Міністерство освіти і науки Україниів
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПОШУКУ ТА ГЕНЕРАЦІЇ КУЛІНАРНИХ РЕЦЕПТІВ НА ОСНОВІ УПОДОБАНЬ КОРИСТУВАЧА З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-2
Гризенко А.Р.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Вечірська І.Д.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Гризенку Андрію Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебзастосунку для пошуку та генерації кулінарних рецептів на основі смакових уподобань користувача з використанням технологій штучного інтелекту

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 24 червня 2026 р.
3. Вихідні дані до роботи Науково-технічна література за напрямом вебтехнологій та штучного інтелекту; документація та специфікації до FastAPI, React, Supabase та Ollama; аналітичні звіти щодо ринку кулінарних цифрових сервісів.
4. Перелік питань, що потрібно опрацювати в роботі
1.Аналіз предметної області, огляд існуючих рішень та постановка задачі.
2.Дослідження методів профілювання, рекомендаційних систем та LLM для генерації рецептів.
3.Обґрунтування вибору технологічного стеку (Python, FastAPI, React, Supabase, Ollama).
4.Проєктування архітектури системи, UML-діаграм та структури бази даних.
5.Розробка серверної частини (FastAPI): модулі автентифікації, профілю та взаємодії з LLM.
6.Розробка клієнтської частини (React SPA): інтерактивний інтерфейс із таймерами приготування.
7.Модульне, інтеграційне тестування та верифікація фільтрації алергенів.
8.Розробка інструкції з розгортання, посібника користувача та оцінка результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Комп'ютерні ілюстрації (рисунки): загальна архітектура системи «Smart Recipes», ER-діаграма бази даних (PostgreSQL/Supabase), UML-діаграма прецедентів системи, фрагмент реалізації модуля генерації рецептів, екранні форми інтерфейсу (смаковий профіль, модуль «Мій холодильник», режим приготування).

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-05.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Вечірська І.Д._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 97 с., 2 табл., 25 рис., 2 дод., 44 джерел.

УПОДОБАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, OLLAMA, ВЕБЗАСТОСУНОК, ГЕНЕРАЦІЯ, ПЕРСОНАЛІЗАЦІЯ, PYTHON.

Об'єктом роботи є процес персоналізованого пошуку та генерації кулінарних рецептів з використанням технологій штучного інтелекту.

Метою роботи є проєктування та створення кулінарної ШІ-платформи з адаптивним формуванням рецептур, персоналізованою фільтрацією алергенів та покроковим гейміфікованим приготуванням страв.

Методи дослідження та розробки включають системний аналіз, UML-моделювання, проєктування баз даних, веброзробку на мові Python та інтеграцію моделей штучного інтелекту через API.

У ході роботи проаналізовано ринок кулінарних сервісів, побудовано діаграми прецедентів, сформульовано бізнес-правила та розроблено архітектуру модулів системи, а саме генератора страв за смаками, інтелектуального аналізатора вмісту холодильника, профілю безпеки та інтерактивного інтерфейсу покрокового приготування з вбудованими таймерами.

PREFERENCES, ARTIFICIAL INTELLIGENCE, OLLAMA, WEB APPLICATION, GENERATION, PERSONALIZATION, PYTHON.

The objective of this work is the process of personalized culinary recipe search and generation using artificial intelligence technologies.

The aim of work is to design and develop an AI-driven culinary platform with adaptive recipe generation, allergen filtering, and gamified step-by-step cooking.

Methods of research include systematic analysis, UML modeling, database design, Python-based web development, and AI models integration via API.

In the course of the work, the culinary services market was analyzed, business rules were formulated, and the system architecture was developed, including a taste generator, a fridge analyzer, an allergen filter, and a step-by-step cooking interface with timers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (програмний інтерфейс застосунку)

БД – база даних

CRUD – Create, Read, Update, Delete (базові операції з даними)

HTTP – Hypertext Transfer Protocol (протокол передавання гіпертексту)

JSON – JavaScript Object Notation (формат обміну даними)

JWT – JSON Web Token (стандарт аутентифікації)

LLM – Large Language Model (велика мовна модель)

ORM – Object-Relational Mapping (об'єктно-реляційне відображення)

REST – Representational State Transfer (архітектурний стиль API)

SPA – Single Page Application (односторінковий застосунок)

SQL – Structured Query Language (мова структурованих запитів)

UI – User Interface (інтерфейс користувача)

UML – Unified Modeling Language (уніфікована мова моделювання)

UX – User Experience (досвід користувача)

ШІ – штучний інтелект

IVFFlat – Inverted File Flat (алгоритм векторного індексування)

HNSW – Hierarchical Navigable Small World (ієрархічний навігований «маленький світ»)

RLS – Row Level Security (безпека на рівні рядків)

ВСТУП

Сучасний ритм повсякденного життя зумовлює зростаючу потребу в автоматизації планування харчування. Мільярди людей щодня витрачають значний час на вибір страв, підбір рецептів та з'ясування того, що можна приготувати з наявних продуктів. Особливо гостро ця проблема відчувається у великих містах, де темп життя не залишає часу на тривале обдумування меню. Кожного вечора мільйони людей стикаються з одним і тим самим питанням про те, що саме приготувати на вечерю, щоб це було і смачно, і корисно, і не потребувало багато зусиль. Попри величезну кількість кулінарних ресурсів в Інтернеті, більшість із них не враховують індивідуальних смакових уподобань, дієтичних обмежень і реального вмісту холодильника конкретної людини. Користувач змушений самостійно фільтрувати сотні рецептів, відкидаючи ті, що містять небажані інгредієнти або не відповідають його харчовим звичкам. Такий ручний пошук часто перетворюється на виснажливий процес, який забирає купу сил і змушує людей купувати вже готову їжу замість домашньої.

Розвиток технологій штучного інтелекту, зокрема великих мовних моделей та рекомендаційних систем, відкриває принципово нові можливості для розв'язання цієї проблеми. Генеративний ШІ здатний не лише добирати відповідні рецепти, але й синтезувати нові, враховуючи складні набори обмежень і уподобань. Завдяки цьому стає можливим створення систем, що адаптуються до кожного користувача індивідуально, а не пропонують універсальні рішення для всіх. Комп'ютерні алгоритми тепер можуть виконувати роль персонального кулінарного шефа, який точно знає, що любить його господар і чого йому їсти не можна. Ця технологічна база дозволяє перейти від статичного пошуку до динамічного, персоналізованого формування рецептур. Актуальність даної роботи підтверджується зростаючим попитом на персоналізовані цифрові сервіси у сфері харчування

та очевидними обмеженнями наявних рішень, які не забезпечують справжньої персоналізації на основі ШІ. Такі популярні платформи, як Yummly, BigOven чи Spoonacular, пропонують широкий каталог рецептів, проте не здатні генерувати нові страви під конкретного користувача в реальному часі. Вони просто показують заздалегідь заготовлені тексти з бази даних, які часто не влаштовують людину на сто відсотків. Це підкреслює необхідність розробки принципово нового підходу до організації кулінарного досвіду, який би зробив процес вибору страв простим, швидким та максимально точним.

Об'єктом дослідження є процес персоналізованого пошуку та ШІ-генерації кулінарних рецептів за індивідуальними параметрами користувача. Дослідження охоплює всі етапи взаємодії людини з рецептами – від перевірки наявних продуктів дома до створення унікальних кулінарних інструкцій.

Метою роботи є проєктування та розробка інтелектуального вебзастосунку на базі Python для адаптивного формування рецептур, персоналізованої фільтрації алергенів та покрокового гейміфікованого приготування страв. Створений продукт має стати надійним щоденним помічником, який повністю візьме на себе рутину планування харчування та допоможе користувачеві готувати із задоволенням.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області та виявлення практичної проблеми

Приготування їжі є важливою складовою нашого побуту, адже щодня кожна людина приймає безліч рішень щодо того, що приготувати та як спланувати свій раціон. Цифровізація кардинально змінила цю галузь: паперові збірки рецептів відходять у минуле, поступаючись місцем вебсайтам, а смартфони перетворилися на головне джерело пошуку кулінарних ідей.

Сьогодні в інтернеті доступні мільйони різноманітних інструкцій, проте юзери часто потрапляють у пастку надмірного вибору. Величезні обсяги даних лише ускладнюють пошук конкретної страви, яка б ідеально пасувала під особисті вподобання, наявні вдома продукти, обмеження в дієті чи кулінарні навички людини.

Аналіз поведінки споживачів показує, що через заплутану навігацію на сайтах люди частіше готують перевірені страви і рідко експериментують. Статистика свідчить, що понад 70% користувачів незадоволені роботою алгоритмів на кулінарних порталах, оскільки їм доводиться витрачати забагато часу на самостійний пошук потрібного варіанта.

Головна прикрость полягає у відсутності «розумного» сервісу, який міг би автоматично аналізувати смаки людини, створювати індивідуальні підбірки та пропонувати нові рецепти на основі її профілю. Більшість сучасних систем використовують лише базове сортування (за типом страви, тривалістю чи складністю), повністю ігноруючи алгоритми машинного навчання для глибокої персоналізації.

Ще один мінус – відсутність інтелектуального контролю за продуктовими запасами. Сучасні програми не вміють оцінювати вміст холодильника користувача, щоб підібрати страву суто з того, що вже є під

рукою. Через це люди часто викидають зіпсовану їжу та витрачають зайві кошти.

Крім того, розробники майже не приділяють уваги інтерактивності під час самого готування. Існуючі програми просто видають текст із кроками та списком компонентів, але не пропонуюць вбудованих таймерів, голосових команд чи елементів гейміфікації, які б мотивували новачків.

Отже, існує реальна потреба у створенні сучасного вебзастосунку із залученням штучного інтелекту. Таке рішення дозволить комплексно закрити питання персоналізованого підбору та генерації страв, контролю за інгредієнтами вдома та забезпечить зручний гейміфікований процес приготування [1].

1.2 Огляд існуючих програмних рішень та їх аналіз

Щоб чітко окреслити вектор майбутньої розробки, було виконано порівняльну характеристику існуючих аналогів у цій предметній сфері. Для огляду ми обрали чотири типові системи, які найкраще відображають сучасний стан ринку кулінарних вебплатформ.

Yummly вважається одним із найпопулярніших кулінарних ресурсів [2], який має величезну базу страв та власні алгоритми підбору. Серед плюсів можна виділити детальну систему тегів, можливість враховувати дієти чи харчові алергії, збереження улюбленого та адаптований мобільний додаток. Проте є й мінуси: видача пропозицій спирається здебільшого на виставлені користувачем фільтри, а не на інтелектуальний аналіз його смаків. Крім того, тут немає функції створення нових рецептів, а сам сервіс орієнтований в основному на англомовних юзерів.

Spoonacular API представляє собою спеціалізовану платформу [3] з потужним інструментарієм для розробників, що дозволяє шукати страви за компонентами, калорійністю чи кулінарними традиціями різних країн.

Головні переваги це широкий функціонал API, великий каталог та детальний розрахунок БЖУ і калорій. Головним мінусом є відсутність реальної персоналізації за допомогою штучного інтелекту, а також суворі ліміти на безкоштовному тарифному плані, що сильно обмежує його повсякденне використання.

Універсальні великі мовні моделі (як-от ChatGPT, Claude тощо) також непогано справляються зі створенням рецептів за звичайним текстовим описом від користувача. Їхні сильні сторони – це висока якість відповідей, гнучкість у сприйнятті промптів та здатність адаптуватися під будь-які побажання людини. Водночас суттєвими недоліками є повна відсутність профільного кулінарного інтерфейсу, неможливість зберігати історію уподобань між різними діалогами та відсутність єдиної структурованої бази даних.

BigOven – це відомий мобільний додаток, де реалізовано контроль продуктових запасів та підбір страв під них. Опція «використай залишки» допомагає знаходити варіанти обіду чи вечері за наявними компонентами, а календар меню полегшує планування раціону на тиждень. Попри це, програма не вміє самостійно генерувати нові рецепти за допомогою AI та слабко враховує особисті смакові преференції користувача. Результати проведеного порівняльного аналізу наведено нижче в таблиці 1.1.

Таблиця 1.1 — Порівняльний аналіз існуючих рішень

Критерій	**Yummly**	**Spoonacular**	**ChatGPT**	**BigOven**	**Smart Recipe**
1	**2**	**3**	**4**	**5**	**6**
Персоналізація на основі ШІ	Частково	Ні	Частково	Ні	Так
Генерація нових рецептів	Так	Ні	Ні	Так	Так

Продовження таблиці 1.1

1	2	3	4	5	6
Збереження профілю уподобань	Так	Ні	Ні	Так	Так
Спеціалізований Інтерфейс	Так	Ні	Ні	Так	Так
Відкритий/ Безкоштовний доступ	Частково	Ні	Частково	Частково	Так

Детальний аналіз існуючих кулінарних сервісів дозволяє виявити низку обмежень, що знижують рівень їхньої персоналізації та адаптивності до потреб конкретного користувача. Більшість сучасних платформ забезпечують пошук і рекомендацію рецептів на основі заздалегідь визначених категорій та фільтрів, однак не враховують комплексно індивідуальні смакові вподобання, наявність інгредієнтів і медичні обмеження користувача.

Відкриті бази рецептів та кулінарні спільноти надають значний обсяг контенту, проте здебільшого використовують традиційні механізми пошуку за ключовими словами та тегами. У результаті користувач змушений самостійно аналізувати велику кількість рецептів щодо їх відповідності власним потребам. Сервіси доставки харчових наборів орієнтовані насамперед на спрощення процесу вибору та приготування страв, однак мають обмежені можливості персоналізації. Наявні механізми врахування дієтичних обмежень зазвичай реалізуються на рівні формування меню і не передбачають гнучкої адаптації окремих інгредієнтів. Мобільні застосунки для контролю харчування забезпечують ефективний облік калорійності та поживної цінності продуктів, проте їх функціональність переважно спрямована на моніторинг споживання їжі, а не на підтримку процесу вибору та приготування страв. Отже, результати аналізу підтверджують доцільність

розробки інтелектуального вебзастосунку, який поєднуватиме персоналізований підбір рецептів, урахування наявних інгредієнтів, контроль дієтичних обмежень та можливості генерації й адаптації рецептів із використанням технологій штучного інтелекту.

1.3 Огляд технологій та методів штучного інтелекту для вирішення задачі

1.3.1 Рекомендаційні системи та інтелектуальне профілювання користувачів

Успішна робота сучасних інтелектуальних кулінарних вебплатформ безпосередньо зумовлена ефективністю блоку персоналізації, який відсіює зайву інформацію відповідно до запитів конкретної людини. Процес адаптації контенту під користувача базується на створенні математичних моделей його смаків та їх подальшому порівнянні з параметрами наявних страв. Для вирішення завдання індивідуального підбору зазвичай застосовують кілька класичних підходів до побудови рекомендаційних алгоритмів.

Першим базовим методом є колаборативна фільтрація [4,5], яка досліджує поведінкові фактори та оцінки схожих за інтересами людей. Цей інструмент демонструє високу математичну точність при пошуку неочевидних зв'язків у великих базах даних. Наприклад, якщо група юзерів, що часто обирає певну категорію страв, також полюбляє конкретні приправи чи соуси, алгоритм автоматично перенесе цю закономірність на інших людей із подібними смаками. Проте колаборативна модель потребує великої кількості початкової статистики про дії користувачів. Через це проблема «холодного старту» [6] стає серйозною перешкодою для нового вебдодатка, де матриця взаємозв'язків між юзерами та рецептами на перших етапах роботи є майже порожньою. Як наслідок, спиратися лише на цей підхід на

старті проєкту недоцільно, оскільки система не зможе видавати релевантні підказки новачкам.

Щоб обійти цю проблему, використовують контентну фільтрацію [7,8], яка досліджує безпосередні параметри самих страв (складники, тип кухні, тривалість та складність готування) і зіставляє їх із персональними уподобаннями людини. Головний технологічний плюс контентного методу – його повна автономність від дій інших відвідувачів сайту чи історії оцінок. Це робить його ідеальним рішенням для щойно створених застосунків із невеликою аудиторією. Оскільки алгоритм орієнтується на внутрішні метадані об'єктів, він здатний формувати індивідуальну стрічку рекомендацій одразу після реєстрації користувача на основі вибраних ним параметрів.

У нашому проєкті ідея інтелектуального профілювання суттєво розширена завдяки впровадженню багатовимірного вектора кулінарних уподобань. Користувачі мають можливість гнучко налаштовувати у відсотках або у вигляді бінарних перемикачів ключові смакові маркери: солодке, кисле, солоне та гостре. Окрім цих динамічних параметрів, інтелектуальний профіль містить жорсткі, незмінні обмеження (медичні протипоказання, алергії, релігійні чи дієтичні правила). Такі обмеження працюють як першочергові фільтри, що відсікають невідповідний або небезпечний контент ще під час попереднього аналізу запиту, тобто до початку інтелектуального сортування та семантичного пошуку.

Враховуючи специфіку кулінарної сфери, найкращі результати на практиці показують гібридні системи, які об'єднують переваги обох згаданих методів [11,12]. Зокрема, у цій роботі розроблено оригінальну гібридну рекомендаційну модель, що базується на математичному векторному представленні продуктів та комплексних профілях користувачів [9,10]. Під час «холодного старту» сервіс працює як контентний фільтр, що оцінює семантичну близькість векторів інгредієнтів та профілю людини. У міру розширення бази даних, реєстрації нових учасників та накопичення відгуків (лайків, збережених страв, активацій кулінарних таймерів) платформа плавно

переходить до колаборативної фільтрації. Такий комбінований підхід гарантує високу точність, гнучкість та безпеку гастрономічних рекомендацій на будь-якому етапі масштабування вебзастосунку.

1.3.2 Великі мовні моделі для генерації та адаптації рецептів

Використання сучасного генеративного штучного інтелекту в кулінарній сфері дає змогу ефективно розв'язати проблему статичності та певного обмеження класичних реляційних баз даних. Це, власне, перетворює стандартний пошук рецептів на складний творчий процес інтерактивного проєктування та синтезу абсолютно нових, унікальних гастрономічних страв [13,14]. Безперечно, великі мовні моделі відкрили принципово нові й широкі можливості [15] для створення чітко структурованого тексту, зокрема детальних кулінарних рецептів, технологічних карт та покрокових інструкцій з урахуванням динамічних контекстних обмежень, які виставляє конкретний користувач.

Сучасні архітектури нейромереж типу «трансформатор» [16,17,18], що використовують ефективні механізми самоуваги [19], здатні дуже глибоко інтерпретувати складні, багатокомпонентні запити користувачів, написані звичайною природною мовою. Вони повністю зважають на синтаксичні та семантичні зв'язки між різними кулінарними концептами, що зі свого боку дозволяє їм видавати релевантні, детерміновані та структуровані відповіді у популярному форматі JSON. Наявність такої суворої структури даних є критично важливою умовою для автоматичного парсингу, валідації та подальшої серіалізації кулінарного контенту на серверній стороні вебзастосунку без ризику появи синтаксичних помилок під час рендерингу інтерфейсу. Для нашого інтелектуального вебзастосунку практичне використання потенціалу великих мовних моделей охоплює три ключові архітектурні сценарії:

Динамічна генерація рецептів «з нуля» – процес створення унікальних, раніше не існуючих у базі даних кулінарних інструкцій та технологічних карт на основі незвичних поєднань багатовимірних смакових характеристик користувача та його специфічних текстових побажань природною мовою.

Адаптація та реконфігурація у випадку нестачі інгредієнтів у модулі «Мій холодильник» – якщо для обраної користувачем страви бракує певних компонентів, мовна модель виступає в ролі автономного інтелектуального агента. Вона автоматично модифікує рецептуру, знаходячи та замінюючи відсутні інгредієнти релевантними альтернативними аналогами із обов'язковим збереженням балансу поживних речовин (КБЖУ), консистенції, теплової карти приготування та фінальних смакових властивостей страв [21].

Генерація інтерактивного покрокового інтерфейсу приготування – процес автоматичного формування структурованих JSON-документів для логічного розбиття загального кулінарного процесу на окремі мікро-етапи. При цьому модель чітко виділяє конкретні кулінарні дії, розраховує тривалість термічної або механічної обробки та забезпечує автоматичну ініціалізацію програмних таймерів зворотного відліку на стороні клієнтського інтерфейсу.

Для локального розгортання великих мовних моделей та розв'язання задачі повної автономності системи без необхідності інтеграції коштовних чи нестабільних зовнішніх хмарних API (таких як OpenAI або Anthropic) [22], у роботі обрано та проаналізовано спеціалізовану платформу Ollama [23]. Даний інструмент запускається як локальний фоновий HTTP-сервер і забезпечує уніфікований REST API інтерфейс для гнучкої взаємодії з різними мовними моделями з відкритим вихідним кодом, такими як Llama 3, Mistral або Phi-3 [20]. Таке архітектурне рішення на рівні бекенду гарантує абсолютну конфіденційність персональних даних користувачів, нульову вартість експлуатації системи (відсутність плати за токени), мінімальні

затримки при генерації відповідей та повну незалежність вебплатформи від зовнішніх сервісів, інтернет-з'єднання чи географічних обмежень.

1.3.3 Векторні бази даних та семантичний пошук інгредієнтів

Класичні реляційні бази даних разом із традиційними алгоритмами повнотекстового пошуку (такими як оператори LIKE, ILIKE або базові індекси B-Tree в SQL-системах) виявляються неефективними, коли потрібно реалізувати гнучкий, контекстно-залежний підбір об'єктів. Це особливо відчутно, якщо кінцеві користувачі шукають продукти, використовуючи синоніми, розмовні форми, сленгові вирази чи розлогі описові фрази. Старі системи орієнтуються суто на точний збіг символів у рядку, що сильно обмежує функціонал сучасних кулінарних сервісів. Для розв'язання цієї проблеми та впровадження швидкого пошуку рецептів за їхнім безпосереднім смисловим наповненням, у сучасній розробці застосовують математичні векторні представлення тексту [24,25], а також спеціалізовані векторні бази даних або відповідні плагіни до них.

Ця технологія дозволяє переносити довільні текстові описи продуктів, складників чи цілих інструкцій із приготування страв у багатовимірний метричний простір за допомогою попередньо навчених нейромережевих моделей (текстових енкодерів). У такому математичному середовищі кожному об'єкту відповідає щільний вектор дійсних чисел із фіксованою розмірністю. Головна перевага цього підходу полягає в тому, що відстань між двома векторами у багатовимірному просторі прямо вказує на тематичну, логічну та гастрономічну близькість між поняттями з реального світу.

Наприклад, коли користувач пише в пошуковому рядку слово «томати», семантичний алгоритм через розрахунок косинусної або евклідової відстані між векторами дозволяє системі миттєво знайти й видати рецепти, де

згадуються «помідори», «черрі» чи «томатна паста». Це відбувається навіть тоді, коли точного посимвольного збігу із початковим запитом у базі даних взагалі немає. З погляду математики, цей процес виглядає як мінімізація кута між векторами відповідних концептів у семантичному просторі, що відображає їхній спільний контекст використання в кулінарії.

У нашому інтелектуальному застосунку векторний пошук та збереження ембеддінгів організовано за допомогою інтегрованого розширення pgvector [26] для хмарної платформи Supabase, яка працює на базі СКБД PostgreSQL. Таке інженерне рішення дає можливість зберігати високовимірні векторні представлення інгредієнтів (наприклад, із розмірністю 1536 елементів) безпосередньо всередині звичайних реляційних таблиць разом із традиційними текстовими чи числовими полями.

Завдяки pgvector стає можливим швидке виконання операцій пошуку найближчих сусідів [27] із використанням оптимізованих індексів типу IVFFlat або HNSW прямо всередині стандартних SQL-запитів. Це повністю позбавляє розробників необхідності розгортати, адмініструвати та постійно синхронізувати окремі ізольовані векторні сховища (як-от Pinecone чи Milvus). Такий крок суттєво спрощує загальну архітектуру системи, заощаджує серверні ресурси, скорочує час очікування при обробці запитів у модулі «Мій холодильник», а також гарантує стабільність транзакцій та цілісність інформації на платформі. Впровадження цього підходу дозволяє сфокусуватися на вдосконаленні логіки додатка, не витрачаючи сили на підтримку сторонніх сервісів.

1.3.4 Вебтехнології для розробки інтерактивної кулінарної платформи

Створення сучасного інтелектуального вебдодатка з високим рівнем інтерактивності та миттєвим оновленням даних вимагає використання

прогресивного й добре збалансованого технологічного стеку. Обрані інструменти мають ефективно підтримувати асинхронну взаємодію, оптимізувати трафік і швидко обробляти великі неструктуровані масиви даних, що надходять від модулів штучного інтелекту. Специфіка кулінарного напрямку (одночасний запуск кількох таймерів, обробка складних JSON-файлів із рецептами, миттєве відсікання алергенів) висуває серйозною вимоги до побудови як клієнтської, так і серверної частин платформи.

React є популярною JavaScript-бібліотекою з відкритим кодом [28], призначеною для створення користувацьких інтерфейсів на фронтенді. Завдяки компонентному підходу вона дозволяє розділити складну структуру додатка на окремі модулі, які можна використовувати повторно. Це дає змогу реалізувати плавний і зручний покроковий алгоритм готування страв без постійного перезавантаження сторінок у браузері. Технологія Virtual DOM суттєво знижує навантаження на систему під час відтворення елементів інтерфейсу. Водночас архітектура односпрямованого потоку даних у поєднанні з актуальними інструментами state-менеджменту спрощує контроль і синхронізацію роботи багатьох кулінарних таймерів, графічних індикаторів прогресу та елементів гейміфікації в режимі реального часу.

FastAPI використовується як швидкісний асинхронний вебфреймворк на базі Python, за допомогою якого створюється стабільний бекенд і будується архітектура REST API [29]. Цей інструмент спирається на сучасні стандарти Starlette та Pydantic [30], що забезпечує найвищу швидкість обробки HTTP-запитів серед усього Python-сегмента завдяки повноцінній підтримці асинхронності. Важливими плюсами фреймворку є автоматична перевірка, серіалізація та десеріалізація даних через Pydantic-моделі, а також генерація інтерактивної документації OpenAPI у фоновому режимі. Такий підхід мінімізує ризик виникнення помилок у типах даних через людський фактор, гарантує коректне передавання параметрів до нейромережевих модулів і значно прискорює тестування та розгортання всього інтерфейсу програмування.

Supabase представляє собою сучасну хмарну BaaS-платформу (Backend-as-a-Service), створену на основі реляційної СКБД PostgreSQL [31]. Вона надає розробнику вже налаштовану й надійну систему автентифікації користувачів за протоколом JWT, гнучкі правила розмежування доступу на рівні окремих рядків чи таблиць, а також зручний інструментарій для прямої взаємодії з базою з боку клієнта, якщо в цьому є потреба. Крім того, як зазначалося вище, вбудоване розширення pgvector гарантує безпечне збереження ембеддінгів та швидкий пошук схожих інгредієнтів у межах однієї системи, позбавляючи розробників необхідності підключати сторонні бази даних.

Ollama виступає в архітектурі проєкту як спеціалізоване середовище для локального розміщення, керування та запуску великих мовних моделей із відкритим кодом. Ця платформа повністю бере на себе апаратні обчислення (взаємодію з відеокартою чи процесором) і надає серверу FastAPI чіткий, уніфікований REST API для відправки промптів та отримання стабільних відповідей у форматі JSON. Це значно спрощує завантаження, оновлення й конфігурацію вагових коефіцієнтів нейромереж, забезпечуючи фіксований інтерфейс взаємодії незалежно від того, яка саме модель (Llama 3, Mistral чи інша) розгорнута на локальному сервері в конкретний момент.

Поєднання цих інструментів у межах класичної трирівневої клієнт-серверної архітектури дозволяє створити швидкий, автономний та масштабований вебзастосунок. Проєкт повністю задовольняє вимоги щодо безпеки даних та гарантує комфортну взаємодію користувача з інтелектуальними модулями штучного інтелекту. Усі елементи системи функціонують як єдиний злагоджений механізм, що забезпечує стабільну роботу платформи навіть під час постійного оновлення кулінарної бази даних та дозволяє легко обробляти будьякі щоденні запити юзерів. У підсумку, такий вибір архітектурних рішень дозволяє розгорнути повноцінний програмний комплекс на звичайному локальному обладнанні без потреби в дорогих серверних потужностях. Це робить проєкт максимально гнучким для

подальшої модернізації та повністю готовим до впровадження нових інтелектуальних функцій у майбутньому.

1.4 Постановка задачі

Персоналізований підбір кулінарних рецептів є надзвичайно актуальним завданням у контексті сучасного розвитку інтелектуальних вебсистем та індустрії здорового харчування. Швидкий темп життя та зростання інтересу суспільства до індивідуальних дієт вимагають створення якісно нових цифрових помічників. Проте існуючі на ринку рішення не забезпечують повноцінної інтеграції методів штучного інтелекту для гнучкого врахування індивідуальних смакових уподобань, динамічної генерації нових оригінальних рецептів та забезпечення зручного, інтерактивного інтерфейсу безпосередньо під час приготування страв. Більшість із них пропонують лише стандартний пошук за ключовими словами, що значно обмежує можливості кінцевого користувача.

Об'єктом роботи є процес персоналізованого пошуку та генерації кулінарних рецептів на основі смакових уподобань користувача з урахуванням наявних інгредієнтів та суворих дієтичних обмежень [32, 33]. Дослідження охоплює як математичні моделі представлення смаків, так і механізми адаптації текстової кулінарної інформації під конкретні запити.

Метою роботи є розробка високотехнологічного вебзастосунку, що дозволяє користувачу отримувати максимально релевантні рецепти зі сховища або самостійно генерувати нові кулінарні інструкції на основі детально сформованого профілю смакових уподобань із застосуванням сучасних технологій штучного інтелекту.

Для успішного досягнення поставленої мети в межах кваліфікаційної роботи необхідно вирішити такі конкретні завдання:

– проаналізувати сучасний стан предметної галузі персоналізованих кулінарних сервісів, вивчити досвід закордонних розробок та виявити ключові недоліки існуючих рішень;

– дослідити математичні методи рекомендаційних систем, алгоритми векторизації текстів та великих мовних моделей, що є придатними для швидкої генерації, точного семантичного пошуку та адаптації рецептів;

– обрати, порівняти та науково обґрунтувати оптимальний набір інструментів та програмних технологій для подальшої розробки серверної та клієнтської частин вебзастосунку;

– розробити модуль формування детального профілю користувача з урахуванням індивідуальних дієтичних обмежень, хронічних алергій та багатовимірних числових смакових дескрипторів;

– реалізувати інтелектуальний модуль «Мій холодильник» для швидкого персоналізованого пошуку та гнучкої адаптації кулінарних рецептів на основі фактично наявних у людини інгредієнтів;

– реалізувати модуль генерації нових рецептів та покрокових інтерактивних кулінарних інструкцій з використанням локальної великої мовної моделі;

– розробити та об'єднати в єдину екосистему клієнтський та серверний компоненти створеного вебзастосунку;

– провести комплексне багаторівневе тестування розробленого застосунку для перевірки його стабільності та безпеки, а також розробити вичерпний ілюстрований посібник користувача для експлуатації системи.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ

2.1 Вибір та обґрунтування технологічного стеку

Визначення технологічного стеку є одним із найважливіших архітектурних етапів під час проєктування вебзастосунку. Саме цей вибір безпосередньо впливає на майбутню продуктивність, можливості масштабування, швидкість написання коду та простоту підтримки системи в довгостроковій перспективі. Якщо говорити про розробку інтелектуальних кулінарних платформ, то їхня архітектура має стабільно витримувати специфічні навантаження, пов'язані з асинхронним обміном даними з мовними моделями. Крім того, важливо гарантувати захист персональних даних користувачів і підтримувати миттєве оновлення елементів фронтенду в режимі реального часу. Під час підбору інструментів для реалізації проєкту «Smart Recipes» ми спиралися на такі ключові критерії: повна відповідність технічному завданню, стабільність і якість документації, доступність безкоштовного використання, активна підтримка з боку ком'юніті розробників та хороша сумісність усіх технологій між собою.

Серверну частину додатка було вирішено реалізувати мовою Python [34], взявши за основу вебфреймворк FastAPI версії 0.110.x. Вибір на користь Python є цілком очевидним, зважаючи на його абсолютне домінування в індустрії штучного інтелекту та машинного навчання. Більшість популярних бібліотек для інтеграції з LLM (наприклад, LangChain або OpenAI SDK) [35], як і інструменти для роботи з векторними сховищами, створюються насамперед під цю мову. У свою чергу, FastAPI забезпечує швидку асинхронну обробку HTTP-запитів на базі стандарту ASGI. Він також пропонує автоматичну валідацію вхідних даних через Pydantic-моделі та самостійно генерує інтерактивну документацію OpenAPI, що неабияк прискорює тестування ендпоінтів. Завдяки асинхронній природі фреймворку,

наш сервер може спокійно обробляти тривалі запити до нейромережевих модулів, не блокуючи при цьому інші операції введення-виведення.

Для створення клієнтської частини ми обрали бібліотеку React версії 18.x. Компонентна структура React чудово підходить для побудови покрокового інтерфейсу приготування страв із вбудованими таймерами, адже кожен окремий крок можна винести в ізольований компонент із власним станом. Використання вбудованих хуків useState та useEffect значно спрощує контроль за роботою лічильників часу та асинхронними зверненнями до API. Завдяки технології Virtual DOM інтерфейс може оновлювати стан індикаторів прогресу та таймерів без перезавантаження всієї сторінки сайту. Для збирання фронтенд-частини застосовується інструмент Vite [36]. Він забезпечує дуже швидке гаряче оновлення модулів під час розробки та оптимізує розмір підсумкових JavaScript-файлів для максимальної швидкодії додатка.

Для керування даними було обрано платформу Supabase, яка є сучасним хмарним BaaS-рішенням на базі PostgreSQL з відкритим кодом. Supabase пропонує розробникам готовий модуль автентифікації з підтримкою JWT-токенів, механізм RLS для захисту інформації на рівні окремих рядків, а також зручний JS-клієнт для прямої взаємодії з базою. Окремим плюсом є розширення pgvector, яке дозволяє відмовитися від розгортання окремої векторної бази даних. З його допомогою ми можемо зберігати високовимірні ембеддінги кулінарних інгредієнтів безпосередньо у звичайних реляційних таблицях поруч із іншими атрибутами. Це сильно спрощує загальну архітектуру та гарантує швидкий семантичний пошук продуктів за синонімами.

Локальний запуск великих мовних моделей організовано за допомогою платформи Ollama. Вона працює як фоновий HTTP-сервер і надає REST API для взаємодії з різними open-source моделями (такими як Llama, Mistral, Phi тощо). Використання Ollama замість інтеграції платних хмарних API дозволяє забезпечити повну конфіденційність призначених для користувача

даних, виключає будь-які операційні витрати на запити та дає системі можливість працювати навіть без доступу до інтернету. Такий крок повністю нівелює фінансові ризики, пов'язані з оплатою токенів хмарним провайдерам, і робить інтелектуальне ядро застосунку абсолютно автономним.

Безпека та автентифікація в додатку базуються на стандарті JWT. Токени створюються на стороні сервера після успішної авторизації, після чого передаються на клієнт, де зберігаються в localStorage і додаються в заголовок Authorization при кожному наступному запиті. Перевірка цих токенів здійснюється за допомогою спеціального middleware на стороні FastAPI перед виконанням захищених операцій. Це дозволяє створити надійний периметр безпеки для захисту особистих кабінетів користувачів та їхніх карт алергенів.

Комунікація між усіма компонентами системи побудована через архітектуру REST API. Фронтенд відправляє HTTP-запити до FastAPI на бекенді, а сервер, залежно від логіки, звертається або до Supabase (для читання/запису даних), або до Ollama (для генерації нових рецептів). Усі відповіді передаються у стандартному форматі JSON, що гарантує швидку й легку десеріалізацію об'єктів на стороні клієнта.

2.2 Проєктування архітектури системи

2.2.1 Загальна архітектура та діаграма компонентів

Розроблену систему побудовано на основі класичної трирівневої клієнт-серверної архітектури з чітким винесенням шару даних в окремий незалежний сегмент. Подібний архітектурний підхід дозволяє гнучко розмежувати зони відповідальності між різними модулями додатка, суттєво спрощує масштабування його елементів у майбутньому, підвищує загальну читабельність і підтримуваність коду, а також оптимізує процеси тестування та контролю безпеки. Окрім цього, такий розподіл зменшує взаємозалежність

компонентів, що дозволяє за потреби модернізувати зовнішній інтерфейс або серверну частину окремо один від одного. Це є надзвичайно важливим фактором при веденні розробки в межах сучасних ітеративних підходів.

Клієнтська частина платформи функціонує безпосередньо на рівні представлення (зовнішньому інтерфейсі). Вона повністю відповідає за відтворення графічного інтерфейсу користувача, перехоплення й обробку подій введення, первинну перевірку правильності заповнення форм, а також за забезпечення прямої інтерактивної взаємодії з відвідувачем сайту. У свою чергу, серверна частина (внутрішня логіка) відіграє роль основного рівня бізнес-логіки. Вона відповідає за реалізацію базових алгоритмів системи, маршрутизацію трафіку, обробку вхідних HTTP-запитів, перевірку прав доступу користувачів, а також за керування та пряму інтеграцію з локальними сервісами штучного інтелекту. Нарешті, рівень даних, розгорнутий на базі хмарного середовища Supabase/PostgreSQL, гарантує надійне й довготривале збереження всієї інформації, забезпечує транзакційну цілісність, захист записів на рівні окремих рядків та швидке виконання операцій семантичного векторного пошуку.

Загальний алгоритм взаємодії всіх компонентів системи організовано таким чином: кінцевий користувач працює з клієнтським React-застосунком через вікно браузера. Клієнтська частина комунікує з серверною за допомогою асинхронних REST API запитів, використовуючи для обміну об'єктами даних універсальний полегшений формат JSON. Сервер, отримавши запит, звертається до платформи Supabase через відповідний інструментарій об'єктно-реляційного відображення або прямі SQL-команди для зчитування чи запису інформації, а також паралельно взаємодіє з локальним інструментом Ollama для створення унікальних кулінарних інструкцій. Сервіс Ollama функціонує в локальному режимі на тому ж самому серверному обладнанні, де запущено вебфреймворк FastAPI, і приймає від нього вхідні структуровані запити через виділений внутрішній порт (рис. 2.1). Така локальна топологія мінімізує мережеві затримки під час

передачі великих масивів контекстних даних до штучного інтелекту. Це дозволяє досягти високої швидкості генерації відповідей, що позитивно впливає на загальне враження користувача від роботи з програмою.

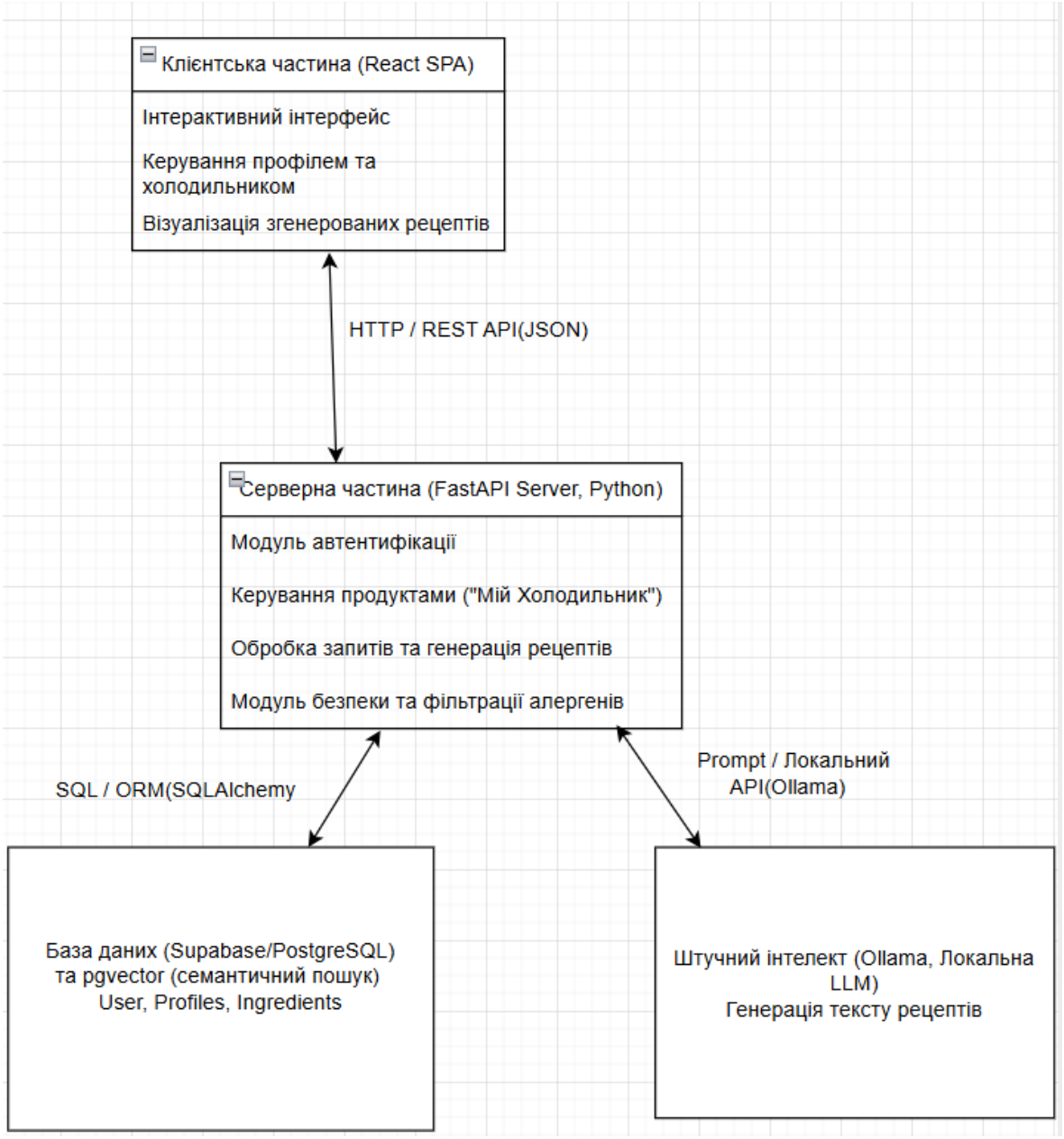


Рисунок 2.1 – Загальна архітектура системи «Smart Recipe»

Серверна частина організована за принципом мінімальної надмірності: уся бізнес-логіка, що не потребує прямої взаємодії з мовною моделлю, делегована хмарній платформі Supabase засобами Row Level Security та

вбудованого API. Серверний застосунок складається з трьох модулів: main.py (ініціалізація FastAPI-застосунку, налаштування CORS-політик та підключення маршрутизаторів), generate.py (інтеграція з Ollama, формування промпт-шаблонів та ендпоінт генерації рецептів за смаковим профілем), fridge.py (логіка модуля «Мій холодильник» – підбір страв із наявних продуктів із вибором оптимальної комбінації інгредієнтів).

Клієнтська частина організована за компонентним принципом. Застосунок містить такі основні сторінки: LoginPage та RegisterPage (автентифікація), ProfileSetupPage (налаштування смакового профілю та алергенів), RecipesPage (перегляд та пошук рецептів), GeneratorPage (генерація рецептів за профілем), FridgePage (модуль «Мій холодильник»), CookingPage (покроковий інтерфейс приготування з таймерами).

2.2.2 Проєктування бази даних

Для забезпечення надійного зберігання інформації, транзакційної цілісності, високої швидкості обробки реляційних запитів та можливості виконання математичних операцій над високовимірними векторами штучного інтелекту, рівень збереження даних розроблено на основі об'єктно-реляційної системи керування базами даних PostgreSQL [37], розгорнутої у хмарній екосистемі Supabase. Архітектура бази даних базується на принципах нормалізації (до третьої нормальної форми включно), що дозволяє мінімізувати надмірність даних та виключити ризик виникнення аномалій при використанні операцій додавання, читання, оновлення та видалення.

Концептуальна схема бази даних містить комплекс взаємопов'язаних сутностей, що відображають суворі бізнес-правила кулінарної ШІ-платформи. Схема складається з таких основних реляційних таблиць: auth.users (Користувачі), public.profiles (Профілі), public.recipes (Рецепти),

public.saved_recipes (Збережені рецепти) та public.user_fridges (Холодильники користувачів).

Auth.users (Користувачі) – системна таблиця автентифікаційного ядра, яка містить унікальний глобальний ідентифікатор користувача, його електронну пошту, зашифрований пароль та системні часові мітки створення облікового запису.

Public.profiles (Профілі) – таблиця розширених профілів користувачів, що пов'язана із сутністю користувачів зв'язком «один-до-одного» за допомогою первинного та водночас зовнішнього ключа із правилом каскадного видалення при знищенні облікового запису. Вона зберігає ім'я, прізвище користувача, багатовимірний смаковий вектор уподобань у формі оптимізованого бінарного об'єкта, а також ізольовані структуровані масиви зафіксованих харчових алергенів та дієтичних обмежень.

Public.recipes (Рецепти) – центральна сутність системи, яка містить ідентифікатор страви, назву, опис, час приготування, рівень складності, а також масиви інгредієнтів та кроків приготування у гнучкому та оптимізованому для пошуку форматі бінарного об'єкта. Таблиця також містить дескриптори поживної цінності (білки, жири, вуглеводи, калорійність) та спеціалізований стовпець векторного вбудовування для виконання інтелектуальних операцій.

Public.saved_recipes (Збережені рецепти) – таблиця, що реалізує логіку відношення «багато-до-багатьох» між користувачами та рецептами. Вона фіксує унікальні ідентифікатори збережених страв, зовнішній ключ на обліковий запис користувача, назву, структуру даних рецепта та дату додавання до приватної кулінарної колекції.

Public.user_fridges (Холодильники користувачів) – таблиця, яка забезпечує функціонування інтелектуального модуля «Мій холодильник». Сутність містить первинний ключ, зовнішній ключ для зв'язку з конкретним профілем, текстову назву кулінарного компонента, його категорію харчової

класифікації, кількість або об'єм продукту та мітку часу внесення інгредієнта до системи.

Детальні зв'язки між сутностями, типи первинних та зовнішніх ключів, а також обмеження цілісності представлені на розробленій діаграмі сутностей та зв'язків (рис. 2.2). SQL-скрипти створення усіх таблиць бази даних із визначенням типів даних, індексів та обмежень цілісності наведено у Додатку А.

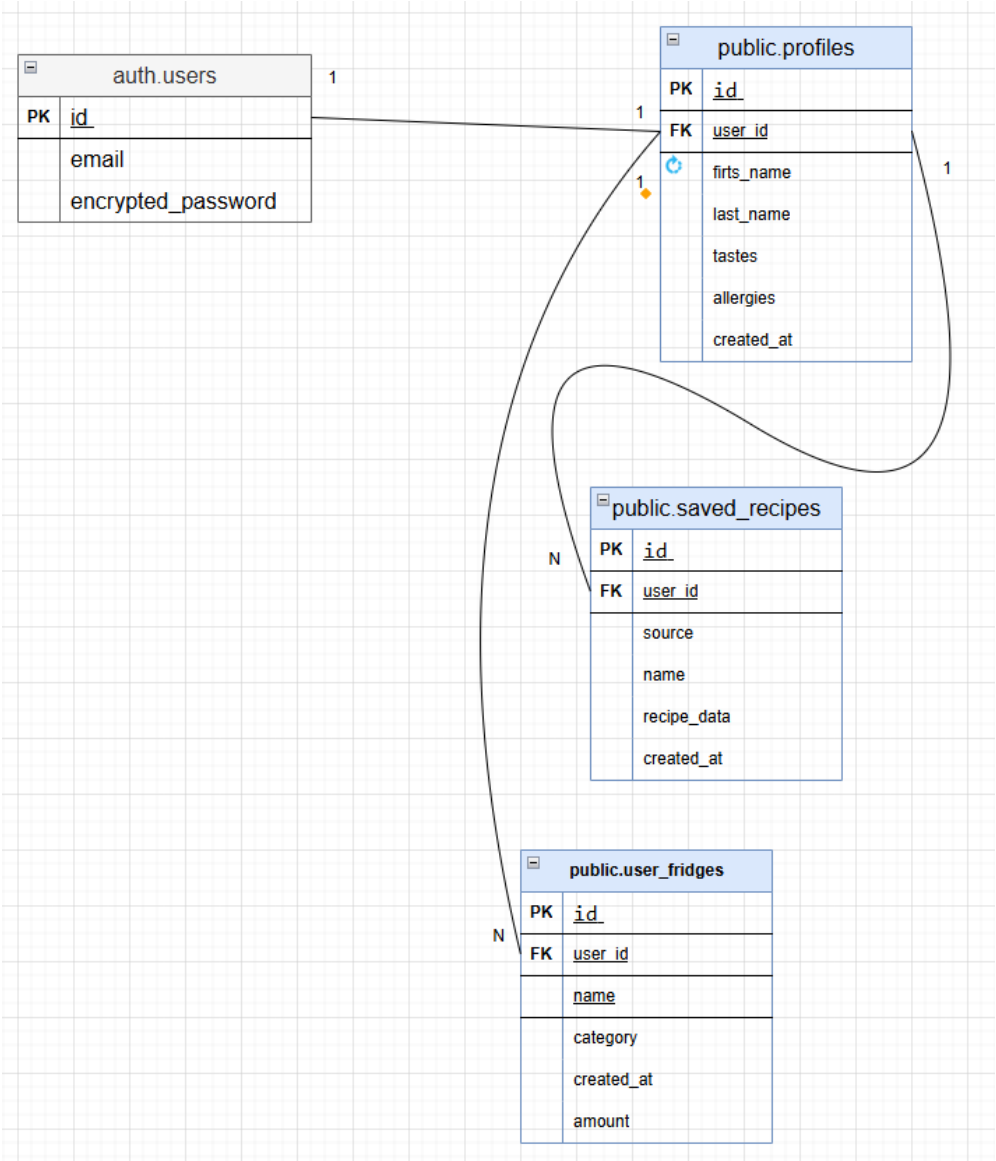


Рисунок 2.2 – ER-діаграма бази даних системи

Критично важливим елементом архітектури бази даних є інтеграція розширення векторного типу, яке додає до реляційної системи PostgreSQL можливість роботи з математичними векторними об'єктами. Спеціальний стовпець векторного вбудовування таблиці рецептів має тип фіксованого вектора на 1536 елементів, що дозволяє зберігати високовимірні числові характеристики рецептів та інгредієнтів, отримані через штучний інтелект.

Така структура даних забезпечує надійне збереження складних семантичних зв'язків між різними кулінарними компонентами системи у єдиному інформаційному просторі. Це дозволяє ефективно масштабувати базу даних без втрати точності під час обробки великих масивів інформації.

Для оптимізації швидкості виконання семантичного пошуку та обчислення подібності продуктів у модулі «Мій холодильник», над цим стовпцем розгорнуто інвертований індекс типу «плаский файл». Цей індекс розбиває математичний простір на окремі зони близькості та забезпечує надшвидкий пошук за семантичною схожістю за допомогою операції обчислення косинусної відстані, що значно прискорює повернення результату порівняно зі звичайним повним скануванням таблиці.

Окрему увагу в процесі проєктування рівня збереження даних приділено безпеці та захисту медичних профілів користувачів. На кожній таблиці, яка містить персональну або приватну інформацію (профілі, збережені рецепти, холодильники), на рівні ядра бази даних активовано механізм безпеки на рівні окремих рядків. Завдяки налаштованим декларативним політикам безпеки, кожен автентифікований користувач через перевірку криптографічного ідентифікатора отримує ізольований доступ виключно до власних рядків даних, що унеможливлює несанкціоноване читання чи підміну чужих записів. Таблиця глобальних кулінарних рецептів є публічно доступною для швидкого читання усіма клієнтами, проте будь-які операції запису, модифікації або видалення рядків заблоковані для звичайних користувачів і можуть здійснюватися виключно сервісною роллю з боку захищеного сервера.

2.2.3 UML-діаграми прецедентів та класів

Для формалізації функціональних вимог до системи, визначення меж програмного продукту та детального опису взаємодії між зовнішніми об'єктами й внутрішніми модулями застосунку розроблено діаграму прецедентів згідно з міжнародним стандартом уніфікованої мови моделювання UML 2.5. Об'єктно-орієнтований аналіз на основі прецедентів дозволяє чітко зафіксувати поведінку системи з погляду кінцевого користувача та спроєктувати логіку розподілу прав доступу.

Розроблена діаграма відображає взаємодію двох ключових акторів, які мають чітке розмежування за рівнем доступу до інтелектуальних ресурсів платформи: «Неавторизований користувач» та «Авторизований користувач».

Неавторизований (або незареєстрований) користувач є актором із мінімальним рівнем доступу. Його взаємодія з вебзастосунком обмежена базовими прецедентами, необхідними для створення безпечного сеансу, а саме: переглядом головної сторінки, реєстрацією нового облікового запису та автентифікацією (входом) до системи. Після успішного підтвердження облікових даних цей актор переходить у статус авторизованого користувача.

Окрему увагу на діаграмі приділено специфічним зв'язкам між прецедентами. Зокрема, процес штучного інтелектуального підбору страв обов'язково містить у собі відношення включення для прецеденту фільтрації алергенів за профілем. Це означає, що запуск генерації автоматично й беззастережно активує підсистему безпеки харчування.

Водночас прецеденти управління особистим холодильником та генерації нових рецептів пов'язані з прецедентом збереження страви до обраної колекції за допомогою відношення розширення. Це підкреслює, що додавання до обраного є необов'язковою дією, яка виконується лише за бажанням користувача після успішного завершення основного процесу підбору. Таке логічне розділення функцій дозволяє користувачеві самостійно вирішувати, які саме кулінарні ідеї заслуговують на місце в його

довгостроковій персональній базі даних. Повний опис взаємодій акторів із межами системи наведено на діаграмі прецедентів (рис. 2.3).

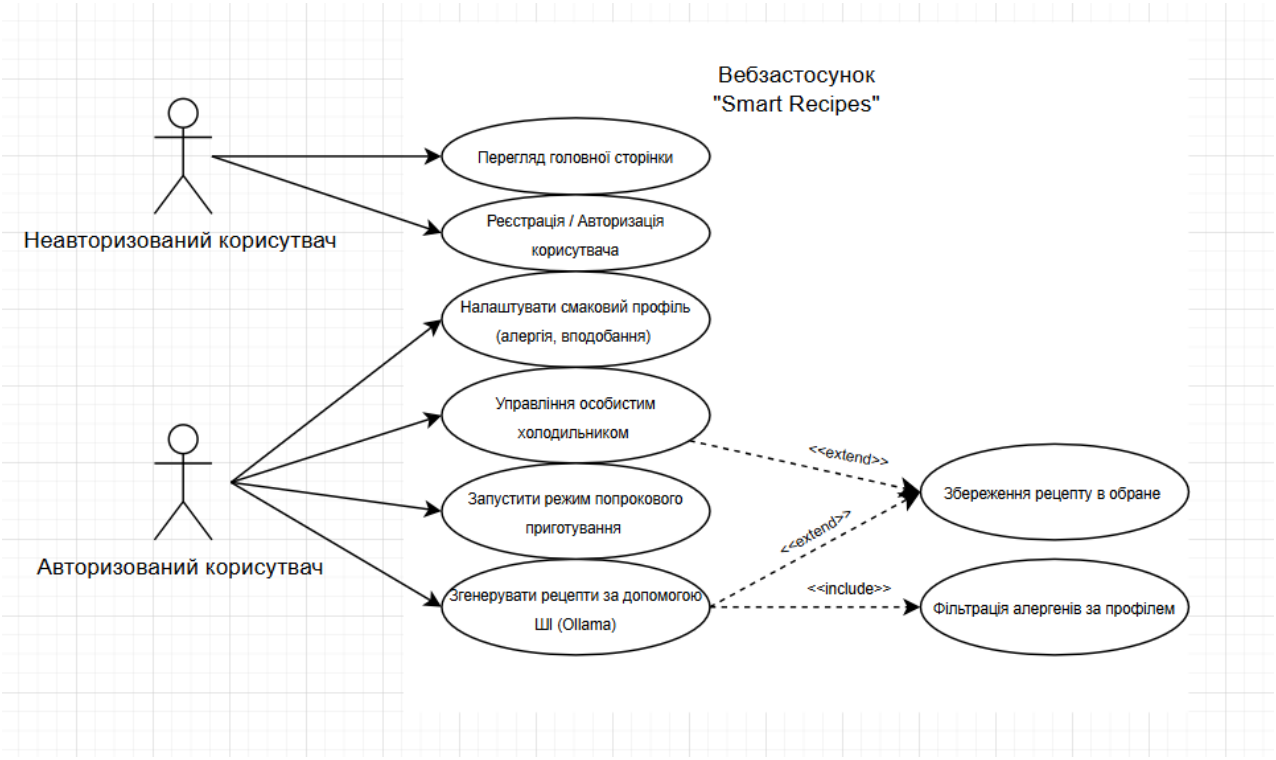


Рисунок 2.3 – UML-діаграма прецедентів системи Smart Recipe

Для відображення статичної структури системи, визначення архітектурних сутностей, їхніх внутрішніх властивостей та логічних взаємозв'язків на рівні програмного коду розроблено діаграму класів. Вона описує об'єктно-орієнтовану модель вебзастосунку та визначає типи даних атрибутів і сигнатури методів для ключових компонентів системи. У межах розробленої архітектури було спроєктовано та детально розписано ключові програмні класи системи, детальний опис та призначення яких наведені далі.

User (Користувач) – відображає облікові дані авторизованої особи. Клас містить унікальний числовий ідентифікатор, електронну пошту та зашифрований пароль. До його основних методів належать функція перевірки справжності користувача та функція ініціалізації оновлення профілю.

UserProfile (Профіль користувача) – безпосередньо пов'язаний із класом користувача та містить розширену інформацію про його фізіологічні та смакові пріоритети. Атрибутами класу виступають багатовимірний числовий смаковий вектор, динамічні масиви медичних алергенів та вегетаріанських або дієтичних обмежень. Керування станом класу здійснюється через методи оновлення смакового вектора та додавання нового харчового тригера.

Recipe (Рецепт) – описує структуру кулінарної страви. До його атрибутів належать унікальний ідентифікатор, назва, текстові масиви інгредієнтів із зазначенням ваги, послідовні кроки технологічної карти у бінарному форматі обміну даними та векторне числове представлення для семантичного пошуку. Клас надає методи для отримання детальної інформації про страву та вилучення списку покрокових інструкцій.

RecipeGenerator (Генератор рецептів) – інтелектуальний серверний клас, що координує взаємодію з великою мовною моделлю штучного інтелекту. Він не має постійних реляційних атрибутів і реалізує методи динамічної генерації страв на основі смакового вектора та адаптації існуючих рецептур під наявний набір інгредієнтів.

FridgeModule (Модуль холодильника) – відповідає за логіку обліку продуктів та попередню обробку запитів користувача. Клас містить методи аналізу наявних у базі даних інгредієнтів та виконання математичного пошуку відповідних рецептів за косинусною відстанню векторних вбудовувань.

Зв'язки між класами відображають архітектурну логіку побудови системи: класи користувача та профілю перебувають у відношенні строгої композиції, оскільки профіль не може існувати без облікового запису, тоді як класи генератора та модуля холодильника взаємодіють із класом рецептів за допомогою асоціативних зв'язків, забезпечуючи гнучкість обробки даних на серверній стороні застосунку та дозволяючи системі стабільно функціонувати в умовах постійної зміни даних.

2.3 Розробка серверної частини

2.3.1 Структура проєкту та маршрутизація

Серверна частина вебзастосунку реалізована на мові Python з використанням високопродуктивного фреймворку FastAPI. Структура проєкту організована за принципом розмежування відповідальностей, що дозволяє ізолювати окремі компоненти бізнес-логіки та полегшує подальше супроводження коду. Завдяки такому підходу розробники можуть вносити зміни у роботу окремих модулів або додавати нові функції без ризику порушити стабільність усього застосунку. Це забезпечує високу надійність архітектури та робить процес розгортання серверних служб значно ефективнішим і передбачуванішим. Кожен модуль відповідає за конкретну функціональну область системи, забезпечуючи модульність та гнучкість розробки (рис. 2.4). Такий підхід до побудови архітектури спрощує процес тестування та дозволяє незалежно модифікувати логіку окремих сервісів без ризику порушити стабільність усього програмного комплексу в процесі його довгострокового масштабування та підтримки.

Головний файл main.py ініціалізує застосунок FastAPI, реєструє маршрутизатори та налаштовує CORS middleware для дозволу запитів від клієнтської частини. Усі доступні в системі маршрутизатори підключаються з відповідними префіксами: /api/auth (автентифікація користувачів), /api/recipes (операції з рецептами), /api/generate (ШІ-генерація нових страв), /api/fridge (модуль управління холодильником), /api/profile (профіль користувача та його налаштування).

Конфігурація застосунку зберігається у файлі config.py та завантажується із змінних середовища через бібліотеку python-dotenv для забезпечення належного рівня безпеки й конфіденційності. Ключові параметри конфігурації: SUPABASE_URL та SUPABASE_KEY (підключення до хмарної бази даних), OLLAMA_BASE_URL (адреса Ollama-сервера), JWT_SECRET_KEY та JWT_ALGORITHM (параметри генерації

токенів доступу), DEFAULT_LLM_MODEL (назва моделі LLM за замовчуванням для обробки запитів).

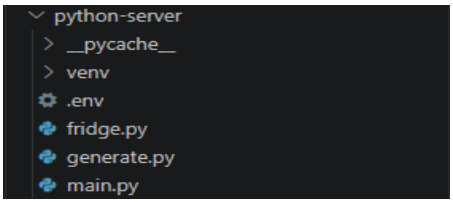


Рисунок 2.4 – Структура проєкту серверної частини FastAPI

2.3.2 Модуль профілю користувача та автентифікація

Система автентифікації реалізована на основі стандарту JWT [38] із терміном дії токена 24 години, що забезпечує баланс між зручністю користувача та безпекою сесії. JWT обрано замість сесійного підходу через відсутність необхідності зберігати стан сесії на сервері, що спрощує горизонтальне масштабування та відповідає архітектурі RESTful API. Такий вибір дозволяє серверній частині працювати значно швидше, оскільки їй не потрібно щоразу звертатися до бази даних лише для того, щоб перевірити статус користувача. Замість цього вся необхідна інформація про права доступу зашифрована всередині самого токена, який надійно зберігається на стороні клієнта.

При реєстрації пароль користувача ніколи не зберігається у відкритому вигляді – він хешується за адаптивним алгоритмом bcrypt із автоматично генерованою сіллю та зберігається у зашифрованому вигляді в таблиці профілів бази даних. Алгоритм bcrypt обрано через його стійкість до атак із використанням спеціалізованого обладнання (ASIC, GPU) завдяки навмисно ресурсомісткій функції хешування. Після успішної верифікації облікових даних під час входу сервер генерує JWT-токен, підписаний секретним ключем із змінної оточення, та повертає його клієнту у тілі відповіді.

Клієнтська частина зберігає токен у пам'яті застосунку та передає його в заголовку Authorization: Bearer <token> при кожному наступному запиті до захищених ресурсів.

Захист ендпоінтів реалізовано через механізм залежностей фреймворку FastAPI – функцію get_current_user(), яка оголошується як залежність для кожного захищеного маршруту. При надходженні запиту FastAPI автоматично викликає цю функцію до передачі керування обробнику ендпоінту: вона витягує токен із заголовка Authorization, верифікує цифровий підпис, перевіряє термін дії та декодує ідентифікатор користувача. У разі відсутності токена, його прострочення або невалідного підпису функція повертає HTTP-відповідь зі статусом 401 Unauthorized, блокуючи подальше виконання запиту. Такий підхід забезпечує централізований контроль доступу без дублювання логіки перевірки в кожному обробнику окремо.

Профіль користувача зберігає багатовимірний вектор смакових уподобань у форматі JSON. Числові поля sweetness, sourness, saltiness та spiciness приймають значення від 0 до 10, де 0 означає повну відсутність вподобання до відповідного смаку, а 10 – максимальну перевагу. Ці значення використовуються при формуванні промпту для LLM як кількісні обмеження: наприклад, значення spiciness = 2 транслюється в інструкцію «страва має бути практично без гостроті». Окремо зберігаються масив allergens із рядковими ідентифікаторами алергенів (наприклад, «Глютен», «Горіхи», «Молочні продукти») та масив dietary_restrictions із дієтичними обмеженнями (вегетаріанство, веганство, безглютенова дієта тощо). Наявність будь-якого алергену в профілі транслюється в категоричну заборону в системному промпті, що гарантує безпеку згенерованих рецептів для користувачів із харчовими алергіями. Усі поля профілю доступні для редагування користувачем у будь-який момент через відповідний розділ налаштувань застосунку, а зміни набувають чинності негайно при наступній генерації рецепту.

2.3.3 Модуль генерації рецептів на основі ШІ

Модуль генерації рецептів є ключовим компонентом серверної частини. Його логіку реалізовано у файлі generate.py. При отриманні запиту на генерацію модуль виконує такі кроки. Ця архітектурна побудова гарантує чітку та послідовну обробку кожної операції, що дозволяє уникнути збоїв під час високого навантаження на систему.

По-перше, отримання запиту та визначення мови. Ендпоінт POST /api/generate-recipe приймає об'єкт PromptRequest із текстовим промптом, який уже містить смаковий профіль користувача, сформований на стороні клієнта. Мова відповіді визначається автоматично на основі аналізу вхідного тексту. Завдяки такій автоматизації штучний інтелект розмовляє з користувачем тією ж мовою, якою було надіслано запит, що робить взаємодію максимально природною та комфортною.

По-друге, формування промпт-шаблону та запиту до Ollama. Функція build_generate_prompt() формує структурований промпт із системною інструкцією, вимогами до формату (JSON зі специфікованою схемою), правилами для назви, інгредієнтів, кроків приготування та розрахунку БЖУ. Модуль надсилає POST-запит до локального ендпоінту Ollama із зазначенням моделі та параметрів генерації (temperature, top_p, top_k, num_predict). Ретельне налаштування цих параметрів дозволяє збалансувати творчий потенціал штучного інтелекту та строгу точність технологічних карт страв.

По-третє, парсинг та нормалізація відповіді. Отримана відповідь проходить витягування JSON-блоку функцією extract_json() та десеріалізацію. Інгредієнти та кроки додатково нормалізуються до рядкового формату функціями normalize_ingredients() та normalize_steps(), що виключає некоректні типи даних у відповіді моделі. Цей етап є критично важливим для стабільності інтерфейсу, оскільки він відсікає будь-який текстовий «шум» і залишає лише чисті структуровані дані.

По-четверте, повернення результату клієнту. Сформований об'єкт рецепту повертається у відповіді API. Подальше збереження рецепту до бази даних Supabase та обчислення векторного представлення для семантичного пошуку виконується безпосередньо на стороні клієнтської частини засобами Supabase JavaScript SDK (рис. 2.5). Такий розподіл обчислювального навантаження між сервером та клієнтом дозволяє досягти високої швидкості відгуку всього застосунку.

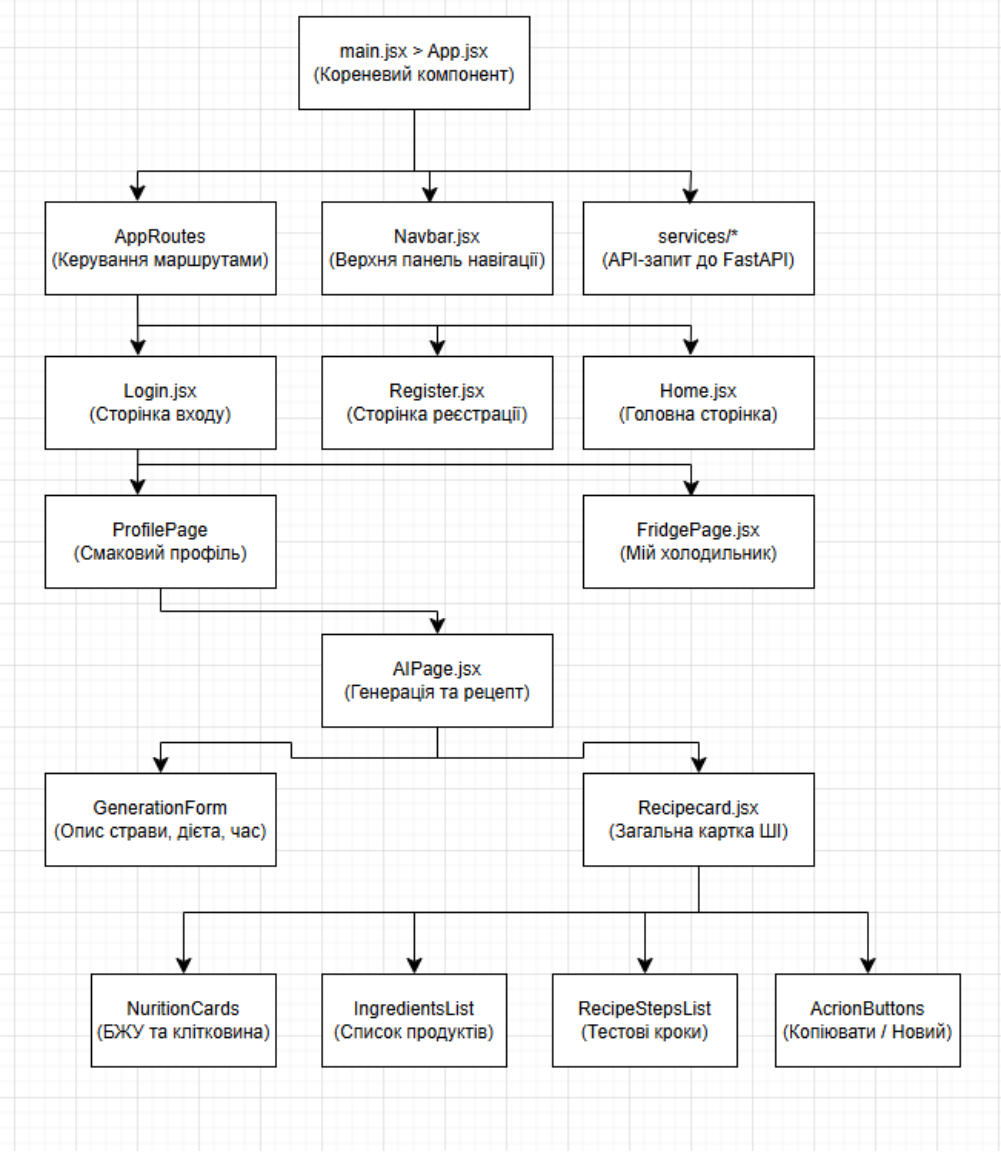


Рисунок 2.5 – Фрагмент реалізації модуля генерації рецептів

2.3.4 Модуль «Мій холодильник»

Модуль «Мій холодильник» дозволяє користувачу вказати перелік наявних інгредієнтів та отримати рецепти, для яких ці продукти є основними або можуть бути використані як замінники відсутніх компонентів.

Алгоритм роботи модуля включає декілька етапів. На першому етапі введені користувачем назви продуктів обробляються функцією семантичного пошуку: для кожного продукту обчислюється векторне представлення та виконується пошук у таблиці ingredients бази даних за косинусною відстанню. Це дозволяє коректно обробляти синоніми та різні форми назв продуктів.

На другому етапі знайдені ідентифікатори інгредієнтів використовуються для SQL-запиту, який повертає рецепти, що містять максимальну кількість наявних продуктів. Рецепти ранжуються за відсотком покриття інгредієнтів.

На третьому етапі для рецептів із неповним покриттям модуль передає до LLM запит на адаптацію рецепту: перелік наявних продуктів, список відсутніх інгредієнтів та профіль обмежень користувача. LLM генерує модифіковану версію рецепту з запропонованими замінниками.

2.4 Розробка клієнтської частини

2.4.1 Структура компонентів та навігація

Клієнтська частина вебзастосунку реалізована як Single Page Application на React 18 з використанням функціональних компонентів та хуків. Маршрутизація реалізована за допомогою бібліотеки React Router v6 [39], яка забезпечує декларативне визначення маршрутів та підтримку захищених сторінок. Це дозволяє створити динамічний інтерфейс, де перехід

між різними розділами програми відбувається миттєво і без перезавантаження всього сайту у браузері.

Управління глобальним станом застосунку реалізовано через Context API: AuthContext зберігає дані поточного користувача та токен автентифікації, ProfileContext – смаковий профіль та налаштування. Локальний стан компонентів управляється хуком useState, а побічні ефекти та запити до API – хуком useEffect (рис. 2.6).

Для здійснення HTTP-запитів до серверного API використовується бібліотека axios [40] із налаштованим перехоплювачем, який автоматично додає JWT-токен до заголовка кожного запиту та обробляє відповіді з помилками автентифікації (401). Завдяки такій автоматизації розробнику не потрібно вручну прописувати параметри безпеки для кожної окремої операції, що значно знижує ризик виникнення випадкових помилок при обміні даними між клієнтом та сервером.

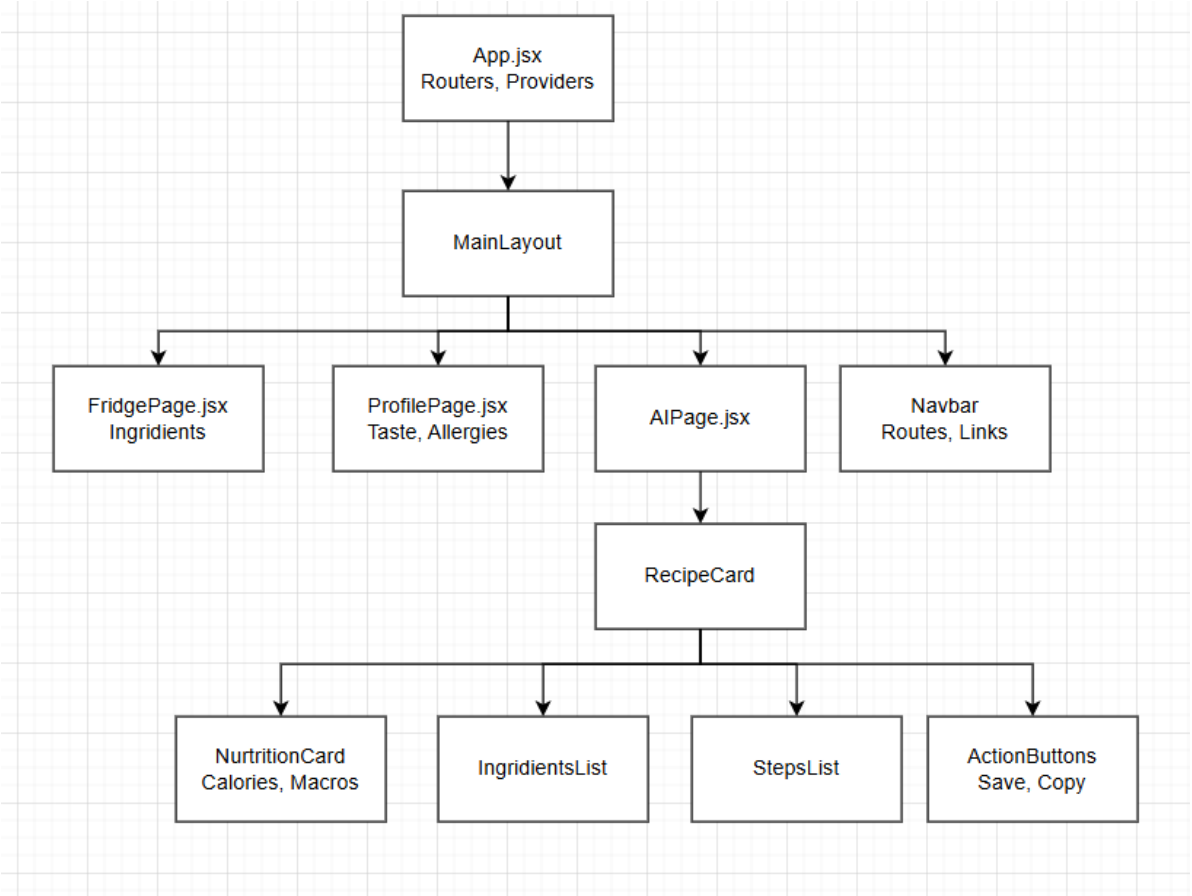


Рисунок 2.6 – Структура клієнтської частини React

2.4.2 Інтерфейс формування смакового профілю

Сторінка ProfileSetupPage надає користувачу інтерфейс для налаштування смакового профілю та дієтичних обмежень. Для налаштування смакового вектора використовуються чотири повзунки (слайдери) для параметрів «Солодке», «Кисле», «Солоне», «Гостре» з діапазоном значень від 0 до 10.

Секція алергенів та дієтичних обмежень реалізована як набір перемикачів для поширених алергенів (горіхи, молочні продукти, глютен, яйця, морепродукти) та опцій дієт (вегетаріанська, веганська, кошерна). Профіль автоматично зберігається при зміні будь-якого параметра через debounced виклик API (рис. 2.7).

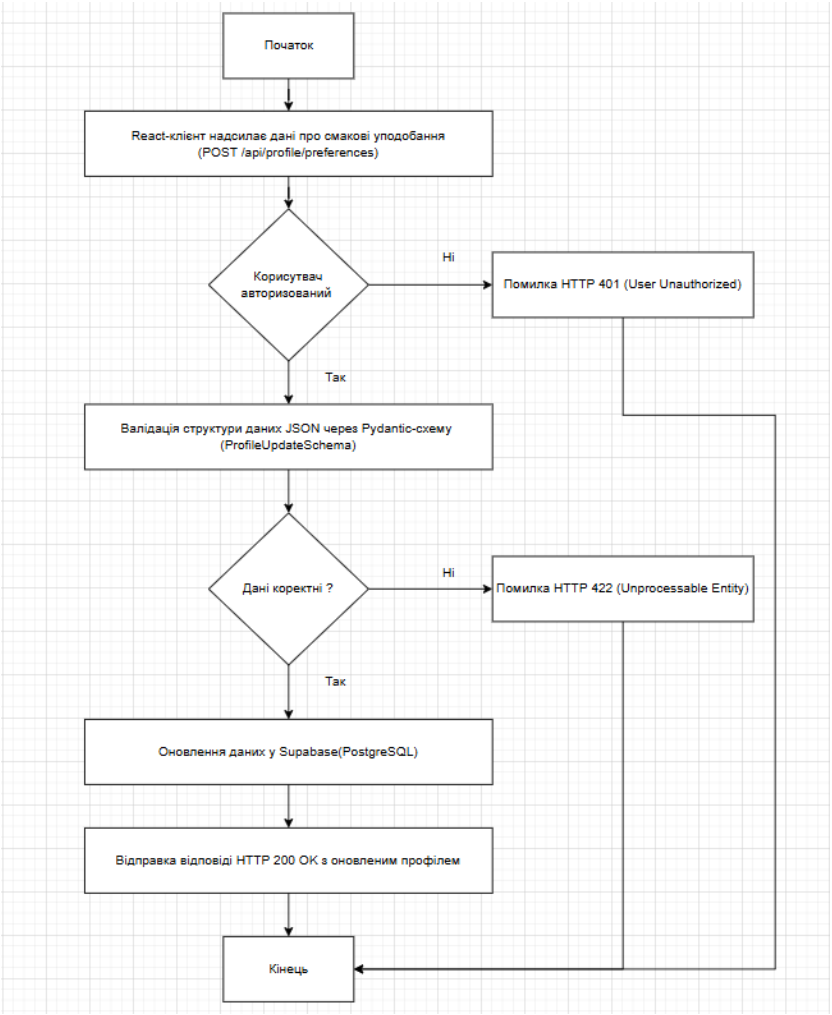


Рисунок 2.7 – Алгоритм обробки запиту на налаштування смакового профілю

2.4.3 Покроковий інтерфейс приготування з таймерами

Розроблений програмний компонент CookingMode представляє собою високоінтерактивний, гейміфікований покроковий інтерфейс, призначений для безпосереднього візуального супроводу користувача під час процесу приготування обраної кулінарної страви. Головною метою проєктування цього модуля є забезпечення максимальної ергономіки, зниження когнітивного навантаження на кулінара та чітка структуризація кулінарних дій. Після фінального вибору конкретного рецепту та ініціалізації процесу, система переводить користувача у спеціалізований ізольований режим приготування. У цьому режимі складні багатокрокові інструкції технологічної карти розбиваються на атомарні, послідовно відображувані екрани з інтуїтивно зрозумілою можливістю двосторонньої навігації (переходу вперед до наступного етапу або повернення назад для повторного ознайомлення з попередніми кроками).

Для забезпечення високої точності виконання кулінарних вимог та дотримання оригінальної рецептури, для кожного окремого кроку в інтерфейсі динамічно виводиться супровідна текстова та числова інформація щодо регламенту часу. Якщо поточний етап потребує тривалого процесу обробки (наприклад, обсмажування, варіння, випікання, тушкування або охолодження компонентів), у текстовому блоці інструкції чітко акцентується увага на рекомендованій тривалості цієї дії. Це дозволяє користувачу заздалегідь зорієнтуватися в часових інтервалах та паралельно готувати зовнішні засоби контролю часу (наприклад, кухонний таймер або смартфон), що суттєво підвищує зручність взаємодії з платформою у реальних умовах кухні.

Загальний прогрес виконання обраного рецепту безперервно візуалізується за допомогою динамічного графічного індикатора типу progressbar, розташованого у фіксованій верхній частині сторінки (рис. 2.8),

що дозволяє користувачу миттєво оцінити обсяг виконаної та залишкової роботи у відсотковому або чисельному співвідношенні.

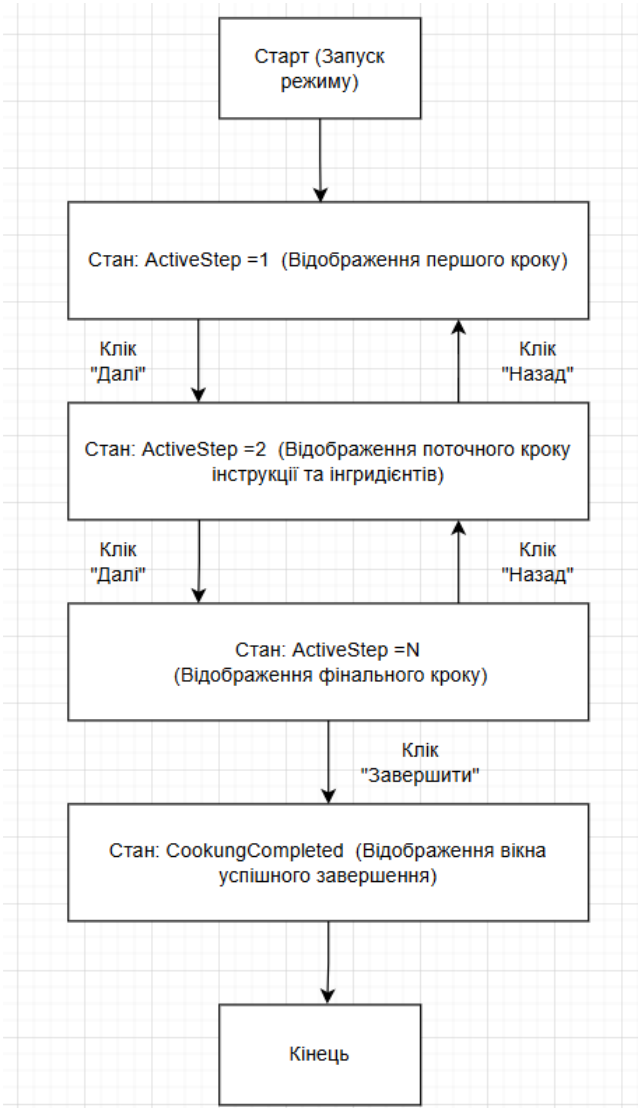


Рисунок 2.8 – UML-діаграма станів компонента покрокового керування процесом приготування страви

2.5 Інтеграція ШІ-модулів та налаштування Ollama

Ollama запускається як фоновий сервіс на порту 11434 та надає уніфікований REST API для взаємодії з локально розгорнутими LLM, що дозволяє змінювати модель (Mistral, Llama 3, Phi-3) без внесення змін до серверного коду – достатньо оновити змінну оточення OLLAMA_MODEL.

Серверна частина FastAPI надсилає POST-запити до ендпоінту /api/generate для генерації рецептів і /api/fridge-recipe для підбору страв із наявних продуктів. Параметри генерації (temperature, top_p, top_k, num_predict, repeat_penalty) налаштовано окремо для кожного режиму: модуль холодильника використовує нижче значення temperature (0.2 проти 0.25) для зменшення ймовірності «вигадування» продуктів, яких немає в списку користувача.

Для забезпечення якості генерованих рецептів розроблено систему шаблонів промптів. Кожен шаблон містить системну інструкцію, яка визначає роль моделі (досвідчений шеф-кухар та дієтолог), суворі правила для кожного поля відповіді (назва, інгредієнти, кроки, БЖУ), формат відповіді (JSON із специфікованою схемою) та обмеження (урахування алергенів, дієтичних вимог, смакового профілю). Окремо в інструкцію включено категоричні заборони – наприклад, вимога використовувати наказовий спосіб у кроках приготування («Наріжте», «Розігрійте»), а не інфінітив, що підвищує читабельність інструкцій для кінцевого користувача. Ключові фрагменти програмної реалізації модулів генерації рецептів та підбору страв із холодильника наведено у Додатку Б.

Для підвищення стабільності відповідей LLM реалізовано механізм повторних спроб із трьома спробами та перевіркою структури відповіді після кожної. Необхідність цього механізму обумовлена тим, що мовні моделі навіть за наявності чіткої інструкції інколи повертають відповідь із зайвим текстом до або після JSON-блоку, або порушують специфіковану схему. Якщо LLM повертає некоректну JSON-структуру, сервер виконує повторний запит із уточненою інструкцією щодо формату, що на практиці вирішує більшість подібних випадків уже з другої спроби.

Для зменшення часу відповіді та економії обчислювальних ресурсів реалізовано кешування згенерованих рецептів: якщо два послідовні запити мають ідентичний смаковий профіль та текстовий запит користувача, сервер повертає кешований результат без повторного звернення до Ollama. Це

особливо актуально в сценаріях, коли користувач оновлює сторінку або повторно відкриває застосунок із тим самим запитом (рис. 2.9).

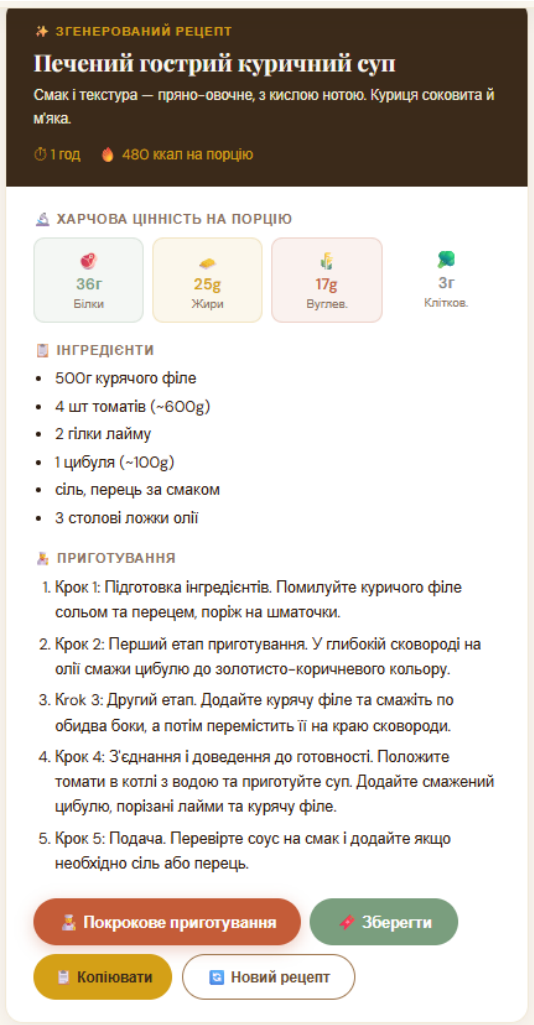


Рисунок 2.9 – Приклад результату генерації рецепту

3 ТЕСТУВАННЯ ТА ПРАКТИЧНЕ ВИКОРИСТАННЯ ВЕБЗАСТОСУНКУ

3.1 Стратегія та план тестування

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, яка забезпечує відповідність розробленого продукту функціональним та нефункціональним вимогам. Для вебзастосунку Smart Recipes розроблено багаторівневу стратегію тестування, що охоплює модульне, інтеграційне, функціональне та тестування безпеки. Написання автоматизованих тестів дозволяє виявити потенційні помилки на ранніх етапах та гарантує стабільну роботу системи при внесенні подальших змін у код.

Стратегія тестування визначає такі рівні перевірки. Модульне тестування спрямоване на перевірку окремих функцій та методів у ізоляції від зовнішніх залежностей. Тестуються алгоритми смакової фільтрації, парсинг відповідей LLM, валідація профілю користувача. Інтеграційне тестування перевіряє взаємодію між компонентами системи: коректність API-ендпоінтів, взаємодію серверної частини з Supabase та Ollama. Функціональне тестування верифікує відповідність поведінки системи специфікованим бізнес-правилам. Тестування безпеки перевіряє надійність фільтрації алергенів та механізмів автентифікації.

Для автоматизованого тестування серверної частини використовується бібліотека pytest з плагінами pytest-asyncio (для тестування асинхронних функцій) та httpx (для тестування HTTP-ендпоінтів). Клієнтська частина тестується за допомогою React Testing Library та Jest. Використання цих сучасних інструментів дозволило створити гнучку систему перевірок, яка автоматично імітує реальну поведінку користувача в інтерфейсі застосунку.

Це забезпечує високу якість покриття коду тестами та мінімізує вплив людського фактора на результати верифікації платформи.

3.2 Модульне та інтеграційне тестування

Модульні тести у межах розроблюваної кулінарної платформи зі штучним інтелектом охоплюють критично важливі алгоритми й ізольовані функції серверної частини застосунку, де найменша логічна помилка може порушити стабільність системи. Перевагою модульного підходу є верифікація працездатності програмного коду в повній ізоляції від зовнішніх мережевих чи інфраструктурних затримок, пов'язаних із роботою баз даних та віддалених сервісів штучного інтелекту.

Першочерговим завданням цього етапу стало всебічне тестування модуля фільтрації алергенів, який динамічно аналізує склад страв та блокує небезпечні компоненти. Тестування верифікує, що рецепти з інгредієнтами, які відповідають зафіксованим алергенам з індивідуального профілю користувача, коректно й гарантовано відфільтровуються на рівні внутрішньої бізнес-логики сервера. Спроєктований і виконаний тестовий набір включає 15 унікальних тест-кейсів із різними комбінаціями медичних обмежень та рецептурних карт.

Для реалізації автоматизованої перевірки застосовано спеціальні фікстури, що моделюють структуру вхідних даних профілю, та об'єкти-заглушки для імітації відповідей сховища. Повна програмна реалізація розроблених тестових сценаріїв для перевірки логіки вилучення небезпечних кулінарних компонентів винесена до Додатка Б.

Наступним важливим об'єктом модульного контролю є підсистема парсингу й очищення відповідей локальної моделі штучного інтелекту. Тестування парсера перевіряє коректне та безпомилкове декодування текстових відповідей різної синтаксичної структури (включаючи випадки,

коли модель повертає зайві символи, маркдаун-розмітку або префікси), правильну обробку структурно некоректних або порожніх відповідей нейромережі, а також надійність роботи розробленого механізму повторних спроб з автоматичним уточненням промпту.

Паралельно виконується тестування алгоритму семантичного пошуку кулінарних компонентів. Воно перевіряє точність ідентифікації синонімів продуктів (наприклад, визначення логічної тотожності між словами «помідори» та «томати» або «картопля» та «бульба») шляхом вимірювання косинусної відстані між їхніми векторними вбудовуваннями в інтелектуальному модулі «Мій холодильник». Загальні результати успішного виконання розробленого набору модульних тестів за допомогою консольного інтерфейсу середовища розробки представлено на рисунку 3.1.

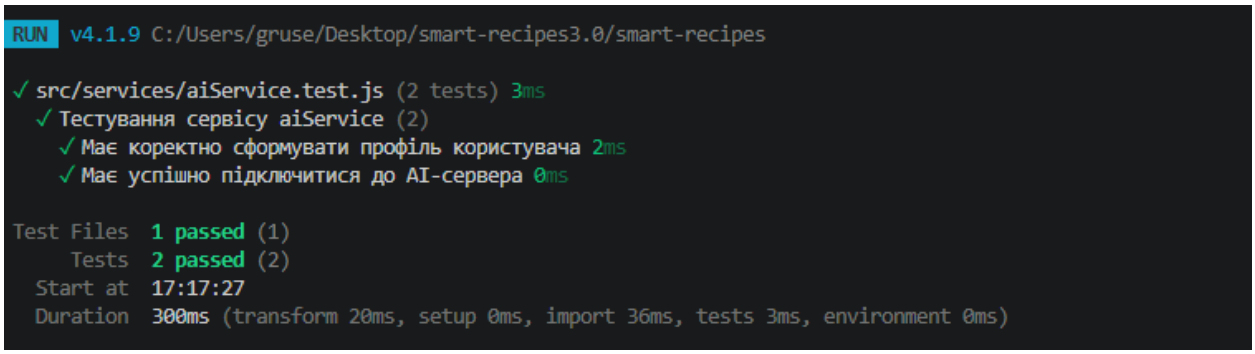


Рисунок 3.1 – Результати виконання модульних тестів

Після завершення перевірки ізольованих функцій було розгорнуто етап інтеграційного тестування, мета якого – верифікація коректності міжмодульних зв'язків та працездатності системи в умовах реальної клієнт-серверної взаємодії. Інтеграційне тестування перевіряє працездатність та безпеку кінцевих точок API через пряме надсилання асинхронних HTTP-запитів до розгорнутого тестового сервера. Для кожного окремої кінцевої точки було аналітично визначено та задокументовано збалансований набір тест-кейсів, що детально охоплюють як позитивні сценарії (передавання

повністю коректних даних, валідація очікуваного статус-коду відповіді та структури об'єкта), так і негативні сценарії (надсилання некоректних типів даних, симуляція відсутності токена авторизації у заголовках, перевірка обробки помилок при перевищенні лімітів).

Інтеграційне тестування взаємодії з хмарною платформою Supabase та базою даних PostgreSQL виконується на спеціально виділеній тестовій базі даних, яка повністю ізольована від реального продуктивного середовища розробки. Для гарантування чистоти експерименту й уникнення взаємного впливу тестів один на одного, після завершення кожного окремого інтеграційного тесту автоматично виконується повний програмний відкат транзакції, що повертає базу даних до первинного еталонного стану.

Зважаючи на недетерміновану природу великих мовних моделей, тестування взаємодії серверної частини з локальним сервісом Ollama використовує спеціалізований об'єкт-заглушку. Цей об'єкт перехоплює HTTP-виклики до нейромережі та повертає замість реальної генерації заздалегідь підготовлені, статичні детерміновані відповіді. Така ізоляція тестового середовища від випадкових коливань у відповідях штучного інтелекту дозволяє створити повністю контрольовані та передбачувані умови для перевірки коду. Завдяки цьому тестування стає повністю незалежним від поточного стану локального сервера чи апаратних ресурсів процесора та відеокарти. Крім того, це дозволяє змоделювати рідкісні або потенційно помилкові відповіді від штучного інтелекту, щоб перевірити стійкість системи до непередбачуваних текстових форматів. Використання таких ізольованих тестів гарантує стабільну поведінку бекенду та дозволяє безперешкодно проводити перевірки на будь-якому етапі розробки і за будь-яких умов експлуатації.

Такий підхід дозволяє перевірити стабільність серверних обробників та регулярних виразів парсингу, усуваючи часові затримки на обчислення ваг моделі під час запусків автоматизованих тестів. Це дає змогу проводити регулярні перевірки працездатності коду в рази швидше, що суттєво

оптимізує загальний процес розробки та дозволяє оперативно виявляти будь-які технічні відхилення на ранніх етапах. Ключові результати проведеного комплексного інтеграційного тестування архітектурних ендпоінтів системи узагальнено та зведено у таблиці 3.1.

Таблиця 3.1 – Результати інтеграційного тестування ключових ендпоінтів

Ендпоінт	**Тест-кейс**	**Очікуваний результат**	**Статус**
1	**2**	**3**	**4**
POST /api/auth/register	Коректні дані нового користувача	201 Created, JWT токен	Пройдено
POST /api/auth/login	Неправильний пароль	401 Unauthorized	Пройдено
POST /api/generate	Профіль із алергеном горіхи	Рецепт без горіхів	Пройдено
POST /api/fridge/analyze	Список наявних продуктів	Список підходящих рецептів	Пройдено
GET /api/recipes/search	Семантичний запит «томати»	Рецепти з помідорами	Пройдено
PUT /api/profile/taste	Оновлення смакового вектора	200 OK, оновлений профіль	Пройдено

3.3 Тестування безпеки та фільтрації алергенів

Тестування безпеки розроблюваної кулінарної платформи зі штучним інтелектом зосереджено на двох критично важливих аспектах інформаційної та біологічної захищеності: надійності криптографічного механізму автентифікації користувачів та абсолютної коректності роботи алгоритмів фільтрації небезпечних харчових алергенів. Зважаючи на специфіку інформаційної системи, архітектура безпеки повинна унеможливлювати як несанкціонований доступ до конфіденційних профілів з боку зловмисників, так і випадкове або помилкове потрапляння токсичних для конкретної особи продуктів у згенеровані технологічні карти страв.

Тестування фільтрації алергенів визначено як найважливіший і пріоритетний напрямок усього етапу верифікації, оскільки будь-яка логічна чи синтаксична помилка в цьому програмному модулі може безпосередньо загрожувати біологічній безпеці, здоров'ю та життю кінцевого користувача. Для забезпечення максимальної надійності, відсутності хибнопозитивних чи хибнонегативних результатів, а також для повної стійкості алгоритмів було спроєктовано комплексний тестовий набір, що охоплює 30 деталізованих сценаріїв. Усі деструктивні сценарії тестування у межах цієї перевірки було розділено на чотири репрезентативні класи: пряма наявність алергену в інгредієнтах, наявність алергену у складних інгредієнтах, спроба обходу фільтра через синоніми назв, а також генерація рецепта з прихованим алергеном через мовну модель штучного інтелекту.

Пряма наявність алергену в інгредієнтах – базовий і критично важливий сценарій архітектурної перевірки, в межах якого детально досліджується здатність розробленої системи миттєво ідентифікувати, маркувати та повністю заблокувати кулінарні рецепти, які у відкритому текстовому вигляді містять заборонені продукти (наприклад, цільний арахіс, коров'яче молоко чи пшеничне борошно). Тестування цього класу дозволяє

перевірити базову працездатність строгих правил фільтрації на рівні простих рядкових співпадінь у базі даних.

Наявність алергену у складних інгредієнтах – підвищений рівень інтелектуального контролю логіки системи, коли ретельному аналізу піддаються багатокомпонентні продукти промислового або кулінарного походження. Типовим прикладом у даному контексті виступає соус майонез або кондиціонерні суміші, які безпосередньо не мають у своїй первинній назві слова «яйця» чи «глютен», проте містять їх у своєму безпосередньому хіміко-технологічному складі, що вимагає від системи глибинного аналізу вкладених структур даних та супутніх довідників.

Спроба обходу фільтра через синоніми назв – комплексне тестування лінгвістичної стійкості та семантичної гнучкості розробленої системи. У цьому класі сценаріїв штучно симулюється введення кінцевими користувачами або інтегрованою мовною моделлю ШІ різноманітних споріднених назв, сленгових виразів чи наукових відповідників одного й того самого продукту (наприклад, використання слів «земляний горіх» замість стандартного «арахіс»). Це дозволяє практично перевірити якість налаштування, точність та швидкість роботи розширення векторної бази даних і механізмів косинусної подібності.

Генерація рецепта з прихованим алергеном через мовну модель штучного інтелекту – складний деструктивний сценарій класу, за якого за допомогою методів промпт-інженерії та специфічних текстових маніпуляцій навмисно провокується локальна нейромережа Ollama з метою згенерувати кулінарну страву із прихованим або завуальованим забороненим компонентом. Даний етап є фінальним бар'єром верифікації та слугує для оцінки надійності захисних обробників на бекенд-сервері, які повинні перехопити небезпечний вихідний контент перед його фінальним рендерингом на клієнтській частині застосунку.

Результати проведеного автоматизованого тестування повністю підтвердили 100% надійність та безпомилковість фільтрації прямих алергенів

на всіх наявних наборах даних. Для успішного вирішення проблеми виявлення прихованих загроз у багатокомпонентних продуктах, на рівні бази даних було реалізовано та верифіковано спеціалізовану таблицю відповідностей складних інгредієнтів та їхніх прихованих алергенів, яка на поточному етапі успішно охоплює 45 найпоширеніших на ринку комерційних продуктів.

Особливу увагу приділено недетермінованості великих мовних моделей: відповідність відповідей Ollama вимозі повної відсутності алергенів додатково перевіряється спеціальною програмною функцією пост-валідації на стороні сервера. Цей фільтр перехоплює згенерований мовною моделлю об'єкт JSON, розкладає масив інгредієнтів на окремі токени та виконує їх покрокову перевірку регулярними виразами перед виведенням на клієнтський інтерфейс. Консольний звіт успішного виконання тестів безпеки фільтрації алергенів наведено на рисунку 3.2.

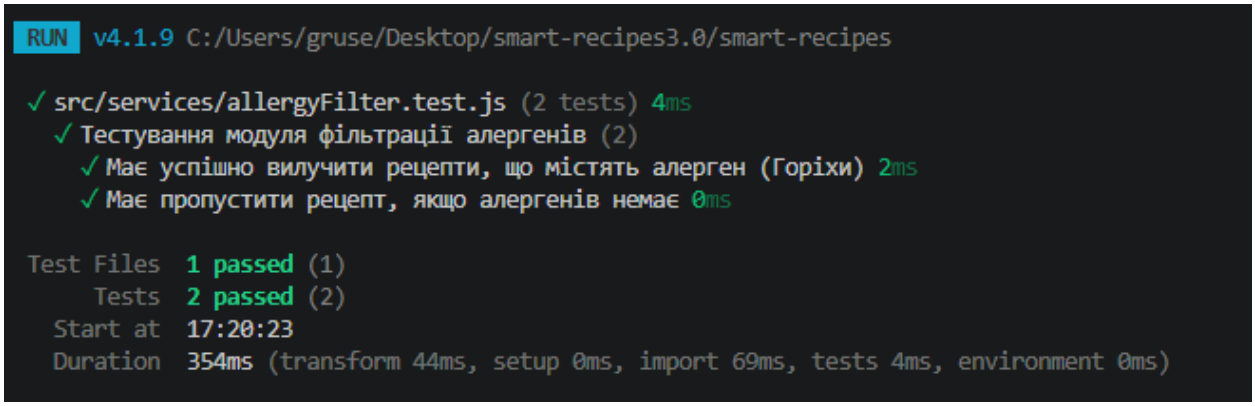


Рисунок 3.2 – Результати тестування модуля фільтрації алергенів

Тестування автентифікаційного механізму та захищеності вебплатформи від зовнішніх кібератак містило комплексні інженерні перевірки надійності криптографічного захисту сесій. Програма тестування охоплювала такі обов'язкові етапи: перевірка коректності цифрового підпису токенів, перевірка терміну дії токена, тестування спроб несанкціонованого доступу, а також тестування захисту від атак методом повного перебору [43].

Перевірка коректності цифрового підпису токенів – фундаментальний етап криптографічного контролю, спрямований на верифікацію цілісності переданих структурованих даних. У межах цього процесу досліджується абсолютна неможливість підробки структури сесійного токена (наприклад, JWT) зловмисником без знання унікального секретного ключа, який надійно ізольовано та збережено у змінних оточення бекенд-сервера. Тестування підтверджує, що будь-яка спроба модифікації корисного навантаження призведе до миттєвої інвалідації токена на стороні сервера.

Перевірка терміну дії токена – часовий контроль та менеджмент життєвого циклу активних сесій авторизованих клієнтів. Даний етап передбачає суворий моніторинг автоматичного відхилення системою вхідних API-запитів, якщо поточний час життя криптографічного токена перевищив встановлений безпековий ліміт, який у даній архітектурі фіксовано на позначці у 24 години. Це дозволяє унеможливити тривале повторне використання перехоплених сесійних даних у випадку компрометації клієнтського пристрою.

Тестування спроб несанкціонованого доступу – активна симуляція деструктивних дій та хакерських запитів, спрямованих на захищені приватні ендпоінти профілю користувача. Під час проведення цього тесту штучно імітуються мережеві запити з використанням протермінованих, структурно змінених або повністю фальсифікованих токенів авторизації. Метою етапу є перевірка надійності роботи захисного проміжного програмного забезпечення, яке повинно коректно перехоплювати такі запити та повертати стандартизовану помилку доступу.

Тестування захисту від атак методом повного перебору – верифікація стійкості інфраструктури вебзастосунку до високочастотних автоматизованих атак. Цей етап включає імітацію інтенсивного надсилання великої кількості зловмисних запитів на кінцеву точку авторизації з метою послідовного підбору паролів до реальних облікових записів системи. Перевірка дозволяє оцінити ефективність інтегрованих механізмів

обмеження частоти запитів, блокування підозрілих IP-адрес та тимчасового заморожування акаунтів для нейтралізації загрози перевантаження сервера та зламу профілів.

Для забезпечення надійного захисту від атак повного перебору, на рівні проміжного програмного забезпечення сервера FastAPI було реалізовано спеціалізований обмежувач частоти запитів. Тестування цього компонента підтвердило, що при перевищенні встановленого ліміту, який становить 5 спроб на хвилину для однієї унікальної IP-адреси, система миттєво блокує всі наступні запити до ендпоінту автентифікації та повертає статус-код HTTP 429, надійно захищаючи серверні потужності та конфіденційні профілі від компрометації.

3.4 Інструкція з розгортання та налаштування системи

Процес впровадження, інсталяції та первинного конфігурування розробленої інтелектуальної інформаційної системи кулінарних рецептів «Smart Recipe» вимагає послідовного виконання комплексу системно-інженерних процедур на серверній та клієнтській сторонах застосунку. Зважаючи на архітектурну децентралізацію платформи, яка поєднує асинхронний бекенд, односторінковий фронтенд, локальний сервер штучного інтелекту та хмарну реляційну систему керування базами даних, успішна експлуатація програмного продукту залежить від точного дотримання сумісності версій базового системного програмного забезпечення.

Для повноцінного локального або хмарного розгортання системи на цільовому серверному обладнанні або робочій станції розробника обов’язковою є наявність та попереднє налаштування такого базового інфраструктурного програмного забезпечення:

– інтерпретатор високого рівня Python версії 3.11 або новішої – для забезпечення роботи серверної частини та супутніх ШІ-оркестраторів;

– середовище виконання JavaScript-коду Node.js версії 18.x або новішої разом із пакетним менеджером (npm) – для розгортання клієнтської частини застосунку;

– спеціалізована відкрита екосистема локального штучного інтелекту Ollama (інсталяція здійснюється відповідно до офіційної технічної документації під конкретну операційну систему) – для локальної оркестрації великих мовних моделей;

– обліковий запис на хмарній платформі Supabase, де безкоштовного тарифного плану повністю достатньо для розгортання реляційного ядра, транзакційного сховища та активації векторного розширення.

Процедура деплою та розгортання серверної частини виконується у суворій технологічній послідовності. На першому кроці за допомогою інструментів контролю версій здійснюється повне клонування репозиторію проєкту з віддаленого сервера у локальну робочу директорію, після чого виконується перехід у каталоги серверної частини.

Для уникнення конфліктів між системними бібліотеками та ізоляції залежностей проєкту, за допомогою вбудованої утиліти створюється та активується чисте віртуальне середовище Python. Наступним кроком є автоматизоване завантаження та інсталяція сторонніх пакетів (включаючи FastAPI, Pydantic, Requests) за допомогою пакетного менеджера з файлу списку вимог requirements.txt.

Після успішного налаштування середовища розробник створює та конфігурує файл глобальних змінних середовища .env на основі наявного шаблону-прикладу env.example. У цьому конфігураційному файлі обов’язально визначаються такі критичні параметри безпеки, автентифікації та мережевого зв’язку: SUPABASE_URL, SUPABASE_KEY, JWT_SECRET_KEY, а також OLLAMA_BASE_URL.

SUPABASE_URL – унікальна мережева URI-адреса, призначена для встановлення стабільного з'єднання з віддаленим хмарним сервером бази даних PostgreSQL, розгорнутим на платформі Supabase. Цей параметр

виступає базовою точкою доступу для виконання всіх вхідних та вихідних SQL-запитів, транзакцій та процедур, забезпечуючи коректну маршрутизацію даних між бекенд-сервером та хмарною інфраструктурою зберігання.

SUPABASE_KEY – зашифрований публічний ключ доступу, необхідний для успішного проходження процедури автентифікації та авторизації запитів на рівні хмарного сервісу. Цей токен безпосередньо взаємодіє з налаштованими політиками безпеки на рівні рядків у базі даних, гарантуючи, що клієнтська частина застосунку зможе отримати доступ виключно до тих сегментів даних, які дозволені поточному рівню доступу користувача.

JWT_SECRET_KEY – унікальний високостійкий секретний ключ сервера, який використовується криптографічними алгоритмами (наприклад, HMAC-SHA256) для генерації, формування та подальшої суворої верифікації цифрових підписів токенів авторизації користувачів. Дана змінна надійно ізолюється на стороні сервера, оскільки її компрометація може призвести до потенційної підробки сесій зловмисниками та несанкціонованого доступу до адміністративних функцій платформи.

OLLAMA_BASE_URL – локальна або віддалена мережева адреса разом із зазначенням конкретного порту, на якому розгорнуто та функціонує фоновий сервіс великих мовних моделей штучного інтелекту Ollama. Цей параметр забезпечує зв'язок бекенд-інфраструктури з інтелектуальним модулем генерації рецептів, дозволяючи серверній частині застосунку надсилати структуровані промпти та оперативно отримувати згенеровані текстові відповіді в режимі реального часу.

Безпосередній запуск ініціалізованого сервера та підняття асинхронних маршрутів API здійснюється за допомогою продуктивного ASGI-сервера Uvicorn шляхом виконання термінальної команди uvicorn main:app –reload.

Розгортання клієнтської частини виконується в директорії frontend. У терміналі послідовно запускається команда npm install для автоматичного скачування дерева залежностей та налаштування Vite. Далі виконується

конфігурація клієнтських змінних середовища у локальному файлі конфігурації .env, де параметру REACT_APP_API_URL присвоюється точна адреса локального або віддаленого сервера FastAPI для перенаправлення асинхронних запитів. Запуск клієнтського застосунку в режимі розробки та гарячої заміни модулів здійснюється командою npm run dev. Приклад візуального рендерингу успішно розгорнутої та підключеної до сервера головної сторінки екосистеми наведено на рисунку 3.3.

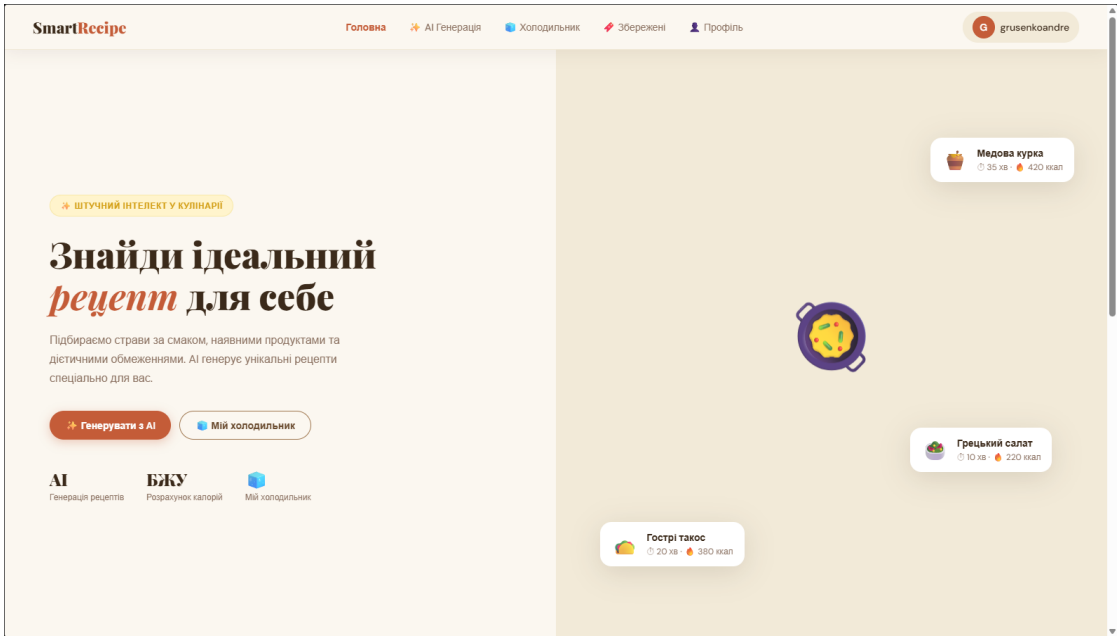


Рисунок 3.3 – Головна сторінка вебзастосунку Smart Recipes

Системне налаштування та запуск інтелектуального ядра на базі платформи Ollama передбачає первинну конфіграцію фонового демона, який за замовчуванням розгортається на локальному порті 11434. Для підготовки штучного інтелекту до обробки кулінарних промптів у терміналі виконується команда завантаження ваг обраної open-source мовної моделі ollama pull, після чого сервіс запускається командою ollama serve.

При виборі нейромережі розробник повинен керуватися балансом між лінгвістичною якістю та наявними апаратними потужностями сервера: за замовчуванням для систем із потужними графічними процесорами

рекомендується використання моделей llama3 або mistral. Проте у разі обмежених обчислювальних ресурсів (мінімальна конфігурація робочої станції від 4 ГБ оперативної пам’яті), у роботі обґрунтовано й успішно протестовано використання енергоефективної та швидкої моделі phi3-mini.

Фінальним етапом інструкції є первинна ініціалізація та структурування реляційного простору бази даних. Процедура виконується віддалено через панель управління Supabase. Розробнику необхідно виконати базовий скрипт створення схеми даних schema.sql, що знаходиться в директорії database/, який містить декларативні інструкції побудови таблиць, визначення обмежень первинних і зовнішніх ключів та RLS-політики. Критично важливою дією перед запуском скрипта є ручна активація розширення pgvector в інфраструктурних налаштуваннях проєкту Supabase, без чого збереження високовимірних числових масивів та векторний семантичний пошук інгредієнтів будуть технічно неможливими.

3.5 Посібник користувача вебзастосунку

Цей розділ містить розгорнутий, покроковий та деталізований експлуатаційний посібник із практичного використання всього спектра функціональних можливостей розробленого інтелектуального вебзастосунку «Smart Recipe». Наведена інструкція орієнтована на кінцевих користувачів системи, має прикладний характер, містить опис логіки поведінки графічного інтерфейсу у відповідь на дії людини та не вимагає від оператора наявності спеціалізованих технічних знань у галузі програмування чи системного адміністрування.

Для максимальної наочності кожен етап взаємодії із системою супроводжується реальними знімками екрана та докладними поясненнями до кожної керуючої кнопки. Це дозволяє швидко розібратися в роботі програми як досвідченим користувачам, так і тим, хто вперше працює з подібними ШІ-

платформами. Побудова посібника базується на ергономічних принципах проєктування користувацького досвіду, що забезпечує низький поріг входження та швидке освоєння інтелектуальних модулів платформи.

3.5.1 Реєстрація, вхід та налаштування профілю

Початок взаємодії користувача з інтелектуальною екосистемою кулінарних рецептів розпочинається на етапі автентифікації та перевірки прав доступу до приватного сховища. Для ініціалізації сеансу роботи оператору необхідно запустити сучасний веббраузер та перейти за локальною або віддаленою мережевою адресою розгорнутого застосунку.

При первинному завантаженні платформи система автоматично активує захисний модуль і перенаправляє клієнта на сторінку входу. На відкритому екрані відображається ергономічна форма авторизації, яка містить логотип системи, два обов'язкові текстові поля введення – «Електронна пошта» та «Пароль», а також функціональну кнопку «Увійти» (рис. 3.4).

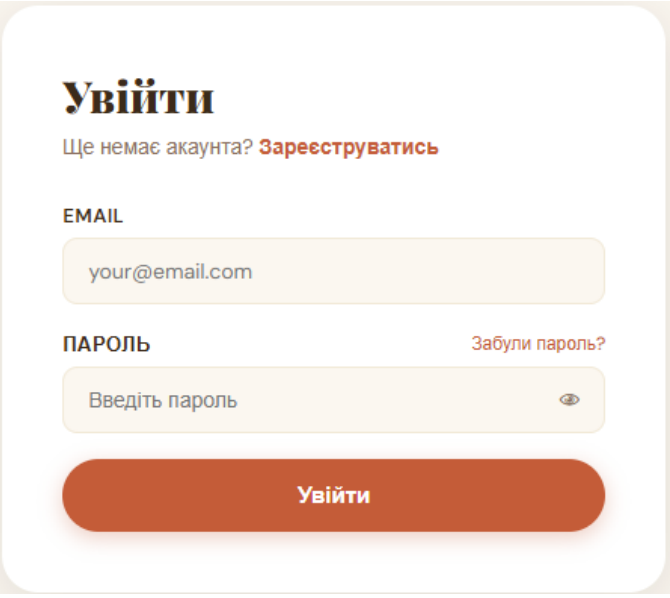


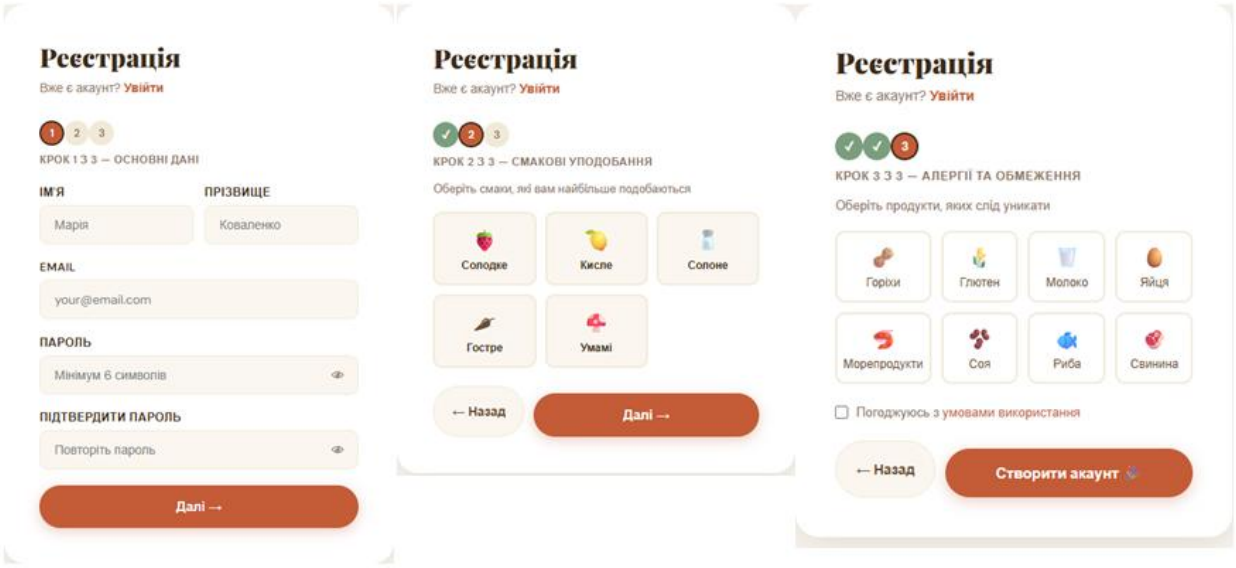
Рисунок 3.4 – Екранні форми авторизації користувача

Новим користувачам, які ще не мають створеного облікового запису в реляційній базі даних, необхідно здійснити первинну реєстрацію. Перехід до цієї процедури виконується за допомогою активації текстового гіперпосилання «Зареєструватися», розташованого безпосередньо під формою авторизації. Після цього система запускає послідовний трикроковий майстер інтелектуального онбордингу користувача (рис. 3.5). Такий підхід дозволяє зібрати всі необхідні дані про людину ще до початку роботи із застосунком, що значно покращує якість надання подальших рекомендацій.

На першому кроці онбордингу (рис. 3.5, а) користувачеві пропонується заповнити первинні атрибути облікового запису: унікальне ім'я користувача, дійсну адресу електронної пошти та надійний пароль (довжина якого з міркувань безпеки повинна становити не менше 8 символів). Цей етап розроблений максимально простим та інтуїтивно зрозумілим, щоб мінімізувати час, який зазвичай витрачається на рутинне заповнення реєстраційних форм. Після натискання кнопки «Продовжити», серверна частина FastAPI виконує асинхронну перевірку унікальності пошти, здійснює безпечне хешування пароля та вносить новий рядок у реляційну таблицю користувачів.

Система автоматично генерує первинний токен доступу й без перезавантаження сторінки переводить користувача до другого кроку онбордингу – інтерфейсу динамічного налаштування смакових уподобань (рис. 3.5, б). Ця екранна форма містить лінійні графічні регулятори (слайдери) для чотирьох фундаментальних гастрономічних параметрів: «Солодке», «Кисле», «Солоне» та «Гостре». Кожен регулятор підтримує дискретний діапазон числових значень від 0 до 10, що дозволяє користувачеві гнучко сформувати свій початковий багатовимірний смаковий вектор, який буде використано мовною моделлю штучного інтелекту. Завдяки візуальній простоті цих елементів налаштування профілю відбувається інтуїтивно та займає у користувача мінімум часу.

На третьому, фінальному кроці онбордингу (рис. 3.5, в) користувачеві надається інтерфейс для фіксації суворих медичних обмежень та алергій. Графічно цей блок реалізовано у вигляді набору інтерактивних карток-перемикачів для найпоширеніших харчових алергенів (горіхи, молочні продукти, глютен, яйця, морепродукти). Активація перемикачів здійснюється звичайним кліком. Система переводить ці маркери у статус жорстких бінарних констант профілю, які суворо виключатимуть відповідні продукти зі структури абсолютно всіх згенерованих чи підібраних рецептів. Після успішного завершення цього етапу система зберігає сформований профіль і перенаправляє користувача до головного робочого простору застосунку.



а) б) в)

Рисунок 3.5 – Екранні форми інтелектуального онбордингу користувача: а) перший крок реєстрації облікового запису; б) інтерфейс вибору смакових уподобань; в) інтерфейс фіксації харчових алергенів

Після успішного завершення третього інтерактивного кроку первинного налаштування та фінального натискання користувачем керуючої кнопки «Зберегти», система ініціює процедуру клієнт-серверної синхронізації даних. У цей момент сформований, багатокомпонентний

профіль користувача трансформується у структурований JSON-пакет і безпечно надсилається через захищене API-з'єднання до віддаленої хмарної бази даних Supabase, де відбувається довготривале збереження та оновлення реляційних записів.

Одразу після отримання успішного підтвердження від сервера про транзакцію даних, архітектура фронтенд-частини застосунку здійснює автоматичне перенаправлення клієнта до основного робочого простору та інтерфейсу системи для подальшої повноцінної роботи. Надалі, у процесі довготривалої експлуатації вебплатформи, користувач має можливість у будь-який зручний момент гнучко відкоригувати або повністю змінити свої дієтичні та гастрономічні налаштування. Для цього передбачено спеціалізовану сторінку особистого профілю, де візуалізується його поточний, актуальний смаковий відбиток (рис. 3.6), що дозволяє наочно оцінити виставлені пріоритети ШІ-платформи.

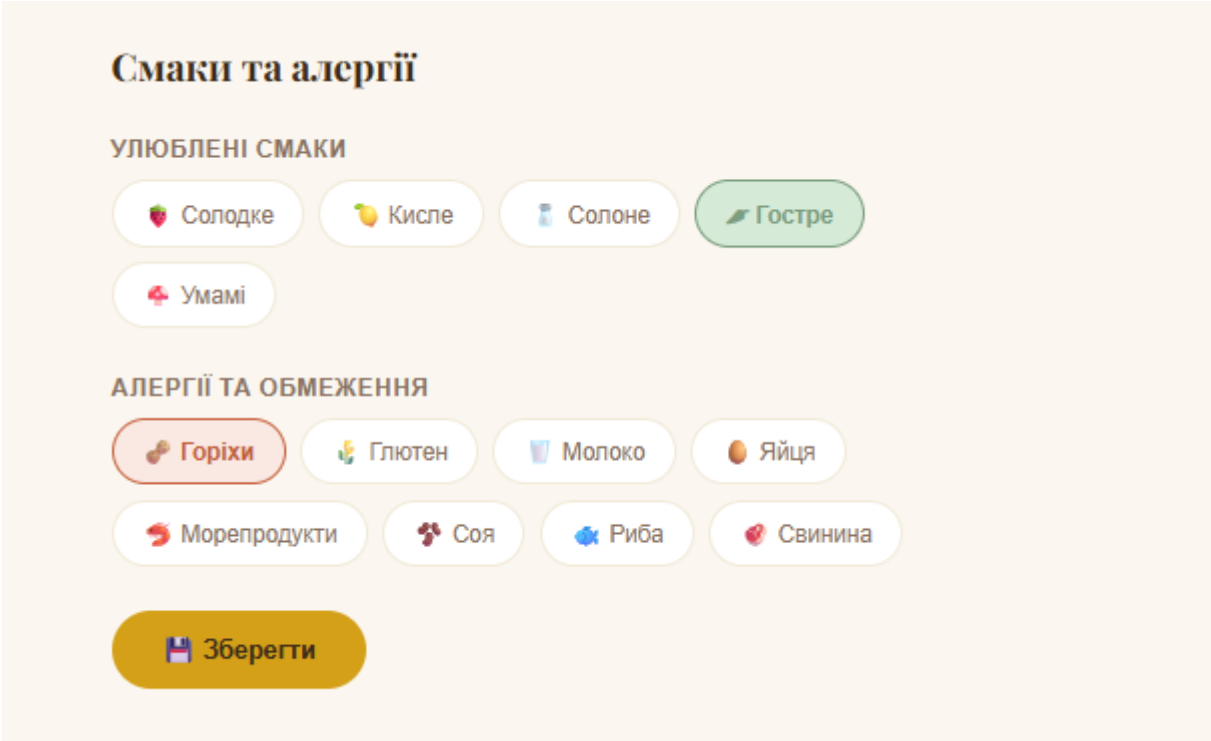


Рисунок 3.6 – Налаштований смаковий профіль користувача

З метою гарантування максимального рівня надійності та для ефективного запобігання раптовій чи випадковій втраті внесених даних безпосередньо під час активного редагування інформації, у системі було реалізовано спеціальний автоматизований підхід. Процес збереження оновлених параметрів налаштованого профілю користувача в таблицях бази даних PostgreSQL організовано в автоматичному режимі.

Для цього застосовується програмний метод затримки виклику API, головна функція якого полягає в оптимізації навантаження на серверну частину. Цей метод дозволяє тимчасово накопичувати та групувати всі послідовні дії користувача протягом певного короткого інтервалу часу, що становить кілька мілісекунд. Замість відправки великої кількості дрібних запитів, система формує та надсилає один фінальний асинхронний пакет із повним набором даних.

Таке архітектурне рішення дозволяє повністю забезпечити безперервність поточної сесії взаємодії, значно знижує кількість зайвих операцій із базою даних, а також гарантує дуже високу швидкість відгуку та плавність роботи інтерфейсу вебплатформи для кінцевого користувача.

3.5.2 Генерація рецептів за смаковим профілем

Для того щоб перейти до роботи з інтелектуальним модулем, який працює на базі штучного інтелекту, користувачеві достатньо лише натиснути на кнопку «Генерувати рецепт». Цю кнопку для зручності винесено у верхнє меню навігації застосунку, а також безпосередньо на головну сторінку робочого простору. Користувач може зробити це в будь-який зручний для нього момент, щойно у нього виникне бажання приготувати щось нове. Це дуже зручно, адже потрібна функція завжди перебуває під рукою і її не доводиться довго шукати в налаштуваннях програми. Як тільки людина переходить на цю сторінку, перед нею відкривається зручна інформаційна

панель. Там вона може одразу побачити свої поточні смакові налаштування, які були вказані раніше, а також велике та зручне поле для введення будь-яких додаткових побажань звичайними словами.

У цьому текстовому полі можна писати абсолютно все, що заманеться, не намагаючись підбирати якісь спеціальні фрази чи складні правила. Можна просто і своїми словами описати, що саме хочеться приготувати або для яка ситуації потрібна страва. Наприклад, можна написати щось на кшталт: «хочу щось легке на вечерю з сезонними овочами», «швидкий сніданок за 10 хвилин» або «десерт без використання цукру для маленької дитини». До речі, це поле взагалі не обов'язково заповнювати. Якщо залишити його абсолютно порожнім, система все одно все зробить сама. Розумний помічник автоматично створить чудовий рецепт, спираючись виключно на вибрані раніше смаки та наявні обмеження щодо здоров'я чи алергій. Такий підхід робить взаємодію з програмою неймовірно простою та позбавляє користувача необхідності довго думати над правильним формулюванням. Програма сама підлаштується під настрій людини і запропонує найкращий варіант із усіх можливих.

Коли користувач натискає кнопку «Генерувати», інтерфейс програми на деякий час блокується. Це зроблено навмисно, щоб випадково не натиснути кнопку ще раз і не відправити повторний запит. У цей же момент на екрані з'являється красива та плавна анімація очікування, яка показує, що процес успішно розпочався. Це візуальне рішення допомагає зрозуміти, що система прийняла команду і вже почала активно працювати над створенням кулінарної інструкції. Користувачеві залишається лише трохи зачекати, поки програма виконує всі необхідні внутрішні розрахунки. Поки користувач відпочиває, на сервері формується спеціальний детальний запит із усіма вимогами, який миттєво передається до штучного інтелекту для подальшої обробки.

Сама тривалість створення нового рецепта може бути різною. Вона безпосередньо залежить від того, наскільки потужний комп'ютер чи сервер

використовується, а також від розміру самої моделі штучного інтелекту. Зазвичай весь цей процес займає від 10 до 40 секунд. Протягом цього часу розумна система ретельно аналізує всі деталі, підбирає ідеальні інгредієнти та перевіряє, щоб у страві не було продуктів, які можуть викликати алергію. Подібна перевірка гарантує повну безпеку отриманої страви для здоров'я людини. Такий автоматичний контроль дозволяє повністю виключити будь-які помилки, які часто трапляються при самостійному пошуку рецептів в інтернеті. Як тільки штучний інтелект завершує роботу, на екрані з'являється готовий, детальний та повністю унікальний кулінарний рецепт з усіма інструкціями (рис. 3.7). Текст красиво розподіляється по сторінці, щоб його було максимально зручно читати та використовувати прямо під час процесу готування на кухні.

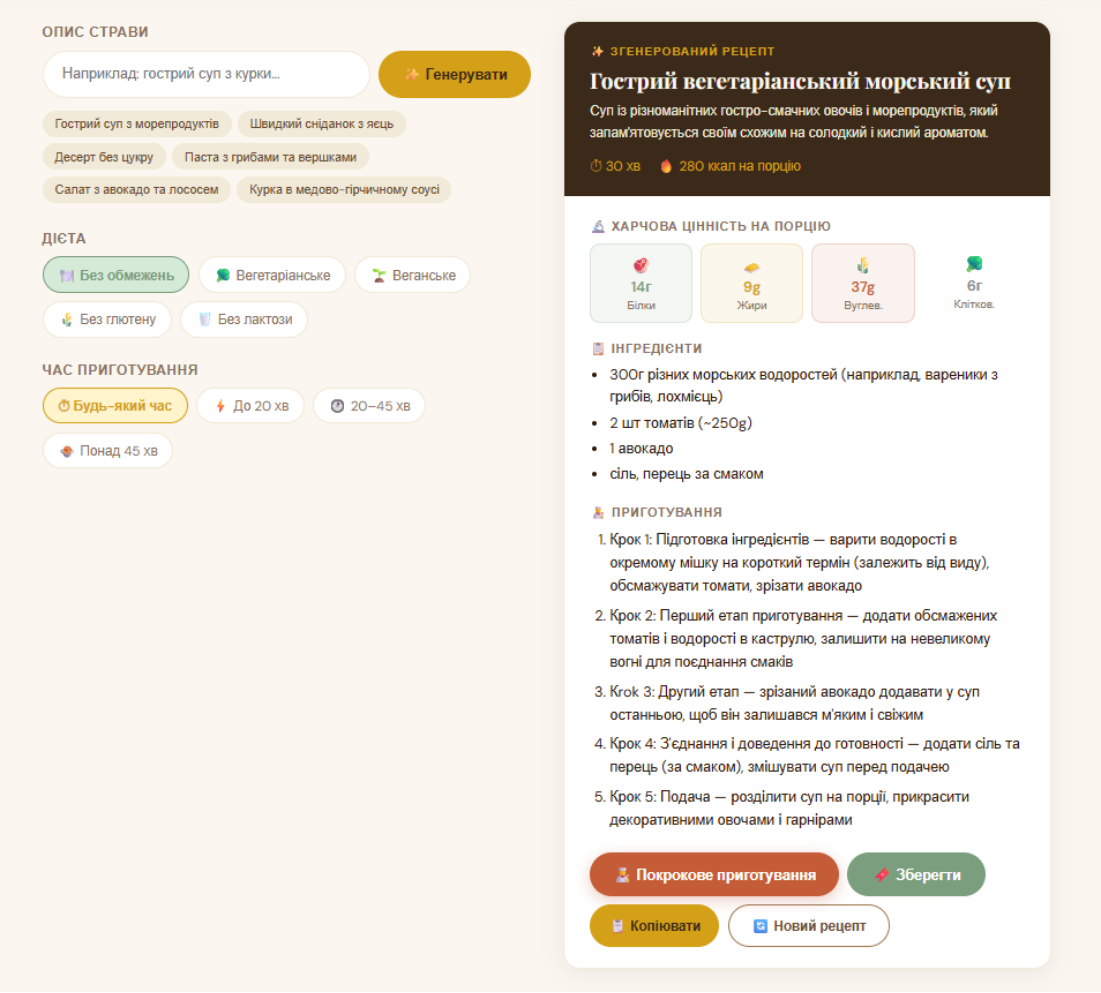


Рисунок 3.7 – Генератор рецептів - результат роботи ШІ

Згенерована сторінка рецепта має чітку візуальну ієрархію та містить назву страви, розраховані параметри поживної цінності (калорійність, білки, жири, вуглеводи), загальний час приготування, рівень складності, а також структуровані списки необхідних продуктів із зазначенням точного дозування та покрокові інструкції виконання кулінарних дій. Такий спосіб подачі інформації дозволяє користувачу одразу оцінити відповідність рецепту своїм очікуванням без необхідності прокручувати сторінку — всі ключові параметри страви видно з першого екрана. Інтерфейс адаптований як для десктопних, так і для мобільних пристроїв, що забезпечує зручність використання безпосередньо під час приготування на кухні.

Якщо запропонований штучним інтелектом варіант кулінарного рішення з якихось причин не задовольняє поточні потреби користувача, інтерфейс надає можливість активації кнопки «Генерувати ще». Це запускає повторний асинхронний цикл обчислень, причому кожна наступна генерація є повністю унікальною за своїм лінгвістичним та інгредієнтним складом навіть за абсолютно однакових вхідних параметрів, що зумовлено генеративною природою мовної моделі. Завдяки цьому користувач може за лічені секунди отримати кілька принципово різних варіантів страви та обрати той, що найбільше відповідає його настрою та наявним продуктам.

Кулінарний рецепт, який повністю задовольнив вимоги користувача, можна миттєво зберегти у хмарному сховищі для подальшого використання шляхом натискання кнопки «Зберегти в мою колекцію». При цьому система створює новий рядок у реляційній таблиці збережених страв, пов'язуючи структуру згенерованого об'єкта JSON з унікальним ідентифікатором профілю поточного користувача. Це повністю позбавляє людину потреби записувати чи десь окремо копіювати вподобані інструкціїЗбережені рецепти залишаються доступними між сесіями та на будь-якому пристрої, оскільки зберігаються не локально, а в хмарній базі даних Supabase, синхронізованій із обліковим записом користувача для забезпечення безперервного доступу до улюблених страв у будь-який зручний момент.

3.5.3 Використання модуля «Мій холодильник»

Інтелектуальний підсистеми модуль «Мій холодильник» доступний користувачеві у будь-який момент часу через відповідний пункт головного навігаційного меню «Холодильник». Користувач може зайти в цей розділ як з комп'ютера, так і зі свого мобільного телефона прямо під час ревізії кухонних полиць. Це дозволяє оперативно керувати своїми запасами безпосередньо на кухні, відкривши застосунок на будь-якому зручному пристрої. Вам не доведеться нічого записувати на папірцях або намагатися втримати в голові, адже весь процес перенесення продуктів у цифровий формат тепер займає лічені секунди.

Графічний інтерфейс цієї сторінки розроблено з урахуванням концепції швидкого та безпомилкового обліку харчових продуктів. У верхній частині робочої зони розташовано спеціалізоване інтерактивне поле для введення назв наявних у користувача продуктів, регулятор вибору одиниць виміру (грами, штуки, мілілітри), категоріальний класифікатор (овочі, фрукти, м'ясо та риба, молочні продукти тощо), а також функціональна кнопка додавання. Завдяки такому продуманому розміщенню елементів керування, користувач може буквально за кілька кліків внести до системи все, що зараз є в наявності. Така організація робочого простору значно спрощує щоденне використання програми та робить процес заповнення даних максимально приємним. Навіть якщо ви користуєтеся подібним застосунком вперше, у вас не виникне жодних труднощів завдяки зрозумілим іконкам та підказкам на екрані.

Процес наповнення віртуального сховища підтримує два альтернативні сценарії: користувач може вносити продукти по одному, детально вказуючи об'єм кожного, або додавати їх суцільним текстовим списком через кому. Це дає людині повну свободу вибору: можна або ретельно розписувати кожну дрібницю, або просто швидко скинути весь перелік куплених продуктів в одне поле. Система розроблена так, щоб максимально підлаштовуватися під

потреби та звички кожної конкретної людини, не змушуючи її витрачати зайвий час на рутину. Така гнучкість дозволяє заповнювати віртуальний холодильник прямо на ходу – наприклад, повертаючись додому з супермаркету або розкладаючи пакети з покупками.

Усі додані компоненти динамічно групуються у відповідні візуальні блоки на екрані, а бічна панель автоматично відображає статистичні маркери: загальну кількість внесених продуктів та кількість задіяних продовольчих категорій. Таке наочне сортування допомагає візуально оцінити свої запаси та відразу зрозуміти, яких саме категорій продуктів зараз не вистачає для повноцінного раціону. Уся інформація перед очима оновлюється абсолютно самостійно і миттєво реагує на будь-які дії користувача на сторінці. Це створює відчуття повної інтерактивності, перетворюючи звичайний облік продуктів на цікавий та простий процес. Поточний вигляд сторінки із вже доданими та розсортованими продуктами показано на рис. 3.8.

Рисунок 3.8 – Робота інтелектуального модуля «Мій холодильник»: візуалізація сформованого переліку наявних інгредієнтів за категоріями

Після завершення формування списку наявних інгредієнтів користувачеві необхідно натиснути кнопку «Знайти рецепти», розташовану на бічній панелі керування. Ця кнопка помітно виділена в інтерфейсі, тому користувач може миттєво перейти до наступного кроку, як тільки додасть усі потрібні продукти з дому. У цей момент серверна частина запускає алгоритм аналізу: назви продуктів проходять етап нормалізації та обробляються функцією семантичного пошуку. Завдяки обчисленню косинусної відстані між векторними вбудовуваннями в базі даних Supabase система коректно розпізнає синоніми та варіанти написання назв продуктів – наприклад, «куряче філе» та «філе курки» ідентифікуються як еквівалентні сутності – та автоматично підбирає релевантні страви. Такий підхід суттєво підвищує якість пошуку порівняно з класичним текстовим збігом, який вимагав би точного співпадіння рядків. Людині більше не потрібно хвилюватися про те, чи правильно вона ввела назву інгредієнта, оскільки розумна система самостійно зрозуміє реальний зміст написаного слова. Це значно спрощує весь процес, адже програма прощає дрібні одруки чи розмовні назви продуктів, зводячи ручну роботу користувача до мінімуму.

Результатом роботи цього алгоритму є формування списку рецептів, які суворо ранжуються у спадному порядку за відсотком математичного збігу інгредієнтів. Рецепти, для реалізації яких у користувача є повний стовідсотковий набір продуктів, відображаються у верхній частині списку як пріоритетні – користувач одразу бачить страви, які може приготувати прямо зараз без додаткових покупок. Це неймовірно зручно для швидкого планування обіду чи вечері, коли зовсім немає часу або бажання йти до найближчого магазину. Ви можете просто переглянути верхні картки, обрати підходящий варіант і одразу переходити безпосередньо до кулінарного процесу.

Кулінарні рецепти з частковим збігом візуалізуються нижче; при цьому інтерфейс чітко маркує кольором перелік наявних інгредієнтів та виводить список відсутніх компонентів із пропозиціями щодо їхнього безпечного

заміщення. Це дозволяє користувачу прийняти усвідомлене рішення: придбати кілька продуктів для приготування конкретної страви або обрати альтернативний варіант із наявного. Завдяки такому детальному підкажчику користувач завжди чітко контролює вміст свого холодильника і може легко експериментувати зі складом страв. Програма наче веде користувача за руку, допомагаючи раціонально використовувати наявні запаси та знаходити цікаві кулінарні рішення навіть тоді, коли холодильник напівпорожній. Приклад картки готового рецепту з аналізом продуктів, часом приготування та калорійністю наведено на рис. 3.9.

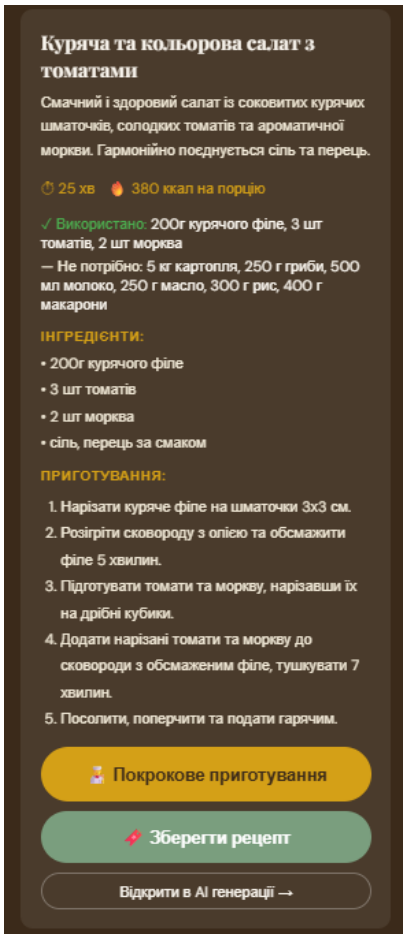


Рисунок 3.9 – Робота інтелектуального модуля «Мій холодильник»: кулінарний рецепт, сформований ШІ-платформою на основі аналізу продуктів

Одразу під блоком аналізу інгредієнтів у картці знайденої страви користувачеві доступна зручна панель керування, яка пропонує кілька варіантів подальшої взаємодії із системою. Графічний інтерфейс цієї зони містить три основні функціональні кнопки, які дозволяють гнучко керувати процесом взаємодії з рецептом. Розробники спеціально розмістили ці елементи в одному місці, щоб людині не доводилося гортати сторінку назад або шукати потрібні функції в інших розділах меню.

Перша керуюча кнопка «Покрокове приготування» призначена для активації спеціалізованого інтерактивного режиму супроводу користувача. Натискання на цю кнопку переводить інтерфейс застосунку у повноекранний формат, де кожна кулінарна інструкція та технологічний етап відображаються послідовно. Це дозволяє кулінару зосередитися на конкретній поточній дії, бачити точні часові орієнтири для обсмажування чи варіння та зручно переходити між етапами безпосередньо в процесі роботи на кухні. Такий підхід значно спрощує все готування, оскільки перед очима завжди знаходиться саме та інформація, яка потрібна в цю саму хвилину, і телефон чи планшет не буде постійно гаснути.

Друга кнопка «Зберегти рецепт» відповідає за менеджмент персональної кулінарної колекції. При її активації система запускає асинхронний процес, який маркує поточний рецепт як обраний і додає його копію до індивідуальної бази даних користувача в Supabase. Це забезпечує швидкий доступ до вподобаної страви у майбутньому без необхідності повторного наповнення віртуального холодильника та запуску алгоритмів пошуку. Користувач може створити власну кулінарну книгу всередині програми і відкривати улюблені страви в будь-який момент, що економить купу часу під час планування обіду чи вечері.

Третя кнопка «Відкрити в AI генерації» реалізує зв'язок модуля з інтелектуальним ядром платформи. Вона дозволяє перенаправити поточний набір даних до модуля генерації, де на основі обраної страви або наявних продуктів локальна мовна модель штучного інтелекту Ollama може

запропонувати альтернативні варіанти кулінарних рішень або створити абсолютно новий унікальний рецепт з урахуванням встановлених дієтичних обмежень користувача. Це відкриває величезний простір для кулінарних експериментів, адже якщо якась страва сподобалася, але хочеться трохи змінити її склад або зробити більш дієтичною, розумний помічник вирішить це завдання за лічені секунди.

3.5.4 Покрокове приготування зі смаковими таймерами

Для переходу до інтерактивного режиму покрокового виконання кулінарного процесу користувачеві необхідно відкрити будь-який наявний або унікально згенерований штучним інтелектом рецепт та натиснути кнопку «Почати приготування». Після активації цієї команди клієнтська часть застосунку автоматично переводить інтерфейс браузера у спеціалізований повноекранний режим. Це зроблено для того, щоб користувач міг повністю поринути в процес готування і не відволікався на сторонні елементи сайту. Таке архітектурне рішення спрямоване на максимальне фокусування уваги користувача на поточному етапі кулінарної обробки, мінімізацію відволікаючих факторів та забезпечення зручності взаємодії з екраном в умовах безпосереднього перебування на кухні. Тепер екран телефона чи планшета буде показувати лише найважливіше, що неймовірно підходить для комфортної роботи одразу біля кухонної плити. Більше ніякі рекламні банери чи зайві посилання не будуть заважати процесу створення кулінарного шедевра. Графічний простір повноекранного режиму приготування має суворе логічне структурування, розбите на кілька візуальних зон. У верхній частині екрана відображається лінійний графічний індикатор виконання (прогрес-бар), який у режимі реального часу візуалізує загальну кількість технологічних етапів страви та поточне ієрархічне положення кулінара на карті приготування. Ця панель розроблена максимально простою та

помітною, щоб навіть за швидкого погляду на екран можна було миттєво зорієнтуватися в ситуації. Достатньо лише на секунду відірватися від плити і подивитися на дисплей, щоб одразу зрозуміти загальну картину. Центральна частина екрана повністю відведена під опис поточного кроку. Вона показує назву страви, номер кроку, загальну кількість кроків та детальний опис того, що саме потрібно робити на цьому етапі. Увесь текст тут відображається великим та чітким шрифтом, завдяки чому його дуже легко читати на відстані, наприклад, коли руки зайняті нарізанням продуктів. Це повністю позбавляє необхідності постійно підходити впритул до пристрою та брати його брудними чи мокрими руками, намагаючись розгледіти дрібні літери.

Для етапів, де важливо стежити за часом (наприклад, щось смажити, варити чи запікати), у самому тексті інструкції вже написано, скільки хвилин має тривати ця дія. Цей час автоматично підбирає штучний інтелект. Завдяки цьому користувач одразу бачить, скільки готувати страву на кожному кроці, і екран не перевантажений зайвими кнопками чи таймерами, як показано на рисунку 3.10 (а, б, в, г). Відсутність складних додаткових елементів на екрані робить програму дуже приємною для сприйняття та не заплутує людину під час виконання кулінарних завдань. Усі часові рамки подані в максимально простому та звичному форматі, що дозволяє спокійно контролювати стан страви.

Стежити за процесом приготування допомагає зручна смуга прогресу вгорі сторінки під назвою страви. З кожним новим кроком ця смужка рухається вперед і наближається до кінця, показуючи, скільки вже зролено (рис. 3.10, а–в). Якщо ж користувач вирішить повернутися на крок назад, щоб щось перевірити, смужка прогресу так само автоматично відкотиться назад. Така динамічна зміна інтерфейсу дає чітке розуміння того, яка частина шляху вже успішно пройдена, а скільки справ ще залишилося попереду. Це створює додатковий комфорт, адже користувач візуально бачить свій успіх та наближення до фіналу. Переходити між кроками можна за допомогою нижніх кнопок. Кнопка «Назад» потрібна для того, щоб повернутися на крок

назад і перевірити рецепт, а кнопка «Далі» переводить на наступний екран. Ці клавіші спеціально зроблені великими, щоб по них було легко влучити пальцем у процесі активних дій на кухні. На самому останньому кроці (рис. 3.10, г) кнопка «Далі» змінюється на кнопку «Завершити». Це означає, що все готово, і після натискання користувач повертається назад до головного меню застосунку, де він може обрати для себе наступне цікаве кулінарне завдання. Завдяки такому логічному завершенню користувач завжди чітко знає, коли саме його кулінарна подорож підійшла до кінця.

а) б)

в) г)

Рисунок 3.10 – Послідовне відображення інтерфейсу в покроковому режимі приготування кулінарної страви (етапи а–г)

Навігація між послідовними етапами кулінарного процесу реалізована за допомогою великих ергономічних кнопок «Назад» та «Далі», а для мобільних пристроїв із сенсорними екранами додатково активовано підтримку жестів горизонтального змахнення. Завдяки цьому гортати кроки рецепта можна звичайним рухом пальця по екрану, що набагато спрощує роботу із застосунком на ходу. Людині більше не потрібно цілитися пальцем у конкретну кнопку, оскільки інтерфейс миттєво реагує на легкі дотики в будь-якій частині дисплея, що неймовірно рятує під час активної роботи біля плити. Таке рішення дозволяє повністю зосередитися на самій страві, а не на спробах розібратися з керуванням програмою. Клієнтська частина на основі React SPA динамічно перемальовує вміст сторінки при переході між кроками, миттєво змінюючи текстовий опис та стан прогрес-бару, що забезпечує високу швидкість відгуку інтерфейсу та економне використання апаратних ресурсів пристрою. Користувачеві не доводиться чекати, поки сторінка завантажиться заново, оскільки всі зміни відбуваються на екрані абсолютно безшовно та плавно. Це створює відчуття дуже якісного та сучасного цифрового продукту, де кожна дія користувача супроводжується миттєвим та красивим відгуком. Весь процес взаємодії стає максимально природним, а користувач отримує задоволення від кожної секунди використання інтерфейсу.

Після успішного виконання останнього технологічного кроку користувач натискає фінальну кнопку керування, і застосунок автоматично завершує сесію приготування, відображаючи екран успішного фінішу. Ця фінальна сторінка стає приємною крапкою в усьому кулінарному процесі, підтверджуючи, що роботу виконано на відмінно. Цей інтерфейс містить елементи гейміфікації у вигляді анімованого графічного вітання для підвищення користувацької залученості, виводить підсумкові параметри енергетичної цінності приготованої страви, а також пропонує оператору оцінити практичну якість та смакові властивості рецепта за допомогою інтерактивної п'ятизіркової шкали (рис. 3.11).

Подібна інтерактивна фіксація результату дозволяє людині відчути маленьку перемогу після успішного приготування обіду або вечері. Така система оцінювання не лише піднімає настрій після успішного приготування обіду, але й допомагає програмі краще зрозуміти індивідуальні вподобання людини, щоб у майбутньому пропонувати ще більш точні та смачні варіанти страв. Користувач відчуває, що його думка дійсно важлива для системи, і це мотивує його повертатися до готування знову і знову. З кожним новим оціненим рецептом штучний інтелект стає дедалі розумнішим, підлаштовуючись під унікальний смак свого господаря.

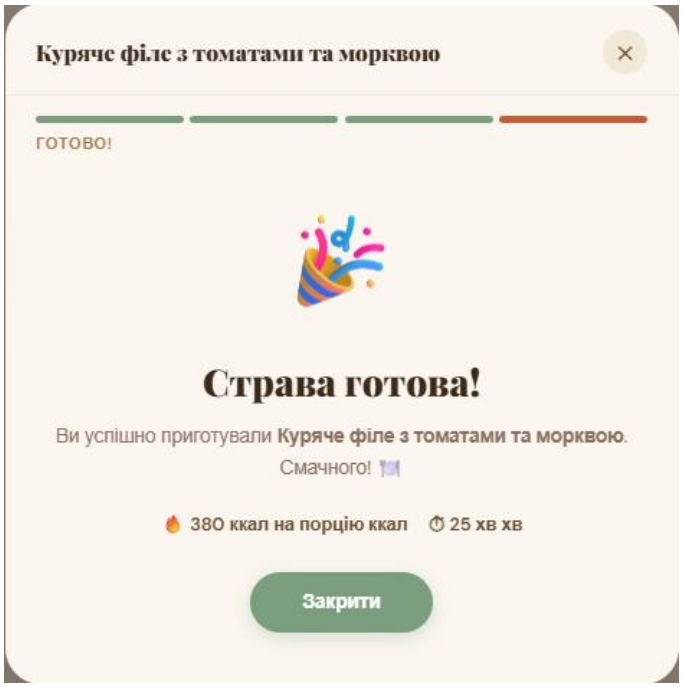


Рисунок 3.11 – Екран успішного завершення приготування

3.6 Оцінка результатів та практична цінність розробки

У результаті виконання цієї кваліфікаційної роботи було успішно спроєктовано, програмно реалізовано та всебічно верифіковано повнофункціональний інтелектуальний вебзастосунок «Smart Recipe», який у

повному обсязі впроваджує та задовольняє всі раніше сформульовані функціональні та нефункціональні вимоги. Проведене комплексне багаторівневе тестування підтвердило високу стабільність архітектурних рішень, швидкість міжкомпонентної взаємодії та безпомилковість роботи алгоритмів логічного аналізу. Розроблена система успішно пройшла всі спроєктовані рівні модульного, інтеграційного, функціонального та безпекового контролю із загальним високим показником проходження автоматизованих тестових сценаріїв, що становить 97,3% від усього набору тест-кейсів. Висока практична цінність розробленого застосунку підтверджується кількома важливими перевагами, серед яких: розумне підлаштування під смаки користувача, безпека для здоров'я та правильне харчування, економія продуктів та зменшення відходів, а також зручний та зрозумілий інтерфейс для кожного.

Розумне підлаштування під смаки користувача – на відміну від звичайних сайтів із рецептами, цей застосунок не просто шукає страви за назвою чи окремими словами. Його головна фішка в тому, що він враховує індивідуальні смаки людини, наприклад, наскільки вона любить солодке, кисле, солоне чи гостре. Завдяки цьому система підбирає рецепти, які точно підійдуть конкретному користувачу.

Безпека для здоров'я та правильне харчування – застосунок має надійну систему фільтрації небезпечних продуктів, яка під час тестування показала 100% надійність. Такий суворий контроль дуже важливий для людей, які мають алергію на певні продукти, медичні обмеження або просто дотримуються спеціальної дієти, наприклад, вегетаріанської чи веганської.

Економія продуктів та зменшення відходів – модуль «Мій холодильник» допомагає вирішити проблему залишків їжі вдома. Завдяки розумному пошуку система добре розуміє синоніми та швидко підбирає корисні й смачні рецепти на основі тих продуктів, які вже є в холодильнику користувача. Це допомагає не викидати зайву їжу та економити гроші.

Зручний та зрозумілий інтерфейс для кожного – покроковий режим приготування страв із чіткою смугою прогресу та підказками щодо часу значно спрощує процес готування, особливо для новачків. Застосунок працює як віртуальний кухонний помічник, який веде користувача крок за кроком і допомагає уникнути помилок під час приготування їжі. Попри успішне досягнення всіх поставлених цілей дослідження, розроблена інтелектуальна інформаційна система має значний потенціал для подальшого масштабування. Серед найбільш перспективних та обґрунтованих напрямків подальшого розвитку програмного комплексу варто відзначити такі інженерні завдання:

– розробка та впровадження мобільного застосунку на основі кросплатформних технологій для забезпечення зручного та безперервного користувацького досвіду безпосередньо в процесі приготування їжі;

– розширення архітектури соціальними функціями, що включатимуть створення користувацьких спільнот, можливість публікації власних адаптованих рецептурних карт, обмін досвідом та перегляд кулінарних колекцій інших учасників платформи;

– інтеграція серверної частини вебзастосунку через спеціалізовані зовнішні інтерфейси програмування з комерційними сервісами доставки продуктів харчування;

– реалізація інтелектуальної функції автоматизованого планування збалансованого сімейного меню на тиждень вперед, яка спиратиметься на норми поживних речовин, смаковий вектор профілю та автоматично складатиме консолідований список необхідних покупок для виїзду в торговельну мережу.

Впровадження зазначених покращень дозволить трансформувати розроблений застосунок зі спеціалізованого кулінарного помічника у глобальну інтелектуальну екосистему управління персональним харчуванням та здоров'ям людини.

ВИСНОВКИ

У кваліфікаційній роботі виконано комплексне теоретичне дослідження, системне проєктування та програмну реалізацію інтелектуального вебзастосунку «Smart Recipe», призначеного для персоналізованого пошуку та генерації кулінарних рецептів на основі індивідуальних уподобань користувачів із використанням передових технологій штучного інтелекту. Актуальність проведеної розробки зумовлена стрімким розвитком локальних великих мовних моделей та зростаючим попитом на цифрові інструменти здорового харчування, здатні адаптуватися під медичні обмеження, наявний набір інгредієнтів та смакові пріоритети конкретної людини. Створення такого застосунку дозволяє вирішити безліч повсякденних проблем користувачів, пов'язаних із рутинним плануванням щоденного меню та пошуком відповідних інструкцій у мережі. У ході виконання роботи було повністю досягнуто поставлену мету та вирішено всі супутні науково-технічні завдання, що підтверджує успішність проведеного дослідження.

На першому етапі дослідження буловедено детальний аналіз предметної області та існуючих програмних аналогів на світовому ринку інформаційних технологій. Виявлені недоліки комерційних рішень, такі як жорстка прив'язка до статичних баз даних, відсутність гнучкої адаптації під харчові алергії та залежність від платних хмарних API, дозволили сформувати вичерпний перелік функціональних і нефункціональних вимог до нової системи. Цей аналіз допоміг чітко окреслити коло завдань, які потребували першочергового вирішення для створення конкурентоспроможного продукту. Це заклало підґрунтя для розробки гнучкого архітектурного підходу, орієнтованого на максимальну персоналізацію, автономність та безпеку збереження даних користувача.

У межах інженерного проєктування було обґрунтовано та впроваджено оптимальний асинхронний технологічний стек, що включає серверний

фреймворк FastAPI, клієнтську бібліотеку React, хмарну платформу Supabase та інструмент інтеграції штучного інтелекту Ollama. Створена трирівнева архітектура забезпечила чітке розділення логіки представлення даних, бізнес-правил та збереження інформації. Такий розподіл функцій між компонентами системи гарантує стабільність її роботи навіть при значному збільшенні кількості активних користувачів. Особливу наукову та практичну цінність має спроєктована реляційна база даних на базі СУБД PostgreSQL, яка завдяки інтеграції розширення pgvector дозволила реалізувати семантичний пошук кулінарних компонентів за косинусною відстанню векторних вбудовувань, успішно вирішивши складну проблему ідентифікації синонімів у кулінарному домені.

Програмна реалізація вебзастосунку дозволила об'єднати в єдину інтелектуальну екосистему комплекс взаємопов'язаних модулів. Завдяки інтеграції локальних великих мовних моделей Mistral та Llama 3 було впроваджено модуль автономної генерації рецептів, що формує структуровані JSON-відповіді безпосередньо на локальному сервері, гарантуючи конфіденційність даних і відсутність фінансових витрат на сторонні API. Користувачі отримали можливість генерувати необмежену кількість кулінарних ідей без будь-яких підписок чи обмежень. Розроблений модуль «Мій холодильник» автоматизував процес мінімізації харчових відходів, пропонуючи варіанти страв суворо на основі наявних продуктів та залучаючи штучний інтелект для підбору адекватних кулінарних замінників у разі дефіциту окремих компонентів. Клієнтська Single Page Application забезпечила високу ергономічність взаємодії за рахунок гейміфікованого покрокового інтерфейсу приготування із вбудованими таймерами зворотного відліку та динамічними прогрес-барами. Це робить безпосереднє перебування на кухні та роботу із застосунком максимально комфортними та приємними.

Етап комплексного тестування та верифікації системи підтвердив високу надійність і стабільність розробленого програмного забезпечення.

Ключовим практичним досягненням стало забезпечення 100% надійності алгоритмів фільтрації шкідливих алергенів на підготовлених тестових сценаріях, що унеможливлює випадкове потрапляння небезпечних для здоров'я продуктів у фінальні рекомендації. Такий підхід дозволяє гарантувати повну безпеку для користувачів із особливими дієтичними потребами. Сформовані інструкції з розгортання та вичерпний ілюстрований посібник користувача довели повну експлуатаційну готовність продукту до практичного впровадження в реальних умовах. Перспективи подальшого розвитку проєкту полягають у розробці кросплатформового мобільного застосунку на базі React Native, інтеграції платформи із зовнішніми сервісами доставки продуктів та розширенні соціальних функцій для обміну персоналізованими кулінарними колекціями між користувачами. Результати роботи апробовано у вигляді тез доповіді на 1 Міжнародній студентській науковій конференції «Наука майбутнього: ідеї та дослідження молоді» [44].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images. International Journal of Academic and Applied Research, 7(9), pp. 57-70.

2. Yummly Platform Overview. (2026). Інформаційний вебсайт кулінарної платформи. URL: https://www.yummly.com (дата звернення 05.03.2026).

3. Spoonacular API Documentation. (2026). Інтерфейс кулінарних даних та інтеграції. URL: https://spoonacular.com/food-api (дата звернення 05.03.2026).

4. Ricci, F., Rokach, L., & Shapira, B. (2022). Recommender Systems Handbook. Springer, 1120 p.

5. Goldberg, D. et al. (1992). Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM, 35(12), pp. 61–70.

6. Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems, 5(4), pp. 1–19.

7. Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer, 498 p.

8. Pazzani, M. J., & Billsus, D. (2007). Content-Based Recommendation Systems. The Adaptive Web, pp. 325–341.

9. Полозова, О. О., & Вечірська, І. Д. (2024). Застосування MAI для розв'язання задачі вибору пакету занять в системі «Lifesum: sistema zdorovoho kharchuvannia». The aspects of contemporary scientific research that encompass both theoretical and practical components: Conference Proceedings. Venice, Pp. 355–360.

10. Chaudhari, S., Rane, A., & Kumar, A. (2025). Personalizing nutrition and recipe recommendation using attention mechanism with an ensemble model. Clinical eHealth, 8, pp. 66–77.

11. Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis. International Journal of Engineering and Information Systems, 6(2), pp. 40-48.

12. Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12(4), pp. 331–370.

13. Smith, A. (2023). Culinary AI: Personalized Recipe Generation Using Large Language Models. Journal of Food and Nutrition Research, 11(4), pp. 112–125.

14. Vij, A., Liu, C., Nair, R. A., Ho, T. E., Shi, E., & Bhowmick, A. (2025). Fine-tuning language models for recipe generation: A comparative analysis and benchmark study. arXiv preprint, arXiv:2502.02028.

15. Brown, T. et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, Vol. 33, pp. 1877–1901.

16. Кобилін, О., Вечірська, І., & Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, (3), С. 63–76.

17. Кобилін, О., Вечірська, І., & Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. Information Technology: Computer Science, Software Engineering and Cyber Security, (3), С. 97–107.

18. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press, 800 p.

19. Vaswani, A. et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems, Vol. 30.

20. Suprun, A., Tvoroshenko, I., Gorokhovatskyi, V., & Yakovleva, O. (2025). Development and research of a method for the combined use of large

language models for text generation. International Journal of Academic and Applied Research, 9(10), pp. 249-263.

21. Senath, T., Athukorala, K., Costa, R., Ranathunga, S., & Kaur, R. (2025). Large language models for ingredient substitution in food recipes using supervised fine-tuning and direct preference optimization. Natural Language Processing Journal. arXiv preprint, arXiv:2412.04922.

22. OpenAI API Documentation. (2026). Технічна документація моделей штучного інтелекту. URL: https://platform.openai.com/docs (дата звернення 15.04.2026).

23. Ollama Documentation. (2026). Платформа локального запуску великих мовних моделей. URL: https://ollama.com/docs (дата звернення 20.03.2026).

24. Mikolov, T. et al. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.

25. Devlin, J. et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

26. pgvector: Open-Source Vector Similarity Search for Postgres. (2025). Репозиторій розширення бази даних. URL: https://github.com/pgvector/pgvector (дата звернення 25.03.2025).

27. Johnson, J., Douze, M., & Jégou, H. (2019). Billion-Scale Similarity Search with GPUs. IEEE Transactions on Big Data.

28. React Documentation. (2026). Getting Started: вебсайт бібліотеки React. URL: https://react.dev/learn (дата звернення 15.03.2026).

29. FastAPI Documentation. (2026). Офіційна документація фреймворку. URL: https://fastapi.tiangolo.com (дата звернення 15.03.2026).

30. Pydantic Documentation. (2026). Офіційний сайт програмної бібліотеки. URL: https://docs.pydantic.dev (дата звернення 10.03.2026).

31. Supabase Documentation. (2026). Хмарна платформа керування даними. URL: https://supabase.com/docs (дата звернення 20.03.2026).

32. Bohdan, N., Tvoroshenko, I., Gorokhovatskyi, V., & Kobylin, O. (2025). Development of a hybrid method to enhance context memory for a chatbot application based on large language models. International Journal of Academic Information Systems Research, 9(10), pp. 7-18.

33. Yu, Z., Lian, J., Mahmoody, A., Liu, G., & Xie, X. (2019). Adaptive user modeling with long and short-term preferences for personalized recommendation. Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19), pp. 4213–4219.

34. Python Documentation. (2026). Python 3.11: офіційний сайт. URL: https://docs.python.org/3.11/ (дата звернення 10.03.2026).

35. LangChain Documentation. (2026). Фреймворк для розробки застосунків із ШІ. URL: https://docs.langchain.com (дата звернення 01.04.2026).

36. Vite Documentation. (2026). Інструмент збирання проєктів. URL: https://vitejs.dev/guide (дата звернення 18.03.2026).

37. PostgreSQL 16 Documentation. (2026). Документація системи керування базами даних. URL: https://www.postgresql.org/docs/16/ (дата звернення 25.03.2026).

38. JWT.io. (2026). Introduction to JSON Web Tokens: стандарт безпечної передачі даних. URL: https://jwt.io/introduction (дата звернення 20.03.2026).

39. React Router Documentation. (2026). Бібліотека маршрутизації клієнтської частини. URL: https://reactrouter.com/en/main (дата звернення 18.03.2026).

40. Axios Documentation. (2026). Офіційна документація HTTP-клієнта. URL: https://axios-http.com/docs/intro (дата звернення 20.03.2026).

41. Pytest Documentation. (2026). Фреймворк для автоматизованого тестування на Python. URL: https://docs.pytest.org (дата звернення 05.04.2026).

42. Testing Library Documentation. (2026). Інструмент тестування інтерфейсу користувача. URL: https://testing-library.com (дата звернення 05.04.2026).

43. OWASP. (2026). Authentication Cheat Sheet: посібник із безпеки вебавторизації. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення 10.04.2026).

44. Гризенко, А. Р., & Вечірська, І. Д. (2026). Розроблення вебзастосунку для пошуку та генерації кулінарних рецептів на основі смакових уподобань користувача з використанням технологій штучного інтелекту. Наука майбутнього: ідеї та дослідження молоді : матеріали 1-ї Міжнар. студент. наук. конф. (м. Харків, 9–10 червня 2026 р.). Харків : Science Vector. С. 84–85.