($user && password_verify($password,
$user['password'])) {
            $_SESSION['user_id'] = $user['id'];
            $_SESSION['user'] = ['name' => $user['name'],
'email' => $email];
```

```
            echo json_encode(['success' => true, 'name' =>
$user['name']]);
            exit;
        } else {
            echo json_encode(['success' => false, 'message'
=> 'Invalid credentials']);
            exit;
        }
    }
```

На лістингу 25 виконується пошук користувача за email і перевірка пароля.

Лістинг *26 – Авторизація користувача за email та паролем*

```
    elseif ($action === 'logout') {
        session_destroy();
        echo json_encode(['success' => true]);
        exit;
    } elseif ($action === 'check') {
        if (isset($_SESSION['user_id'])) {
            echo json_encode(['loggedIn' => true, 'name' =>
$_SESSION['user']['name']]);
        } else {
            echo json_encode(['loggedIn' => false]);
        }
        exit;
    }
```

На лістингу 26 код забезпечує завершення сесії та повертає статус авторизації користувача при запиті.

Завдяки цьому модулю серверна частина підтримує базові операції

аутентифікації та управління сесіями, що є основою для безпечного входу користувачів у веб-застосунку.

3.6.2 Розробка серверної логіки авторизації користувачів

Цей PHP-скрипт реалізує серверну логіку для керування списком улюблених рецептів користувача. Для роботи використовується сесія PHP для ідентифікації користувача, а взаємодія з базою даних відбувається через PDO. Скрипт підтримує три основні операції: додавання рецепту до обраного, отримання списку обраних рецептів та видалення рецепту зі списку.

Лістинг *27 – Перевірка авторизації користувача та обробка дій*

```
session_start();
require_once __DIR__ . '/includes/db.php';

header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
    echo json_encode(['success' => false, 'message' => 
'Неавторизованный пользователь']);
    exit;
}

$action = $_POST['action'] ?? '';
$userId = $_SESSION['user_id'];
```

На лістингу 27 починається сесія, підключається файл для роботи з базою даних, а також перевіряється, чи авторизований користувач, і визначається запитувана дія.

Лістинг *28 – Додавання рецепту до обраного*

```
    if ($action === 'add_favorite') {
        $recipeId = $_POST['recipe_id'] ?? '';
        if (!$recipeId) throw new Exception('ID рецепта не
указан');

        $stmt = $pdo->prepare('SELECT id FROM favorites
WHERE user_id = ? AND recipe_id = ?');
        $stmt->execute([$userId, $recipeId]);
        if ($stmt->fetch()) {
            echo json_encode(['success' => false, 'message'
=> 'Рецепт уже в избранном']);
            exit;
        }

        $stmt = $pdo->prepare('INSERT INTO favorites
(user_id, recipe_id, added_at) VALUES (?, ?, NOW())');
        $stmt->execute([$userId, $recipeId]);

        echo json_encode(['success' => true]);
        exit;
    }
```

На лістингу 28 блок перевіряє, чи рецепт вже доданий користувачем у «Обране», і якщо ні — додає його в таблицю favorites з поточною датою.

Лістинг 29 — *Отримання списку обраних рецептів*

```
    elseif ($action === 'get_favorites') {
        $stmt = $pdo->prepare('SELECT recipe_id, added_at
FROM favorites WHERE user_id = ? ORDER BY added_at DESC');
        $stmt->execute([$userId]);
        $favorites = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
        echo json_encode(['success' => true, 'favorites' =>
$favorites]);
        exit;
    }
```

На лістингу 29 повертається у відповідь JSON-об’єкт зі списком рецептов, які користувач додав у «Обране», впорядкованих за датою додавання.

Лістинг 30 – *Видалення рецепту зі списку обраних*

```
    elseif ($action === 'remove_favorite') {
        $recipeId = $_POST['recipe_id'] ?? '';
        if (!$recipeId) throw new Exception('ID рецепта не
указан');

        $stmt = $pdo->prepare('DELETE FROM favorites WHERE
user_id = ? AND recipe_id = ?');
        $stmt->execute([$userId, $recipeId]);

        echo json_encode(['success' => true]);
        exit;
    }
```

На лістингу 30 відбувається видалення запису рецепту з таблиці favorites для авторизованого користувача.

Лістинг 31 – *Обробка помилок та невірних дій*

```
    else {
        echo json_encode(['success' => false, 'message' =>
'Неверное действие']);
        exit;
    }
```

```
    } catch (Exception $e) {
        echo json_encode(['success' => false, 'message' =>
$e->getMessage()]);
    }
```

На лістингу 31 скрипт обробляє випадки помилкових дій або винятків, повертаючи відповідний JSON з описом проблеми.

Таким чином, цей серверний модуль забезпечує надійну роботу зі списком улюблених рецептів користувача, підтримуючи основні операції додавання, перегляду та видалення.

3.6.2 Розробка серверного проксі для роботи з API Spoonacular

У цьому модулі реалізовано проміжний серверний скрипт на PHP, який забезпечує безпечне звернення до зовнішнього API Spoonacular. Основне завдання цього проксі — приховати API-ключ від клієнтської частини та виконувати запити до API від імені користувача сервера.

Лістинг 32 — *Серверний проксі для перенаправлення запитів до Spoonacular API*

```
    $base = 'https://api.spoonacular.com';
    $path = $_GET['path'] ?? '';
    $queryParams = $_GET;
    unset($queryParams['path']); // видаляємо служебний
параметр

    // Add the API key to the request parameters
    $queryParams['apiKey'] = $apiKey;

    // Forming the full URL for the request
    $url = $base . '/' . ltrim($path, '/') . '?' .
http_build_query($queryParams);
```

```
// Making an HTTP request via cURL
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close($ch);
```

На лістингу 32 приймаються параметри запиту з клієнта, окрім спеціального `path`, який визначає кінцеву точку API. До параметрів додається приватний API-ключ, що зберігається тільки на сервері. Запит перенаправляється до зовнішнього сервісу Spoonacular за допомогою cURL. Таким чином, API-ключ не розголошується в браузері користувача, що підвищує безпеку застосунку.

Цей серверний проксі забезпечує централізовану точку доступу до API, що спрощує подальше управління запитами та їх обробку.

3.7 Розробка дизайну застосунку

Під час проектування зовнішнього вигляду вебзастосунку було зосереджено увагу на зручності користування, візуальній простоті та легкості орієнтування. Основне завдання полягало в тому, щоб створити такий інтерфейс, у якому користувач міг би швидко знаходити потрібні рецепти, фільтрувати їх за параметрами, зберігати у вибране та переглядати детальну інформацію — і все це без зайвих переходів чи перешкод.

На самому початку було обрано кольорову гаму, яка асоціюється з темою кулінарії та здорового харчування. Переважно використовуються світлі відтінки білого, сірого і зеленого кольорів, які не відволікають увагу від вмісту та створюють враження чистоти й простору.

Для поліпшення юзабіліті структуру інтерфейсу було розділено на кілька ключових екранів: головну сторінку, сторінку пошуку, детальну

сторінку рецепту та сторінку з улюбленими рецептами. Усі елементи інтерфейсу адаптуються до ширини екрана, що дозволяє зручно користуватися сайтом як з комп'ютера, так і з мобільного пристрою.

Головна сторінка

Головний екран (див. Рис. 11) містить привітання, поле для пошуку, та розділ персоналізованих рекомендацій, якщо користувач авторизований. У верхній частині знаходиться шапка сайту — логотип, назва сервісу, кнопки входу та реєстрації, а також доступ до профілю та сторінки «Вибране» у разі входу. В центральній частині відображаються рецепти, що рекомендуються на основі попередніх уподобань користувача. Кожен рецепт представлений у вигляді картки з фото, назвою і короткою інформацією.

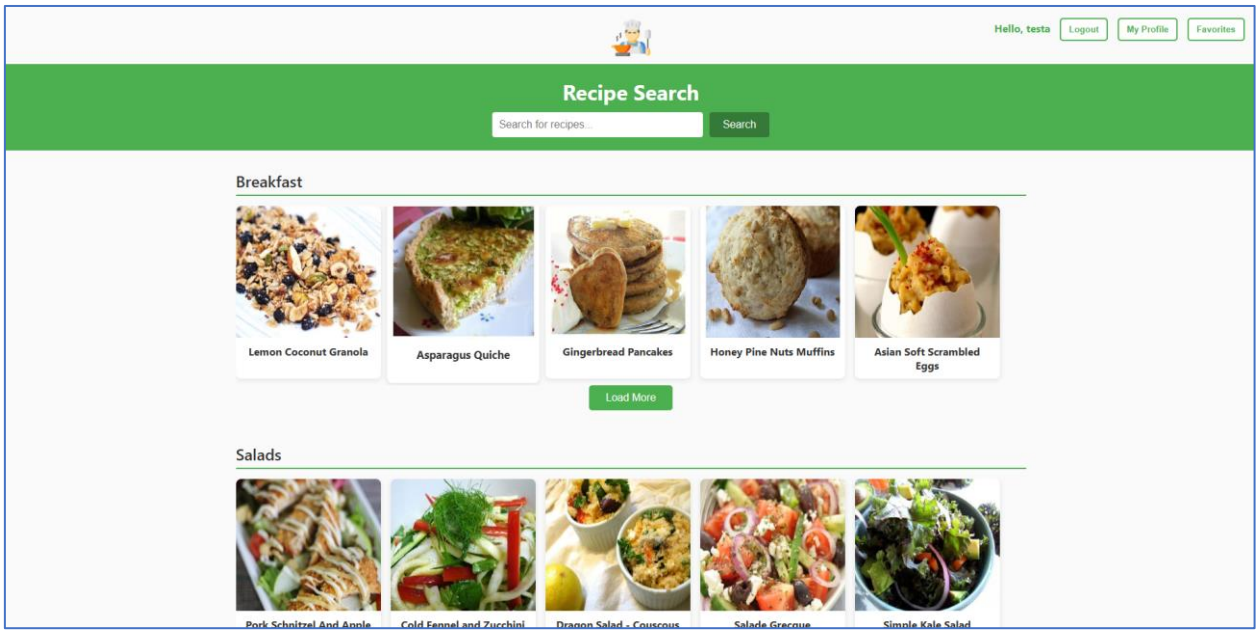


Рисунок *11 – Головна сторінка з персоналізованими рекомендаціями*

Сторінка пошуку рецептів

Сторінка пошуку (див. Рис. 12) реалізована у вигляді двоколонкового макету. Ліва колонка містить фільтри за кухнею, типом страви, дієтою, а також за харчовими показниками — калоріями, білками та жирами. Права частина — це блок результатів у вигляді сітки карток рецептів. Картки мають зручний

вигляд. Це дозволяє користувачу бачити всі результати без порушення структури сторінки.

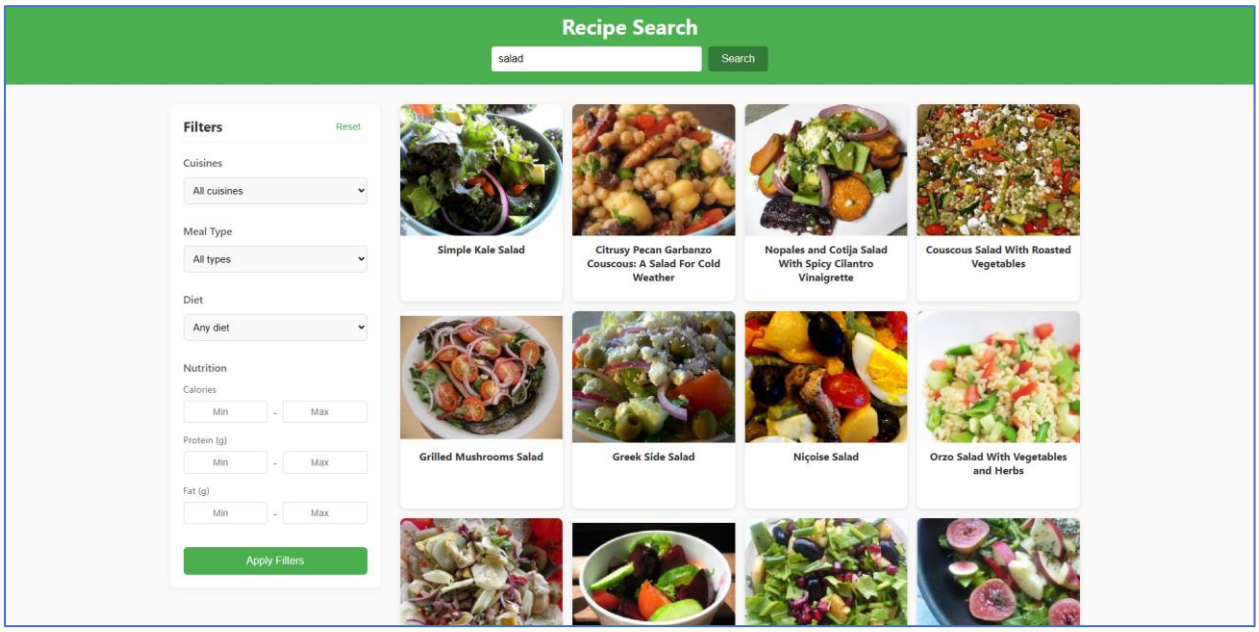


Рисунок *12 – Інтерфейс сторінки пошуку з фільтрами та результатами*

Сторінка перегляду рецепту

На окремій сторінці перегляду рецепту (див. Рис. 13) міститься детальна інформація про обрану страву: велике зображення, список інгредієнтів, покрокова інструкція приготування, харчова цінність, походження рецепту, а також кнопка «Додати в обране». Інтерфейс реалізовано у форматі однієї колонки, що дозволяє зручно читати вміст без прокручування вбік.

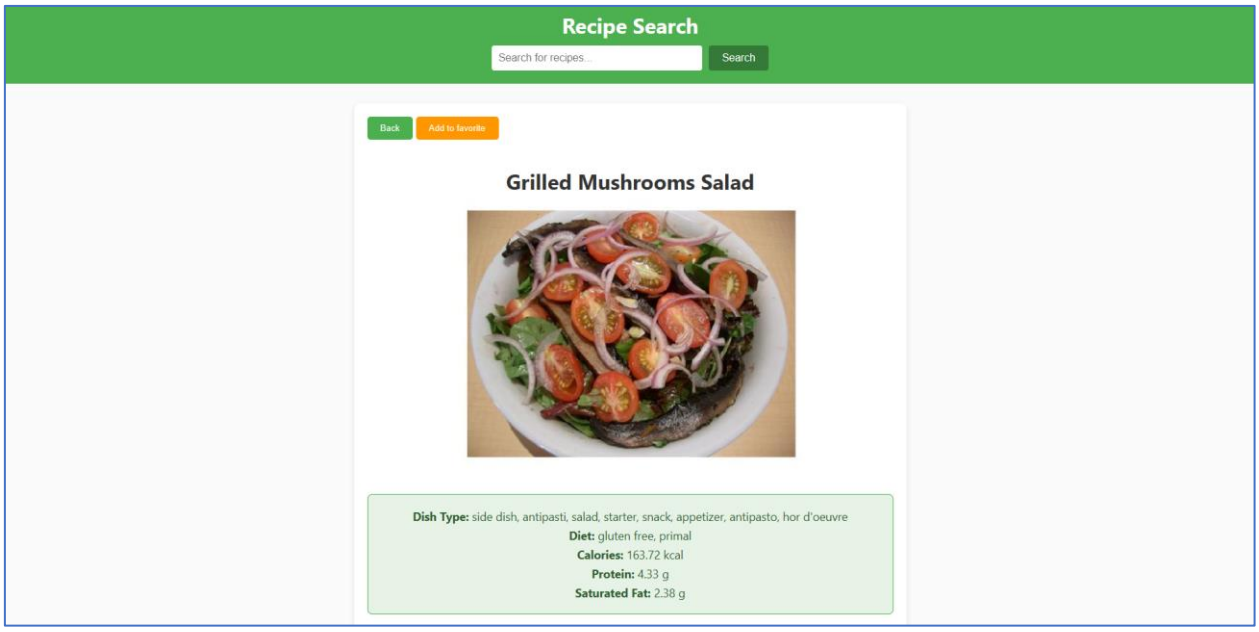


Рисунок *13 – Інтерфейс сторінки перегляду обраного рецепту*

Сторінка профілю користувача

Окрему увагу в інтерфейсі застосунку було приділено особистому кабінету користувача, де має можливість переглядати особисту інформацію (див. Рис. 14). Сторінка профілю реалізована в мінімалістичному стилі, що дозволяє зосередитися саме на функціональності.

На сторінці відображається ім’я користувача та адреса електронної пошти, яку було вказано під час реєстрації. Нижче розміщено список, який дозволяє обрати особисті харчові вподобання.

Після вибору бажаної дієти користувач може зберегти зміни за допомогою кнопки «Зберегти налаштування», розташованої під формою. Після збереження застосунок у подальшому використовуватиме ці параметри при формуванні персоналізованих рекомендацій на головній сторінці.

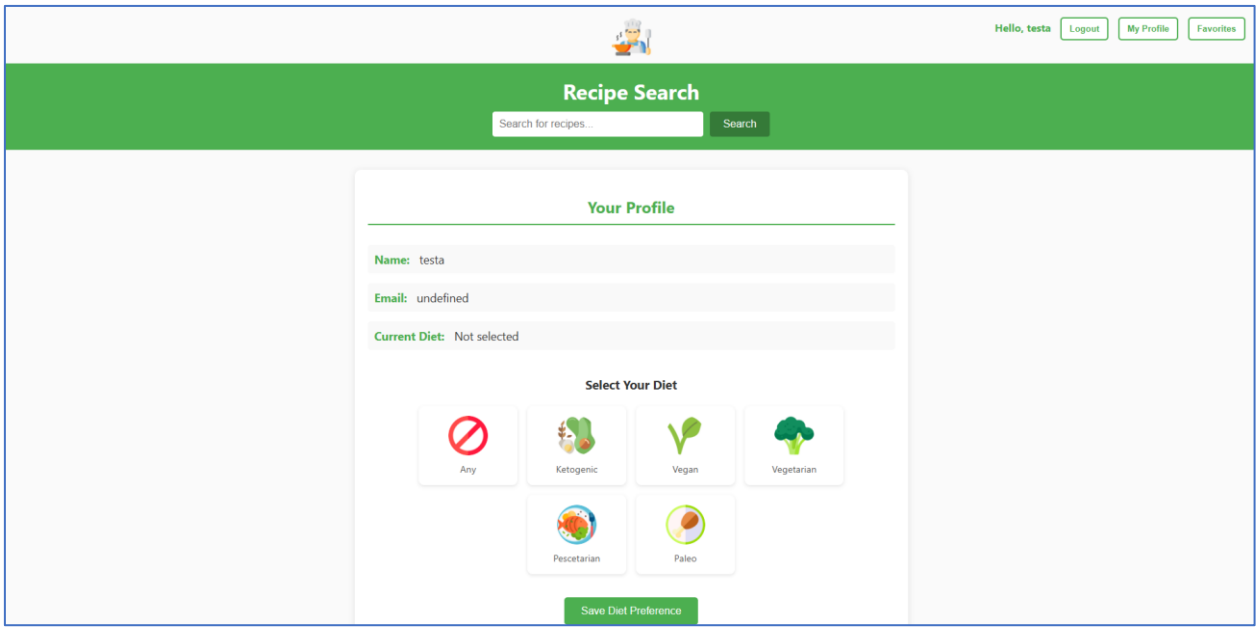


Рисунок *14 – Інтерфейс сторінки профілю користувача*

При створенні зовнішнього вигляду застосунку були враховані сучасні дизайнерські підходи та принципи зручного користувацького досвіду. Інтерфейс залишився простим, логічним та водночас функціональним. Адаптивна верстка дає змогу користуватись системою з будь-якого пристрою, що особливо важливо для цільової аудиторії, яка шукає рецепти не лише на ПК, а й з мобільного телефону чи планшета. Завдяки цьому досягнуто оптимального поєднання естетики, доступності та функціональності.

3.8 Апробація веб-застосунку

Після повного впровадження основної логіки функціонування застосунку, включно з персоналізованим підбором рецептів, пошуком за фільтрами, системою обраного та особистим профілем, було здійснено тестове використання веб-застосунку у середовищі, наближеному до реальних умов.

Для початку користувач потрапляє на головну сторінку, де представлені добірки популярних рецептів, а також персональні рекомендації

(за умови входу в обліковий запис і наявності щонайменше п’яти збережених вподобань). Інтерфейс дозволяє зручно переглядати запропоновані варіанти та швидко переходити до повного опису рецепту.

При введенні запиту в поле пошуку, користувач отримує результати. Є можливість використовувати фільтри: кухня, тип страви, дієтичні обмеження, а також діапазони для калорійності, білків і насичених жирів. Це дозволяє легко адаптувати пошук до особистих харчових вподобань.

При натисканні на будь-який рецепт відкривається сторінка з детальним описом. Тут містяться: список інгредієнтів, спосіб приготування, візуальне оформлення страви, а також кнопка «Додати в обране». Натискаючи її, рецепт додається до особистого списку користувача, збереженого в базі даних.

На сторінці обраного доступний повний перелік усіх доданих рецептів. Користувач має змогу видаляти їх або переходити до приготування. Якщо обране містить достатню кількість страв, то на головній сторінці активується механізм рекомендацій, який підбирає нові рецепти на основі уподобаних: за кухнею, дієтами та схожими інгредієнтами.

Також перевірено роботу особистого профілю. Користувач може переглядати свої дані: ім’я, електронну пошту та встановлювати особисту дієту через відповідний список. Обрані налаштування зберігаються в базі даних та використовуються для фільтрації рецептів.

У процесі тестування було підтверджено, що застосунок швидко реагує на дії користувача, має зрозумілий інтерфейс, а всі основні функції працюють стабільно. Веб-застосунок виконує своє призначення — допомагає користувачам легко знаходити рецепти відповідно до індивідуальних уподобань, зберігати їх та отримувати релевантні поради.

3.9 Висновки до розділу 3

1. У процесі розробки було створено повноцінну архітектуру веб-сервісу з

персональними рекомендаціями рецептів, яка базується на моделі «клієнт–сервер» та дозволяє ефективно взаємодіяти з користувачем і зовнішніми API.

2. Здійснено формалізацію вимог до функціональності застосунку, включно з підтримкою пошуку, фільтрації, додаванням у вибране, збереженням історії переглядів та вподобань, що дозволило побудувати адаптивну та зручну систему.
3. Розроблено інтерфейс користувача, який включає головну сторінку з динамічним пошуком, переглядом рецептів та рекомендацій, а також сторінку детального опису рецептів.
4. Розроблено та впроваджено структуру бази даних, яка включає таблиці для зберігання користувачів, улюблених рецептів, історії пошуків, переглядів і коментарів. Це дозволяє зберігати користувацьку активність і використовувати її для генерації персоналізованих рекомендацій.
5. Перевірено функціональність усіх основних модулів через апробацію веб-застосунку: було перевірено коректність авторизації, додавання в «Обране», генерацію підбору на основі вподобань користувача, а також адаптивність дизайну на різних пристроях.
6. Усі компоненти застосунку структуровані таким чином, щоб забезпечити легку підтримку й подальше масштабування: додавання нових фільтрів, розширення особистого кабінету користувача або впровадження нових моделей рекомендацій.

ВИСНОВКИ

1. У ході роботи було вивчено сучасні технології веброзробки, зокрема методи реалізації клієнтсько-серверної архітектури, способи зберігання та обробки даних у базі даних, а також забезпечення авторизації користувачів у веб-застосунку.
2. Проведено аналіз існуючих кулінарних сервісів, що дозволило сформувати вимоги до функціоналу та інтерфейсу власного проекту, зробивши його зручним у використанні для широкого кола користувачів.
3. Було реалізовано серверну частину, яка забезпечує взаємодію з базою даних, обробку запитів на вхід, реєстрацію, роботу з персональними обліковими даними та обробку запитів до стороннього API.
4. Клієнтська частина застосунку динамічно взаємодіє з сервером, дозволяє користувачам зберігати рецепти у вибране, бачити персоналізовані рекомендації.
5. У результаті реалізовано повноцінний вебсервіс, що дозволяє користувачам отримувати індивідуальні рекомендації кулінарних рецептів на основі їхніх уподобань, з можливістю зручної авторизації, керування обраним та взаємодії з зовнішнім API.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. W3Schools. JavaScript Tutorial. URL: https://www.w3schools.com/js/ (дата звернення: 07.05.2025).
2. Mozilla Developer Network (MDN). Using Fetch. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (дата звернення: 07.05.2025).
3. Spoonacular API Documentation. URL: https://spoonacular.com/food-api/docs (дата звернення: 07.05.2025).
4. PHP Manual. cURL Functions. URL: https://www.php.net/manual/en/book.curl.php (дата звернення: 07.05.2025).
5. PHP Manual. Superglobals. URL: https://www.php.net/manual/en/language.variables.superglobals.php (дата звернення: 07.05.2025).
6. MySQL 8.0 Reference Manual. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 07.05.2025).
7. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 07.05.2025).
8. Bootstrap Documentation. Modal component. URL: https://getbootstrap.com/docs/5.3/components/modal/ (дата звернення: 07.05.2025).
9. HTML Living Standard. WHATWG. URL: https://html.spec.whatwg.org/ (дата звернення: 07.05.2025).
10. OAuth 2.0 and OpenID Connect Overview. URL: https://oauth.net/2/ (дата звернення: 07.05.2025).
11. bcrypt for PHP Password Hashing. URL: https://www.php.net/manual/en/function.password-hash.php (дата звернення: 07.05.2025).
12. JavaScript Event Handling. URL: https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener (дата звернення:

07.05.2025).

13. UX Design Principles. Nielsen Norman Group. URL: https://www.nngroup.com/articles/definition-user-experience/ (дата звернення: 07.05.2025).
14. XAMPP Official Site. URL: https://www.apachefriends.org/index.html (дата звернення: 07.05.2025).
15. RESTful API Design Guidelines. URL: https://restfulapi.net/ (дата звернення: 07.05.2025).

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Патей Євгеній Владиславович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми "Програмне забезпечення систем", адреса електронної пошти se121-10@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Web-застосунок персоналізованих рекомендацій кулінарних рецептів»** відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 02.06.2025 _____ Патей Євгеній Владиславович
(підпис) (прізвище та ініціали) (студент)

Дата 02.06.2025 _____ Безверхий Анатолій Ігорович
(підпис) (прізвище та ініціали) (керівник)