Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 «Інженерія програмного забезпечення»

На тему: *Розробка дискусійної онлайн-платформи кулінарних рецептів*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-21-1
_____ *Дейнека В. П.*

Керівник кваліфікаційної роботи _____ */ Гриценко А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2025

Криворізький національний університет
Факультет: *Інформаційних технологій*
Кафедра: *Моделювання та програмного забезпечення*
Освітньо-кваліфікаційний рівень: *бакалавр*
Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу
студенту групи ІПЗ-21-1 Дейнека Данило Петрович

1. Тема: *Розробка дискусійної онлайн-платформи кулінарних рецептів* затверджено наказом по КНУ №119 від «10» квітня 2025 р.
2. Термін подання студентом закінченого проекту «10» червня 2025 р.
3. Вихідні дані по роботі: *Пояснювальна записка: 59 сторінки, 23 рисунків, 1 додаток, 21 використаних у роботі джерел*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Аналіз онлайн-платформ в галузі кулінарії. Проектування та розробка дискусійної онлайн-платформи. Тестування онлайн-платформи .
5. Перелік графічного демонстраційного матеріалу: Приклади існуючих програмних рішень. Архітектура системи. Структура бази даних. Діаграми використання. Блок-схеми основних алгоритмів. Приклади роботи розробленої програми (інтерфейс корисмтувача).

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*13.02.2025-20.02.2025*
2	*Аналіз інформаційних джерел*	*21.02.2025-09.03.2025*
3	*Визначення вимог до програмного забезпечення*	*10.03.2025-18.03.2025*
4	*Розробка функціональної схеми*	*19.03.2025-23.03.2025*
5	*Розробка алгоритмів*	*24.03.2025-31.03.2025*
6	*Розробка бази даних*	*01.04.2025-14.04.2025*
7	*Розробка програмного забезпечення*	*15.04.2025-13.05.2025*
8	*Тестування програмного забезпечення*	*14.05.2025-20.05.2025*
9	*Оформлення пояснювальної записки*	*21.05.2025-12.06.2025*
10	*Розробка демонстраційних матеріалів*	*13.06.2025-17.06.2025*

Дата видачі завдання: «_____»_____ 2025 р.

Студент _____ Дейнека Д.П.

Керівник роботи _____ Гриценко А. М.

РЕФЕРАТ

ВЕБ-СИСТЕМА, КУЛІНАРНІ РЕЦЕПТИ, , БАЗА ДАНИХ, DJANGO, MYSQL, REST API, АВТОРИЗАЦІЯ, OAUTH 2.0, МОБІЛЬНИЙ ІНТЕРФЕЙС.

Обсяг пояснювальної записки: 59 сторінок. Кількість рисунків — 23, таблиць — 7, додатків — 1, використаних джерел — 21.

Об’єктом розробки є інформаційно-дискусійна веб-система для публікації, перегляду та обговорення кулінарних рецептів користувачами.

Метою роботи є створення повноцінної інтерактивної веб-платформи, що дозволяє користувачам додавати рецепти, здійснювати їх пошук за назвою, категорією та інгредієнтами, залишати коментарі, оцінки, а також авторизуватись через популярні соціальні мережі. Особливу увагу приділено адаптивності інтерфейсу для різних пристроїв та інтеграції з зовнішніми сервісами автентифікації.

Для досягнення поставленої мети проведено аналіз предметної області, визначено функціональні та нефункціональні вимоги до системи. Розроблено архітектуру веб-застосунку з використанням мови програмування Python та фреймворку Django. Серверна частина реалізована у вигляді REST API з використанням Django REST Framework. У якості бази даних обрано MySQL. Реалізовано функції авторизації через протокол OAuth 2.0 з інтеграцією Google і Facebook. Для клієнтського інтерфейсу використано адаптивну верстку, що забезпечує коректну роботу на смартфонах, планшетах і десктопах. Додатково розроблено структуру бази даних у вигляді ER-діаграми, UML-діаграми взаємодії та переходів між сторінками, макети інтерфейсу у форматі wireframe.

У результаті розробки створено стабільний, безпечний та адаптивний веб-застосунок, що дозволяє автоматизувати зберігання, публікацію та обговорення кулінарних рецептів. Робота демонструє практичне застосування знань з програмної інженерії, веб-розробки, проектування баз даних, побудови REST API та впровадження сучасних методів автентифікації.

ABSTRACT

WEB SYSTEM, CULINARY RECIPES, , DATABASE, DJANGO, MYSQL, REST API, AUTHORIZATION, OAUTH 2.0, MOBILE INTERFACE.

Explanatory note volume: 59 pages. Number of figures — 23, tables — 7, appendices — 1, sources used — 21.

The object of development is an information and discussion web system for publishing, viewing and discussing culinary recipes by users.

The goal of the work is to create a full-fledged interactive web platform that allows users to add recipes, search for them by name, category and ingredients, leave comments, ratings, and also log in through popular social networks. Particular attention is paid to the adaptability of the interface for different devices and integration with external authentication services.

To achieve the goal, an analysis of the subject area was conducted, functional and non-functional requirements for the system were determined. The architecture of the web application was developed using the Python programming language and the Django framework. The server part was implemented in the form of a REST API using the Django REST Framework. MySQL was chosen as the database. Authorization functions were implemented via the OAuth 2.0 protocol with Google and Facebook integration. Adaptive layout was used for the client interface, ensuring correct operation on smartphones, tablets and desktops. Additionally, the database structure was developed in the form of an ER diagram, a UML diagram of interaction and transitions between pages, and interface layouts in wireframe format. As a result of the development, a stable, secure and adaptive web application was created that allows you to automate the storage, publication and discussion of culinary recipes.

The work demonstrates the practical application of knowledge in software engineering, web development, database design, REST API construction and the implementation of modern authentication methods.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві інформаційні технології відіграють ключову роль у забезпеченні доступу до знань, сервісів і щоденного функціонування людей. Однією з важливих галузей, де цифрові рішення мають значний попит, є кулінарія — сфера, яка об’єднує мільйони користувачів з усього світу незалежно від віку, професії чи рівня знань. Зростання популярності здорового способу життя, підвищення інтересу до приготування їжі вдома та широке використання мобільних і веб-застосунків сприяли зростанню потреби у якісних платформах, що надають зручний доступ до перевірених кулінарних рецептів.

Наявні рішення на ринку, такі як AllRecipes, Cookpad, Tasty тощо, демонструють високий рівень функціональності, однак не завжди враховують специфіку локальних користувачів, культурні особливості харчування та необхідність адаптації під сучасні вимоги доступності та інтерактивності. Багато з існуючих платформ або надто фрагментовані, або орієнтовані виключно на англомовну аудиторію, або не забезпечують можливість гнучкої взаємодії між користувачами. Тому виникає об’єктивна потреба у розробці вітчизняної платформи, яка б задовольняла потреби українських користувачів, підтримувала локалізацію, пропонувала гнучкий пошук рецептів за інгредієнтами, категоріями, рівнем складності, а також забезпечувала інструменти соціальної взаємодії — оцінки, коментарі, обговорення.

Метою даного дипломного проекту є розробка повнофункціональної веб-системи для зберігання, пошуку, перегляду та обміну кулінарними рецептами. Система повинна забезпечувати авторизацію користувачів через соціальні мережі за допомогою протоколу OAuth 2.0, підтримку адаптивного інтерфейсу для різних типів пристроїв (десктоп, планшет, смартфон), а також надавати можливість користувачам публікувати рецепти, додавати фотографії страв, автоматично перераховувати інгредієнти відповідно до кількості порцій, залишати оцінки та коментарі.

У межах реалізації системи значна увага приділяється зручності користування інтерфейсом, оптимізації структури бази даних, побудові надійної серверної частини з використанням фреймворків Python та Django, а також дотриманню принципів сучасного UI/UX дизайну. Інтерфейс розробляється відповідно до стандартів адаптивної верстки, що дозволяє забезпечити стабільну роботу системи на пристроях з різною роздільною здатністю екранів. Авторизація через популярні соціальні мережі значно спрощує доступ до системи, одночасно підвищуючи рівень безпеки обробки персональних даних.

Практична цінність проєкту полягає в тому, що запропоноване рішення може бути використане як основа для комерційного продукту, освітньої платформи або корпоративної системи управління кулінарним контентом. Система також може бути масштабована та доповнена модулями рекомендацій, дієтичного планування, підбору рецептів за залишками продуктів або взаємодії з іншими сервісами. Структура системи дозволяє легко інтегрувати нові функціональні модулі, а відкритість технологій забезпечує гнучкість у розвитку.

В процесі виконання кваліфікаційної роботи були досліджені аналогічні рішення, сформульовані функціональні вимоги, побудовано архітектуру системи, спроєктовано та реалізовано базу даних, серверну логіку та клієнтську частину інтерфейсу. Результати тестування підтвердили працездатність системи та відповідність її запланованим функціональним можливостям. Розроблена система кулінарних рецептів є сучасним, надійним і масштабованим рішенням, яке здатне задовольнити актуальні потреби користувачів у швидкому, зручному та структурованому доступі до кулінарної інформації.

1 АНАЛІЗ ДИСКУСІЙНИХ ОНЛАЙН-ПЛАТФОРМ В ГАЛУЗІ КУЛІНАРІЇ

1.1 Актуальність розробки дискусійних онлайн-платформ у галузі кулінарії

У сучасному цифровому суспільстві інтерес до кулінарії та здорового харчування стрімко зростає. З огляду на глобалізацію, урізноманітнення гастрономічних вподобань та зростаючий попит на швидкий доступ до якісної та достовірної інформації, розробка ефективних інформаційно-довідкових систем у галузі кулінарії є надзвичайно актуальною. Інтернет став головним джерелом кулінарних знань для мільйонів користувачів, що обумовлює необхідність створення інтелектуальних платформ, які дозволяють швидко знаходити, зберігати, аналізувати та поширювати рецепти з урахуванням індивідуальних вподобань.

Поява великої кількості кулінарних блогів, відеоінструкцій, онлайн-шкіл і соціальних кулінарних спільнот створює великий обсяг інформації, яка потребує структурування, фільтрації та зручного інтерфейсу взаємодії з користувачем. Водночас користувачі прагнуть не лише знайти рецепт, а й мати можливість оцінити його складність, час приготування, переглянути відгуки, адаптувати кількість інгредієнтів під потрібну кількість порцій. Це створює запит на розробку платформ, які виходять за межі статичних архівів рецептів і стають повноцінними динамічними сервісами з елементами соціальної взаємодії.

Сучасні кулінарні платформи повинні бути багатофункціональними, доступними з різних пристроїв, інтегрованими із соціальними мережами та здатними до персоналізації контенту. Це обумовлює потребу у впровадженні інструментів адаптивної верстки, авторизації через OAuth 2.0, динамічного фільтрування контенту, систем рейтингу та рекомендацій. Крім того, зважаючи на інтернаціоналізацію аудиторії, важливо враховувати мовну підтримку та локалізацію даних.

Особливої актуальності набуває розробка кулінарних інформаційно-довідкових систем у контексті освітніх, соціальних та побутових потреб. Для молоді платформи стають джерелом навчання кулінарії, для сімей — способом організації харчування вдома, для професіоналів — каналом поширення власного досвіду та рецептів. Платформа також може бути корисною у сфері туризму та гастрономічного маркетингу, сприяючи популяризації національної кухні, розвитку регіональних кулінарних традицій та підтримці локальних виробників продуктів.

Інформаційно-довідкові кулінарні системи можуть значно покращити якість харчування населення, надаючи користувачам можливість швидко планувати раціон, враховувати дієтичні обмеження, використовувати сезонні продукти. Завдяки інтеграції з мобільними технологіями такі сервіси доступні будь-де і будь-коли, що відповідає вимогам ринку щодо мобільності та зручності.

Таким чином, актуальність створення сучасної кулінарної платформи визначається як практичними потребами широкої аудиторії користувачів, так і загальними тенденціями в розвитку цифрових сервісів, які орієнтовані на зручність, персоналізацію та взаємодію в реальному часі. Це зумовлює необхідність розробки інформаційно-довідкової системи нового покоління, яка поєднує зручність інтерфейсу, гнучкість архітектури, інтелектуальну обробку даних та можливості соціальної взаємодії.

1.2 Огляд існуючих програних рішень дискусійних онлайн-платформ з огляду кулінарних рецептів.

Інформаційні системи для обміну кулінарними рецептами є популярним напрямом серед онлайн-сервісів. Вони виконують функції збереження, категоризації, пошуку, оцінювання та коментування рецептів страв. Сучасні користувачі очікують від таких платформ зручності, адаптивного інтерфейсу, авторизації через соціальні мережі, візуальної привабливості та активної інтерактивності. Проведений аналіз існуючих систем виявив ряд сильних

сторін, однак також показав наявність обмежень та недоліків, які можна врахувати при створенні нової системи.

1. AllRecipes (https://www.allrecipes.com/) (див. рис. 1.1). Одна з найбільш відомих міжнародних платформ з публікації та пошуку рецептів. Пропонує потужний механізм фільтрації, рейтинги, коментарі та інтеграцію з мобільними застосунками. Має велику базу рецептів, які додають як користувачі, так і редактори.

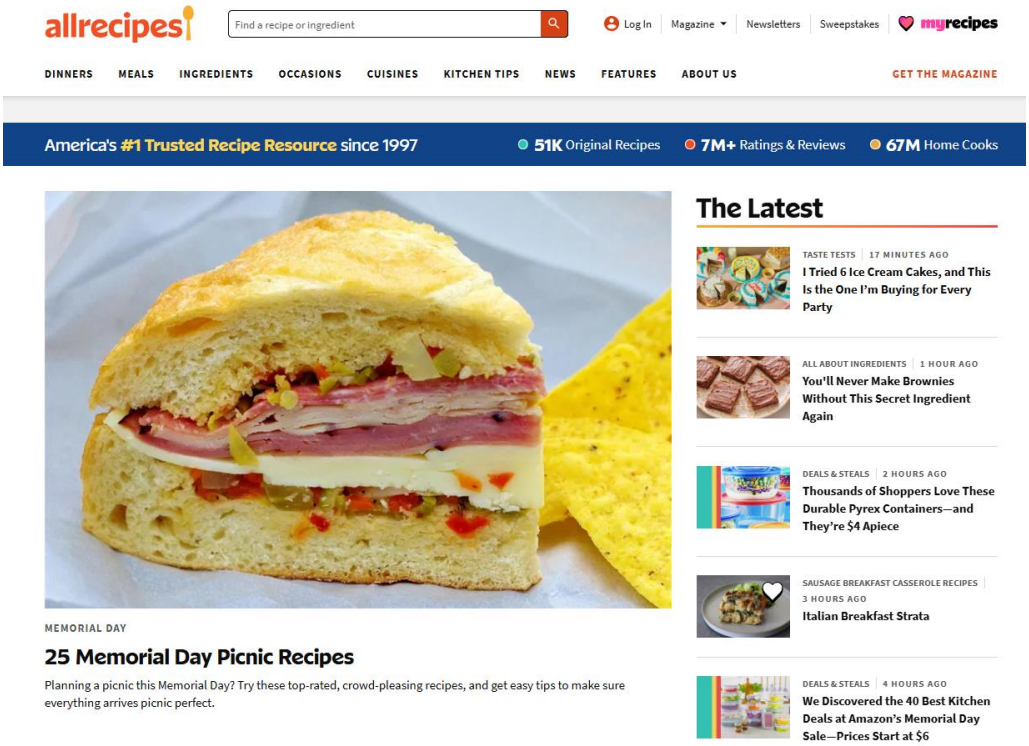

Рисунок 1.1 – Візуальний інтерфейс платформи AllRecipes

Переваги: багатофункціональний пошук, відеорецепти, рекомендації за історією.

Недоліки: відсутність гнучкого пошуку за інгредієнтами, перевантаженість інтерфейсу, неможливість змінювати кількість порцій з автоматичним перерахунком інгредієнтів.

2. Cookpad (https://cookpad.com/) (див. рис. 1.2). Японська платформа, орієнтована на простоту та взаємодію між користувачами. Рецепти публікуються у вигляді коротких інструкцій зі світлинами.

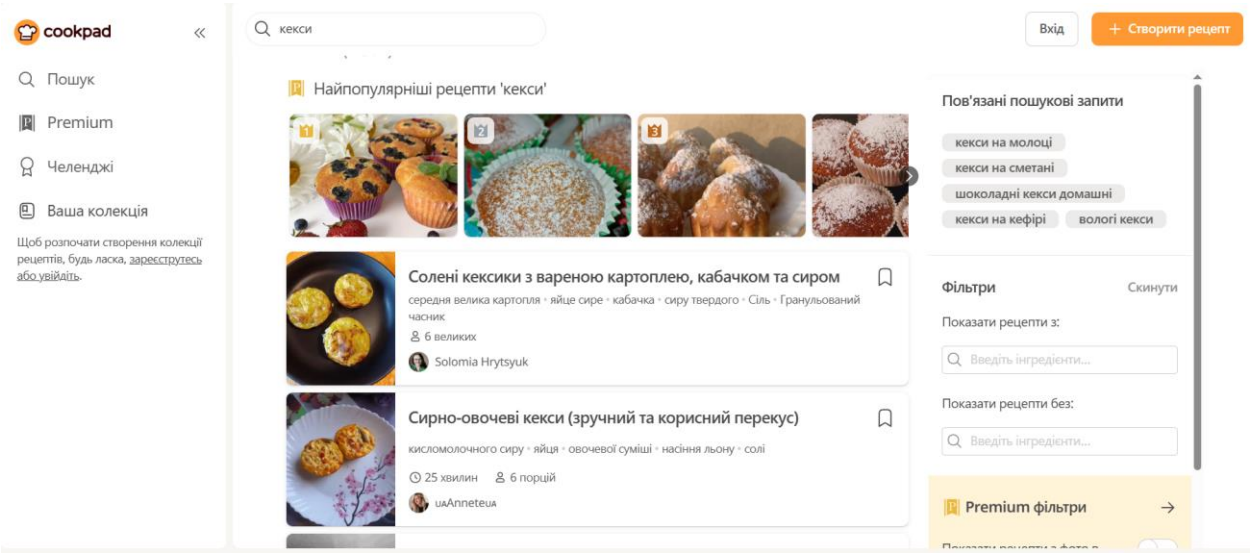

Рисунок 1.2 – Візуальний інтерфейс платформи Cookpad

Переваги: фокус на спільноті, мобільна орієнтованість, проста структура.

Недоліки: обмежена деталізація рецептури, слабка категоризація, мінімальні можливості для оцінювання.

3. Yummly (https://www.yummly.com/). Система, що використовує алгоритми персоналізації та підбірки страв за смаками користувача. Має інтеграцію з розумною побутовою технікою.

Переваги: адаптивна логіка пошуку, сильна інтеграція з зовнішніми платформами, естетичний дизайн.

Недоліки: частина функцій доступна тільки після реєстрації, комунікаційна складова обмежена.

4. Ukrainian Culinary Portals (наприклад, Smachno.ua). Багато локальних платформ орієнтовані на українську аудиторію. Здебільшого вони мають статичну структуру, з мінімальним інтерфейсом.

Переваги: український контент, простота.

Недоліки: застарілий дизайн, відсутність авторизації, слабка інтерактивність, відсутність адаптивності.

1.3 Формування узагальнених вимог до системи

У процесі аналізу предметної області та вивчення аналогічних систем, таких як AllRecipes, Cookpad, Tasty, Ukrainian Culinary Portals було зібрано та систематизовано ключові функціональні характеристики, які мають бути реалізовані в сучасній веб-платформі кулінарних рецептів. Ці вимоги базуються як на потребах потенційних користувачів, так і на загальновизнаних практиках розробки веб-застосунків.

Передусім, для забезпечення широкої доступності системи важливо реалізувати зручний, інтуїтивно зрозумілий інтерфейс користувача. Він має бути адаптований під різні типи пристроїв, включаючи десктопи, планшети та смартфони. Адаптивний дизайн дозволяє зберігати цілісність інтерфейсу незалежно від розміру екрана, що істотно підвищує зручність користування платформою.

Одним із ключових вимог є впровадження авторизації через соціальні мережі — такі як Google, Facebook і Twitter. Використання стандарту OAuth 2.0 забезпечує безпечний доступ до системи без зберігання паролів на стороні сервера. Це знижує поріг входу для користувачів і водночас підвищує рівень довіри до сервісу. Крім того, профільні дані користувача (ім’я, фото, посилання) можуть використовуватися для формування персонального кабінету, а також для відображення авторства рецептів.

Важливим елементом функціональності є механізм додавання, редагування та перегляду кулінарних рецептів. Кожен рецепт має містити назву, опис, фотографію страви, список інгредієнтів із зазначенням кількості, кроки приготування та додаткові параметри — такі як складність, тривалість приготування, тип страви. Також реалізовано динамічне обчислення кількості інгредієнтів залежно від вибраної кількості порцій, що полегшує приготування страви для будь-якої кількості людей.

Пошукова система є критично важливою складовою. Вона повинна забезпечувати пошук рецептів за назвою, ключовими словами, інгредієнтами, а також за категоріями (десерти, перші страви, гарніри, напої тощо). Система

також повинна включати можливості фільтрації за складністю, популярністю, оцінками та іншими параметрами.

Однією з важливих особливостей системи є підтримка активності користувачів через можливість залишати коментарі до рецептів, виставляти оцінки, переглядати рейтинги. Така функціональність створює навколо платформи спільноту користувачів, сприяє її поширенню, а також дозволяє новим користувачам швидше зорієнтуватися у великому масиві рецептів.

Дизайн інтерфейсу має відповідати сучасним вимогам UX/UI. Важливо дотримуватись логічної структури розміщення елементів, використовувати зрозумілу навігацію, акцентні кольори для важливих елементів та приємну типографіку. Інтерфейс повинен виглядати професійно, але залишатися дружнім до користувача. Особливу увагу слід приділяти швидкості завантаження сторінок та інтерактивності елементів.

Таким чином, функціональні вимоги до інформаційно-дискусійної системи кулінарних рецептів включають реалізацію зручного та адаптивного інтерфейсу, підтримку авторизації через соціальні сервіси, повний цикл роботи з рецептами (додавання, редагування, перегляд), динамічний розрахунок інгредієнтів, розширений пошук і систему рейтингу та коментарів. Усі ці вимоги формують сучасний рівень якості веб-платформи, орієнтованої на кінцевого користувача.

Запропонована система, реалізована на основі цих вимог, здатна задовольнити сучасні потреби користувачів, забезпечити зручний доступ до великої бази кулінарного контенту та створити навколо себе активну онлайн-спільноту. Такий підхід дозволяє конкурувати з існуючими платформами та водночас пропонувати нові можливості, яких бракує в аналогічних сервісах.

Висновки по розділу

У межах даного розділу було здійснено поглиблений критичний аналіз сучасного стану ринку програмних продуктів, що належать до класу інформаційно-довідкових систем кулінарного спрямування. Було детально

досліджено функціональні можливості та архітектурні особливості таких популярних платформ, як AllRecipes, Cookpad, Tasty, Food Network, а також окремих локалізованих сервісів. Аналіз проводився з урахуванням факторів зручності користування, адаптивності інтерфейсу, підтримки соціальної інтеграції, алгоритмів пошуку, персоналізації, динаміки взаємодії між користувачами та надійності зберігання структурованої інформації.

Результати аналізу показали, що хоча більшість наявних рішень володіють широким функціоналом, значна частина з них не повною мірою відповідає очікуванням користувачів з точки зору локалізації, персоналізованого контенту та можливостей редагування рецептів під власні потреби. Також виявлено певну недосконалість у логіці пошуку за складом інгредієнтів, що часто знижує релевантність результатів. У деяких сервісах спостерігається перевантаженість інтерфейсу зайвими візуальними елементами, що ускладнює навігацію, особливо на мобільних пристроях.

Окрему увагу було приділено оцінці технологій, що лежать в основі зазначених платформ. Було визначено, що сучасні системи, які претендують на масове використання, повинні реалізовуватись із застосуванням відкритих стандартів, підтримкою адаптивної верстки, використанням RESTful API або GraphQL, засобів безпечної авторизації (OAuth 2.0) та можливістю масштабування. Аналізуючи переваги таких рішень, можна стверджувати, що платформи кулінарних рецептів мають виходити за межі звичайного сховища контенту, трансформуючись у динамічні інтерактивні сервіси з активною соціальною взаємодією.

Крім того, проведено порівняння функціональних характеристик платформ, де особливої уваги надано таким параметрам, як зручність додавання рецептів, редагування кількості порцій, наявність рейтингової системи, можливість пошуку за кількома параметрами, авторизація через соцмережі, а також підтримка мультимовного інтерфейсу. Це дозволило виявити як сильні, так і слабкі сторони існуючих рішень та сформувати

обґрунтовану потребу у створенні нової, вдосконаленої системи з урахуванням виявлених недоліків.

Узагальнюючи результати аналізу, можна зробити висновок, що існує реальна потреба у створенні нової інформаційно-довідкової системи кулінарних рецептів, яка не лише відповідатиме актуальним вимогам користувачів, але й запропонує нові можливості. Така система має бути зручною у користуванні, мати привабливий адаптивний інтерфейс, підтримку авторизації через популярні соціальні платформи, зручні інструменти фільтрації та сортування рецептів, можливість автоматичного перерахунку інгредієнтів, та інтегровану соціальну складову — коментарі, рейтинги, підписки.

На основі цього аналізу сформовано узагальнені вимоги до розроблюваної системи, які будуть використані на наступних етапах проєктування архітектури, бази даних, користувацького інтерфейсу та серверної частини. Це дозволить побудувати ефективну, конкурентоспроможну та орієнтовану на користувача інформаційну систему, що відповідатиме сучасним технологічним стандартам у галузі розробки веб-застосунків.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ДИСКУСІЙНОЇ ОНЛАЙН ПЛАТФОРМИ

2.1 Вибір архітектури та засобів реалізації системи

У процесі розробки інформаційно-дискусійної платформи для обміну кулінарними рецептами важливим етапом був обґрунтований вибір технологій, які забезпечують ефективність, безпеку, масштабованість та зручність користування системою. Обрана архітектура проєкту відповідає вимогам сучасних веб-застосунків і забезпечує можливість подальшого розвитку системи.

Для реалізації системи було обрано класичну архітектуру «клієнт-сервер» з REST API, де фронтенд відповідає за взаємодію з користувачем, а бекенд — за логіку, зберігання даних та авторизацію. Це забезпечує розділення функціональних завдань, полегшує підтримку і масштабування системи (див. рис. 2.1).

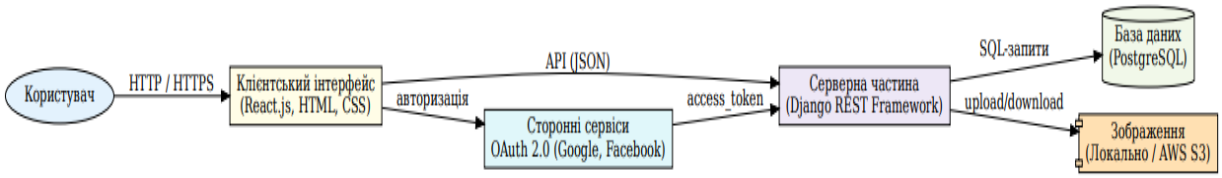

Рисунок 2.1 – Архітектура системи

REST API дозволяє взаємодіяти між клієнтом і сервером через HTTP-запити. Також, в майбутньому, дозволяє реалізувати мобільний застосунок без зміни серверної логіки.

Розробку серверної частини програмного забезпечення буде розроблено на мові Python з використанням фреймворку Django REST Framework. Django забезпечує швидку розробку, високу безпеку, інтеграцію з базами даних, а REST-модуль дозволяє створювати API.

Для реалізації Frontend частини використовується HTML5, CSS3, JavaScript та бібліотека React.js — вона дозволяє створювати адаптивний, швидкий та інтерактивний інтерфейс, що зручно для застосунків із динамічними оновленнями.

Аутентифікацію користувачів реалізовано через OAuth 2.0 з інтеграцією сторонніх постачальників (Google, Facebook та інші). Використання цього протоколу авторизації надає вагомі переваги, зокрема:

- забезпечує високий рівень безпеки;
- зменшує бар’єр входу для нових користувачів;
- сприяє використанню профілю користувача (аватар, ім’я, посилання).

Зберігання інформації у системі передбачає використання СКБД для оперування структурованими данними та стандартне зберігання зображень на сервері з можливістю переходу до AWS S3 або Firebase Storage у майбутньому при масштабуванні системи.

Інтерфейс розроблено з використанням адаптивної верстки, що забезпечує його коректне відображення на смартфонах, планшетах і ПК.

У якості платформи для хостингу системи можуть бути використані сервіси з простим CI/CD, автоматичним масштабуванням та SSL (наприклад Render, Vercel або Heroku).

Обрані для реалізаціє системи технології відповідають вимогам сучасного рівня веб-застосунків: вони безпечні, швидкі, адаптовані для командної розробки та масштабування. Реалізація на базі Django + React дозволяє ефективно відокремити логіку клієнта й сервера, забезпечити інтерактивний інтерфейс та підтримати високий рівень користувацького досвіду.

2.2 Розробка бази даних системи

У процесі розробки інформаційно-дискусійної веб-платформи кулінарних рецептів виникла необхідність у зберіганні та обробці великої

кількості структурованих даних. До таких даних належать: облікові записи користувачів, рецепти, інгредієнти, коментарі, оцінки, категорії страв тощо. Для забезпечення цілісності, надійності та ефективної роботи з цими даними було прийнято рішення використовувати реляційну базу даних.

Основними критеріями вибору системи керування базами даних (СКБД) стали: підтримка складних зв’язків між сутностями, масштабованість, надійність, продуктивність, легкість інтеграції з серверною частиною застосунку та зручність у розробці й супроводі.

На основі аналізу наявних варіантів було обрано MySQL — одну з найпопулярніших реляційних СКБД з відкритим вихідним кодом. Цей вибір був обґрунтований такими перевагами:

1) Поширеність і стабільність - MySQL є однією з найпоширеніших систем у світі, що активно використовується у проєктах різного масштабу. Вона має розвинену екосистему та велику кількість документації, що важливо для розробника.
2) Підтримка складної структури даних, СКБД дозволяє створювати таблиці з зовнішніми ключами, індексами, обмеженнями цілісності, що критично для коректного моделювання взаємозв’язків між сутностями системи.
3) MySQL має високу швидкість обробки запитів та оптимізований механізм індексації, що дозволяє ефективно виконувати пошук по назвах рецептів, інгредієнтах, фільтрацію по категоріях і сортування результатів.
4) MySQL доступна безкоштовно і підтримується спільнотою, що дозволяє уникати витрат на ліцензії та забезпечує повний контроль над СКБД.
5) MySQL підтримується більшістю фреймворків бекенду (у тому числі Django через сумісний драйвер), а також має зручний графічний інтерфейс — MySQL Workbench для візуалізації структури БД та написання запитів.

6) У разі зростання обсягів даних MySQL може бути розгорнута у вигляді кластера або переміщена на віддалений сервер без зміни логіки роботи застосунку (масштабованість та портованість).

Використання бази даних у системі є критично необхідним для збереження, організації, обробки, та швидкого доступу жл інформації про користувачів та рецепти. Обрана СКБД MySQL повністю відповідає технічним та функціональним вимогам проєкту, а також дозволяє забезпечити стабільну та ефективну роботу системи у перспективі її розширення.

Розроблено структуру таблиць бази даних та заявки між ними, що забезпечує виконання функціоналу системи. Структурну діаграму бази даних наведено на рисунку 2.2.

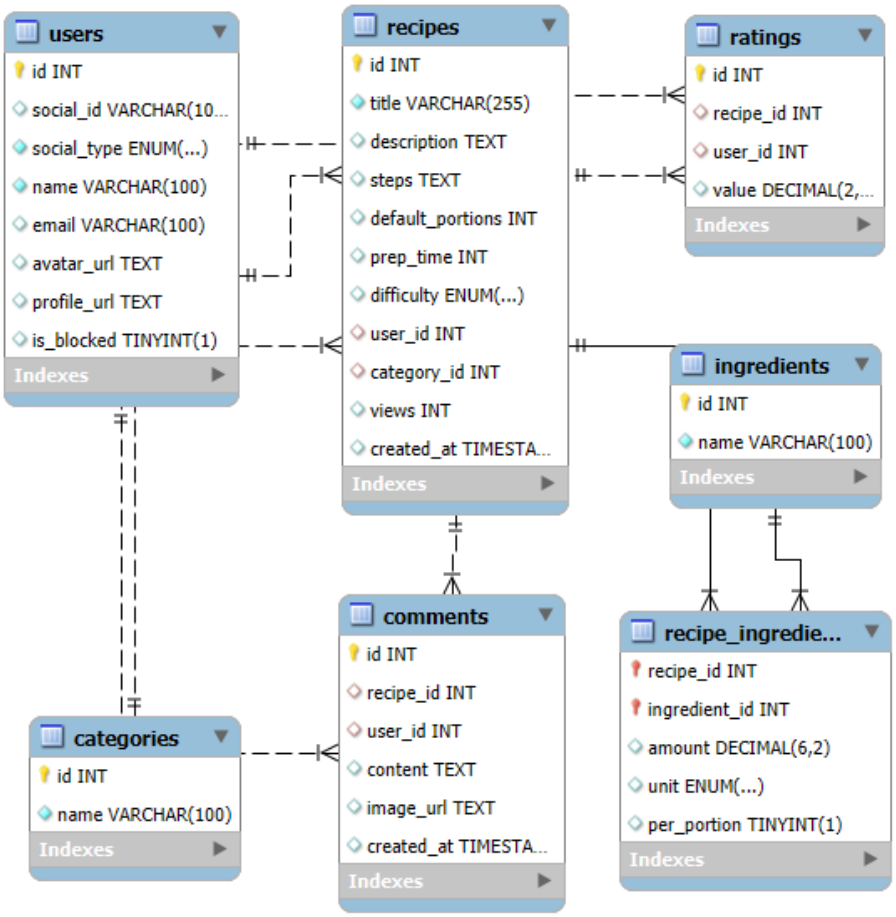

Рисунок 2.2 – ER-діаграма бази даних

В Таблиці Users зберігається інформація про зареєстрованих користувачів в системі. Структура таблиці наведена в таблиці 2.1.

Таблиця 2.1 - Структура таблиці users бази даних

Поле	Тип	Опис
id	PK, INT, AUTO	Унікальний ідентифікатор
social_id	VARCHAR	ID користувача в соц. мережі
social_type	ENUM	Тип соцмережі (facebook, google…)
name	VARCHAR	Ім’я користувача
avatar_url	VARCHAR	Посилання на аватар
profile_url	VARCHAR	Посилання на соціальний профіль
is_blocked	BOOLEAN	Статус блокування

В Таблиці Recipes зберігається інформація про додані рецепти в системі. Структура таблиці наведена в таблиці 2.2.

Таблиця 2.2 - Структура таблиці recipes бази даних

Поле	Тип	Опис
id	PK, INT, AUTO	Унікальний ідентифікатор рецепта
title	VARCHAR	Назва рецепта
description	TEXT	Опис
steps	TEXT	Етапи приготування
default_portions	INT	Кількість порцій за замовчуванням
prep_time	INT	Час приготування в хвилинах
difficulty	ENUM	(easy, medium, hard)
user_id	FK → users.id	Автор рецепта
category_id	FK → categories.id	Категорія рецепта
created_at	TIMESTAMP	Дата створення
views	INT	Кількість переглядів

В Таблиці Categories зберігається інформація про додані категорії рецептів в системі. Структура таблиці наведена в таблиці 2.3.

Таблиця 2.3 - Структура таблиці categories бази даних

Поле	Тип	Опис
id	PK, INT	Унікальний ідентифікатор
name	VARCHAR	Назва категорії (гарніри, десерти...)

В Таблиці Ingredients зберігається інформація про додані інгредієнти страв в базу даних системи. Структура таблиці наведена в таблиці 2.4.

Таблиця 2.4 - Структура таблиці ingredients бази даних

Поле	Тип	Опис
id	PK, INT	Унікальний ідентифікатор
name	VARCHAR	Назва інгредієнта

В Таблиці Recipe_ingredients зберігається інформація про додані інгредієнти до страв (як складові страви), що додані в базу даних системи. Структура таблиці наведена в таблиці 2.5.

Таблиця 2.5 - Структура таблиці Recipe_ingredients бази даних

Поле	Тип	Опис
recipe_id	FK → recipes.id	Рецепт
ingredient_id	FK → ingredients.id	Інгредієнт
amount	DECIMAL	Кількість (в грамах/мл)
unit	ENUM	'g', 'ml'
per_portion	BOOLEAN	Чи вказано на 1 порцію

(зв’язок N:M між recipes та ingredients)

В Таблиці Comments зберігається інформація про додані коментарі користувачів системи до страв. Структура таблиці наведена в таблиці 2.6.

Таблиця 2.6 - Структура таблиці comments бази даних

Поле	Тип	Опис
id	PK, INT	Унікальний ідентифікатор
recipe_id	FK → recipes.id	До якого рецепта
user_id	FK → users.id	Хто коментував
content	TEXT	Текст коментаря
image_url	VARCHAR (NULLABLE)	Фото до коментаря
created_at	TIMESTAMP	Дата створення

В Таблиці Ratings зберігається інформація про оцінки рецептів страв користувачами системи. Структура таблиці наведена в таблиці 2.7.

Таблиця 2.7 - Структура таблиці ratings бази даних

Поле	Тип	Опис
id	PK, INT	Унікальний ідентифікатор
recipe_id	FK → recipes.id	Рецепт
user_id	FK → users.id	Хто оцінив
value	INT (1–5)	Оцінка

Для побудови структури бази даних, яка забезпечує надійне зберігання, ефективний пошук та управління лікарськими засобами, було розроблено відповідний SQL-запит (див. рис. 2.3).

Він реалізує створення таблиць відповідно до концептуальної моделі даних та забезпечує встановлення зв’язків між ними через зовнішні ключі. Основними сутностями системи виступають: Користувачі, Препарати, Пошукові запити, AI-рекомендації та Регіони. Кожна таблиця має чітко визначені атрибути, первинні ключі та типи даних, що відповідають вимогам цілісності та нормалізації даних.

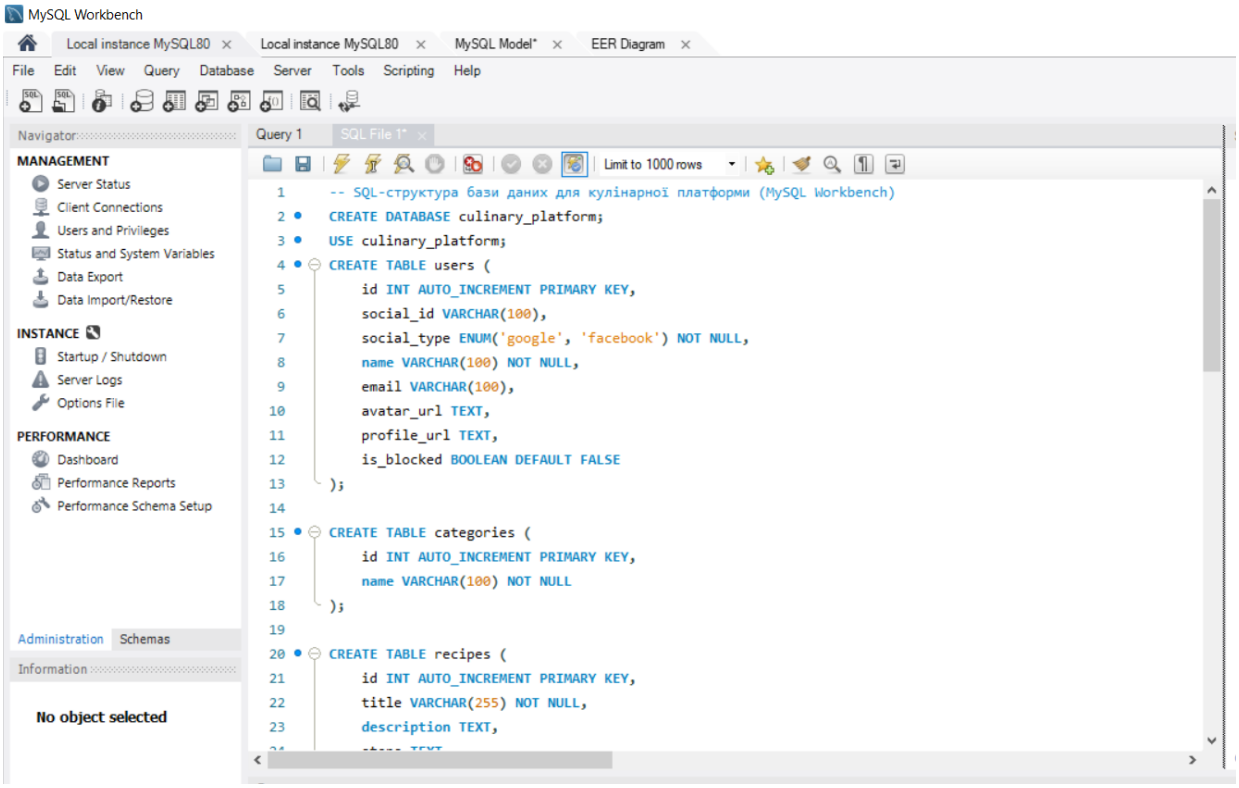

Рисунок 2.3 – Фрагмент вікна MySQL Workbenchпід час створення БД

Окрему увагу було приділено створенню зв’язків типу «один-до-багатьох» (1:N) між користувачами та їхніми запитами, а також між запитами та AI-рекомендаціями. SQL-запит дозволяє автоматизовано створити повну структуру реляційної бази в середовищі PostgreSQL та забезпечити можливість масштабованого підключення численних клієнтських застосунків.

2.3. Розробка візуального інтерфейсу користувача системи

Для інформаційно-дискусійної системи кулінарних рецептів було реалізовано адаптивний, інтуїтивно зрозумілий і функціональний веб-інтерфейс, що враховує потреби користувачів у пошуку, перегляду, оцінці, коментуванні та додаванні рецептів. Розробка проводилася в декілька етапів:

Першим етапом розробки інтерфейсу став аналіз користувацьких сценаріїв: пошук рецептів за назвою чи інгредієнтами, перегляд деталей приготування, фільтрація за категоріями, залишення коментарів та оцінок, автентифікація через соціальні мережі. Це дозволило визначити основні типи сторінок та ключові елементи, які мають бути на кожній з них.

На основі функціональних вимог створено шаблони сторінок (wireframes), що візуалізують компонування інтерфейсу без прив’язки до стилю. Серед реалізованих шаблонів:

- Головна сторінка з рекомендованими рецептами (див. рис. 2.4);
- Сторінка пошуку з фільтрами та полем запиту (див. рис. 2.5);
- Детальна сторінка рецепта з інгредієнтами, інструкцією, рейтингом і коментарями (див. рис. 2.6);
- Сторінка авторизації та профілю користувача(рис. 2.7);
- Додавання або редагування рецепту;
- Сторінка налаштувань користувача;
- Сторінка адміністрування системи.

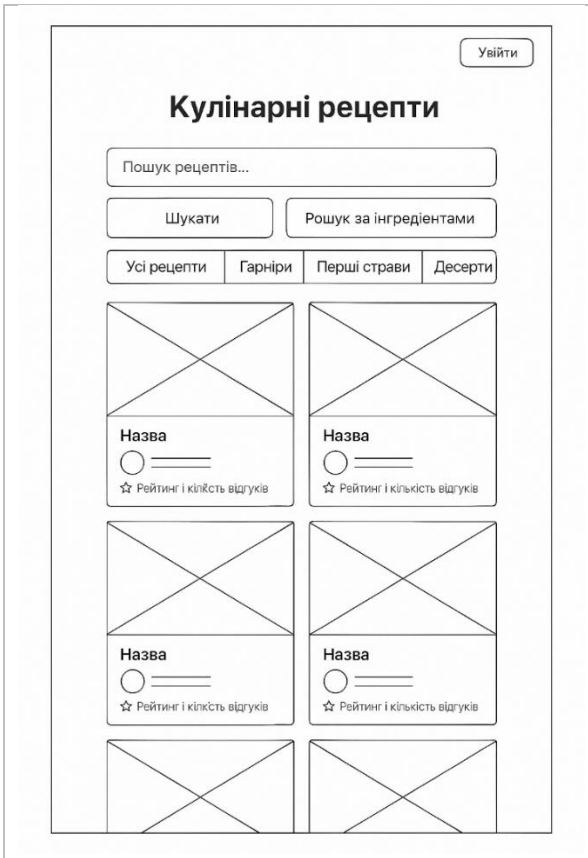

Рисунок 2.4 – Шаблон головної сторінки

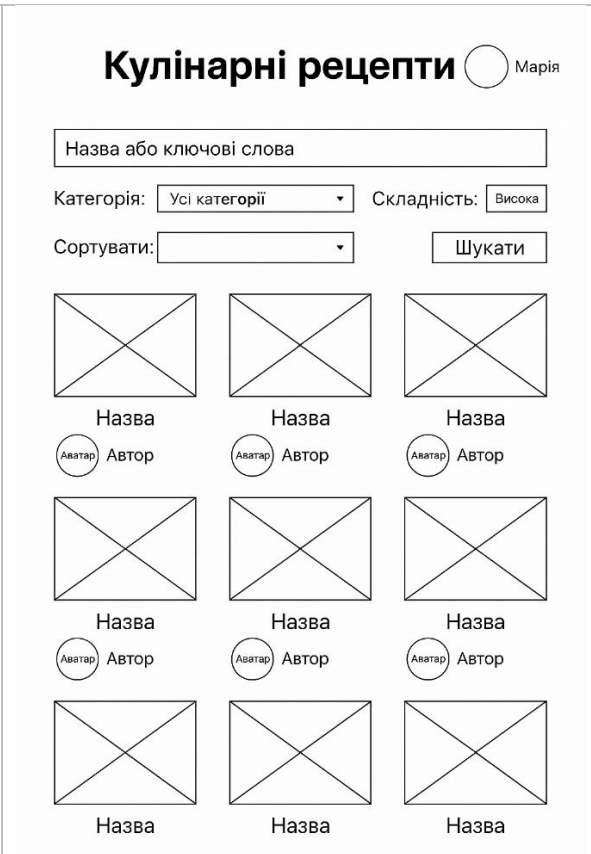

Рисунок 2.5 -Шаблон сторінки пошуку

Було побудовано UML-діаграму переходів між сторінками (рис. 2.8), яка описує можливі переходи користувача в системі. Діаграма допомогла систематизувати логіку навігації, уникнути дублювання функціоналу та оптимізувати шляхи взаємодії.

Інтерфейс оформлено в мінімалістичному стилі з акцентом на читабельність і простоту навігації. Було обрано світлу кольорову схему, зручні для читання шрифти, та уніфіковані кнопки взаємодії. В основі верстки використано бібліотеку Tailwind CSS, яка забезпечила швидке прототипування та консистентність стилів.

Рисунок 2.6 – Шаблон сторінки редагування рецепту

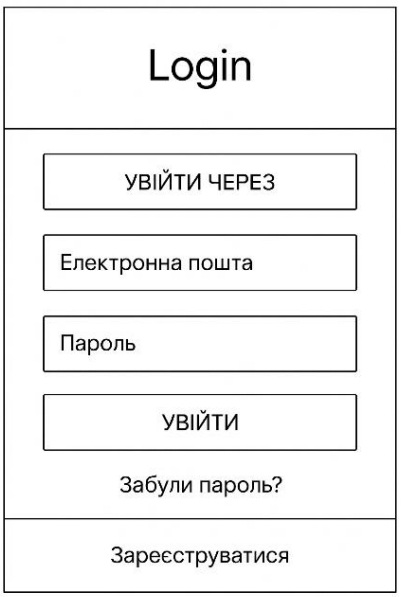

Рисунок 2.7 -Шаблон сторінки авторизації користувачав

Для забезпечення зручності користування на всіх пристроях реалізовано адаптивну верстку інтерфейсу з використанням сіткової системи. Шаблони сторінок були адаптовані для трьох форматів:

- Планшет (середній екран) наведено на рисунку 2.9;
- Смартфон (мобільна версія) наведено на рисунку 2.10;

- Монітор (десктопна версія) на рисунку 2.11.

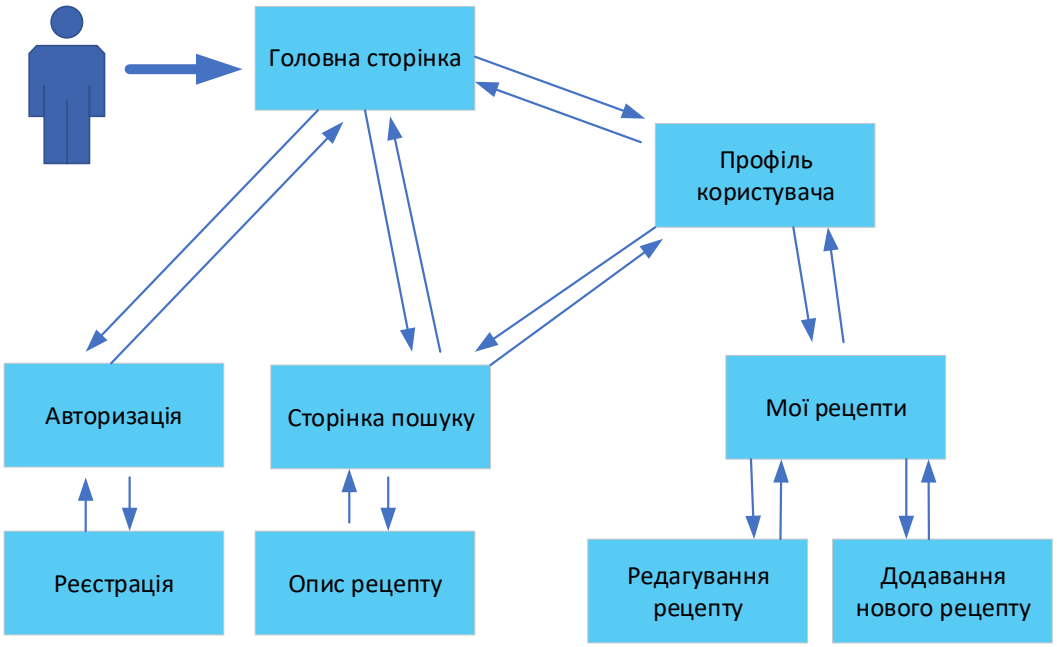

Рисунок 2.8 - UML-діаграма переходів між сторінками.

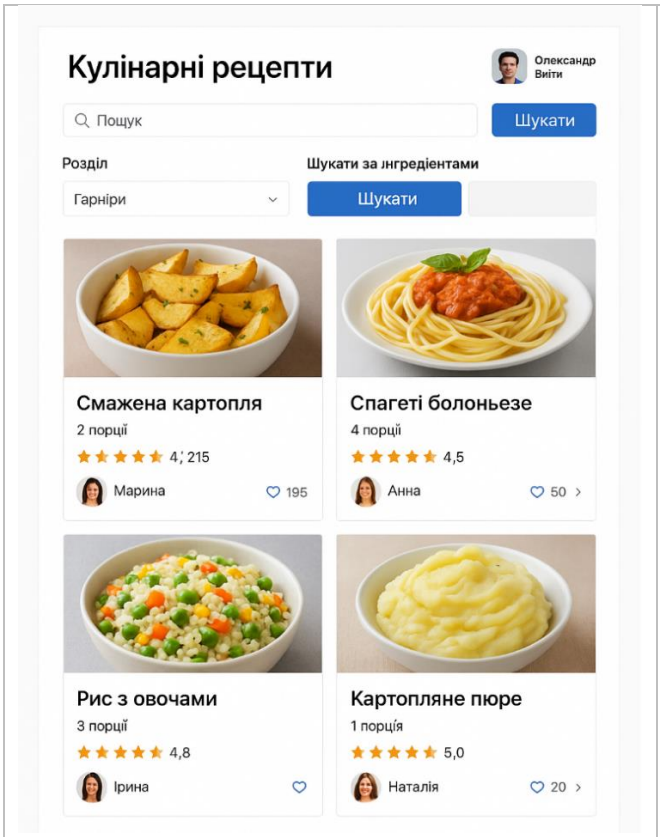

Рисунок 2.9 – Вигляд головної сторніки на планшеті

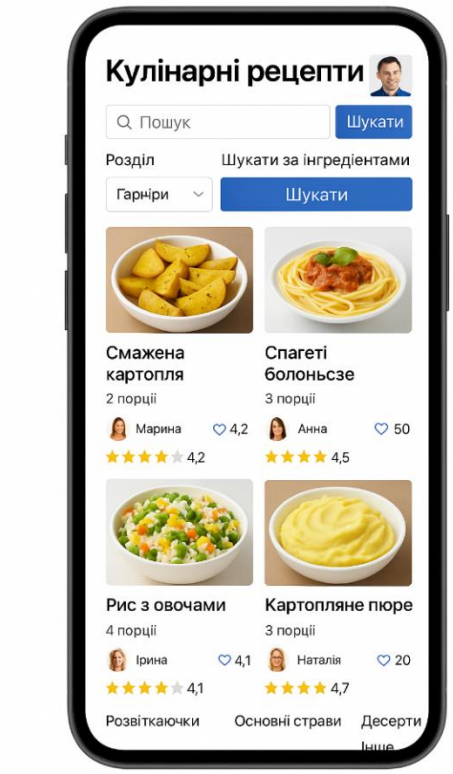

Рисунок 2.10 - Вигляд головної сторніки на смартфоні

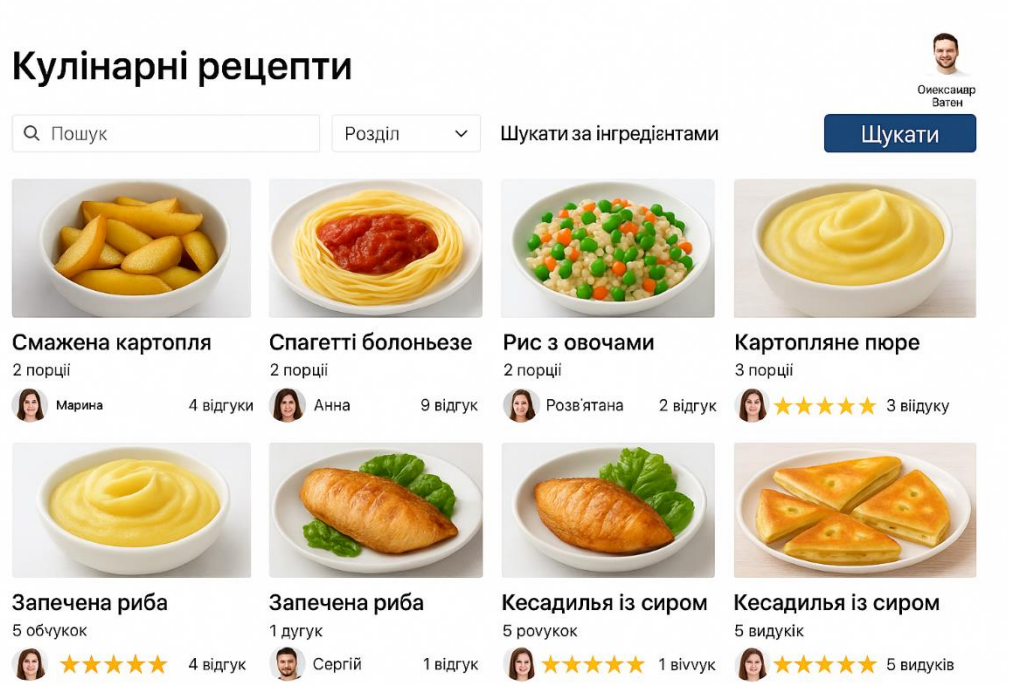

Рисунок. 2.11 – Вигляд адаптивної верстки сторінки на моніторі ПК

Розробка інтерфейсу системи кулінарних рецептів пройшла поетапно: від аналізу функціоналу до створення адаптивного дизайну з урахуванням сучасних вимог до UI/UX. Це дозволило створити зручний, зрозумілий і привабливий інтерфейс, що сприяє активному використанню платформи та позитивному досвіду взаємодії з нею.

2.4. Розробка програмного забезпечення інформаційно-довідкової системи

Процес розробки серверної частини системи кулінарних рецептів здійснювався з використанням мови програмування Python та фреймворку Django з розширенням Django REST Framework. Одним з ключових факторів ефективної реалізації проєкту стало обґрунтоване обрання середовища розробки. Для цієї мети було використано інтегроване середовище програмування (IDE) PyCharm, яке забезпечує широкий набір інструментів для професійної розробки веб-застосунків на Python.

Середовище PyCharm (див. рис. 2.12) було обрано завдяки своїм численним перевагам, зокрема зручному редактору коду з підсвіткою синтаксису, автодоповненням, інструментами відладки, інтеграцією з системами контролю версій (наприклад, Git) та можливістю керування віртуальним середовищем і залежностями проєкту. Крім того, PyCharm підтримує Django як один з основних фреймворків, що дозволяє швидко створювати шаблони проєкту, автоматично генерувати структуру директорій, керувати міграціями бази даних та запускати локальний сервер.

Розробка серверної частини включала створення окремих додатків (apps), кожен з яких відповідав за певний функціональний блок системи: користувачі, рецепти, коментарі, оцінки тощо. У кожному додатку створювались відповідні модулі: models.py для опису структури таблиць бази даних, serializers.py для серіалізації моделей у формат JSON, views.py для обробки HTTP-запитів, а також urls.py для маршрутизації. Структура проекту наведена на рисунку 2.13.

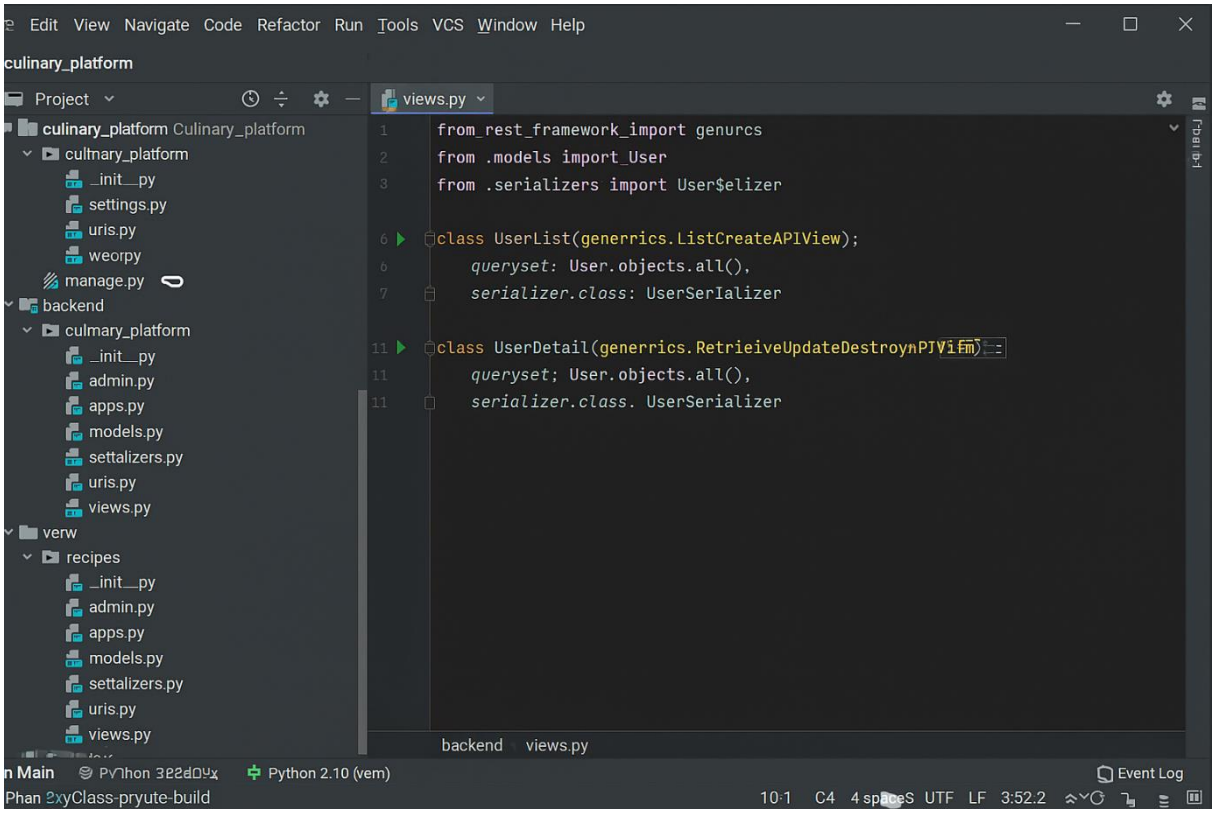

Рисунок 2.12 – Візуальний інтерфейс IDE PyCharm з проектом системи

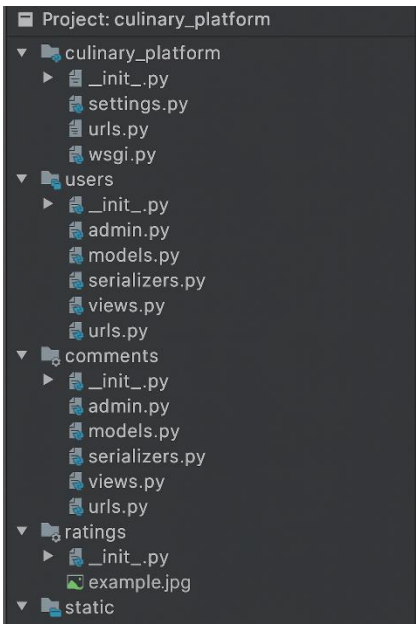

Рисунок 2.13 – дерево файлів проекту в IDE PyCharm.

Усі компоненти було організовано у логічну структуру каталогів, що чітко відображається в лівій частині вікна PyCharm у вигляді дерева проєкту. Це забезпечувало зручність навігації, швидкий доступ до потрібних файлів та полегшувало тестування. Після налаштування віртуального середовища та встановлення всіх необхідних бібліотек (зокрема djangorestframework, mysqlclient, python-dotenv), розробка серверної логіки виконувалась у стабільному та контрольованому середовищі.

Під час роботи було реалізовано REST API, який забезпечував доступ до даних системи через HTTP-запити, включаючи фільтрацію, сортування, автентифікацію через Google OAuth 2.0, додавання та перегляд рецептів, коментарів, оцінок, а також авторизацію користувачів. Для взаємодії з базою даних використовувалась ORM Django, яка автоматично генерує SQL-запити відповідно до оголошених моделей.

Завдяки використанню PyCharm вдалось досягти високої продуктивності розробки, впорядкованості структури коду та зручності керування всіма етапами побудови серверної частини. Це середовище стало не лише інструментом написання коду, а повноцінною платформою управління всім життєвим циклом розробки бекенду системи.

2.5 Реалізація механізму автентифікації користувача системи

У рамках розробки інформаційно-дискусійної платформи кулінарних рецептів було обрано технологію OAuth 2.0 для реалізації механізму авторизації користувачів (див. рис. 2.14).

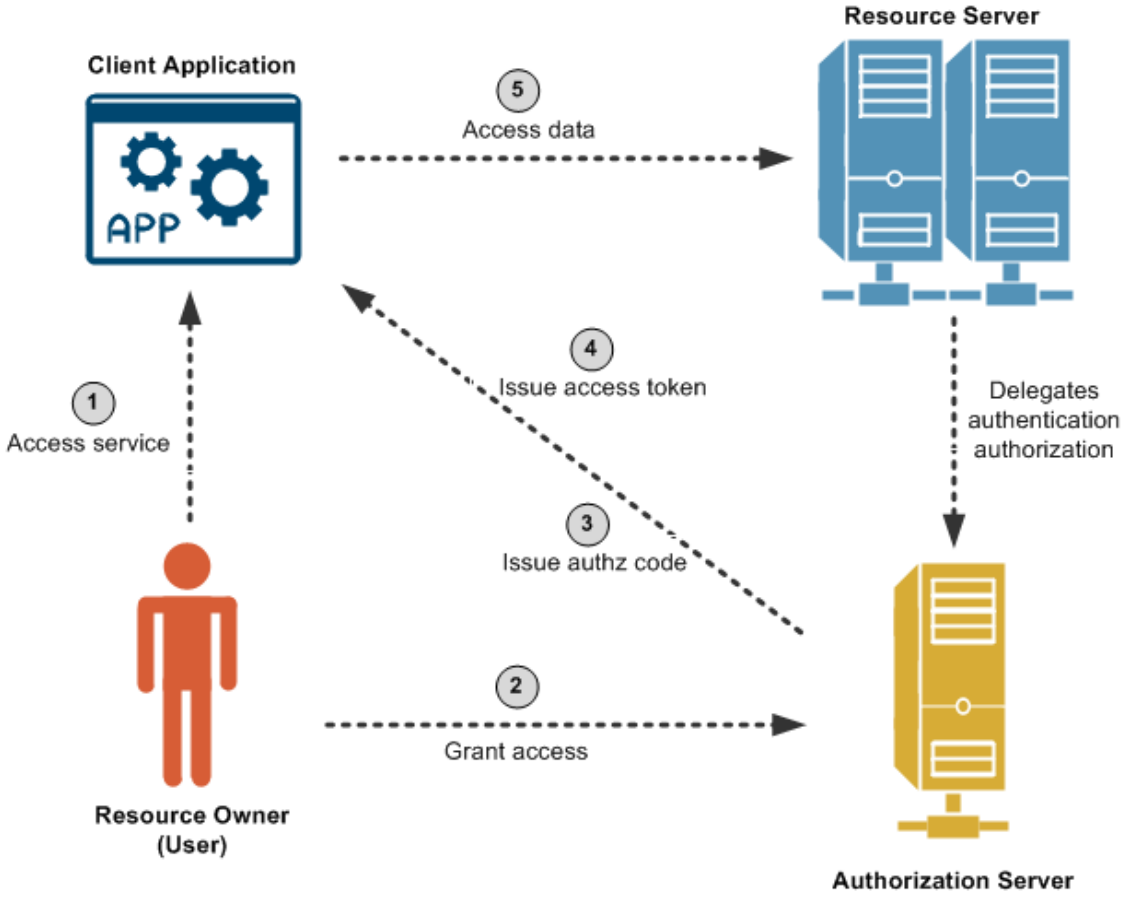

Рисунок 2.14 – Схема інформаційних потоків авторизації по протоколу OAuth 2.0

Цей вибір обумовлений рядом переваг, що дозволяють підвищити безпеку, зручність та масштабованість системи.

1. Підвищена безпека. OAuth 2.0 забезпечує безпечний спосіб входу без необхідності зберігати паролі в системі. Користувачі проходять авторизацію через зовнішніх постачальників ідентифікації (наприклад, Google або Facebook), що мінімізує ризики витоку конфіденційних даних. Таким чином,

система не має доступу до паролів користувача і не потребує реалізації механізмів їх захисту.

2. Зручність входу для користувачів. OAuth 2.0 дозволяє реалізувати вхід «в один клік». Це особливо важливо для платформ з широкою аудиторією, де користувачі очікують швидкий і безпроблемний доступ до функцій. Вхід через знайомі сервіси (Google, Facebook) викликає більше довіри та зменшує кількість відмов від реєстрації.

3. Можливість інтеграції з соціальними мережами. Платформа отримує доступ до базової інформації профілю користувача, такої як ім’я, аватар, посилання на сторінку. Це дозволяє формувати персоналізовані профілі, а також реалізовувати функції переходу на профіль автора рецепта чи написання повідомлення у соцмережі.

4. Масштабованість і підтримка стандартів. OAuth 2.0 є міжнародним стандартом авторизації, підтримується більшістю хмарних та мобільних платформ. Це дозволяє легко інтегрувати кілька постачальників авторизації, не змінюючи логіку системи. У разі розвитку проекту в мобільному напрямку або для корпоративних користувачів, технологія залишатиметься актуальною.

5. Зменшення витрат на обслуговування. Використання OAuth 2.0 звільняє розробників від необхідності створювати і обслуговувати такі функції, як: відновлення паролю, зберігання хешованих паролів, система безпеки доступу. Це також спрощує відповідність вимогам законодавства щодо обробки персональних даних (наприклад, GDPR).

6. Довіра до платформи. Авторизація через відомі платформи сприяє формуванню довіри з боку нових користувачів. Відсутність потреби створювати новий обліковий запис підвищує ймовірність взаємодії з системою.

Блок схема алгоритму авторизації наведена на рисунку 2.15.

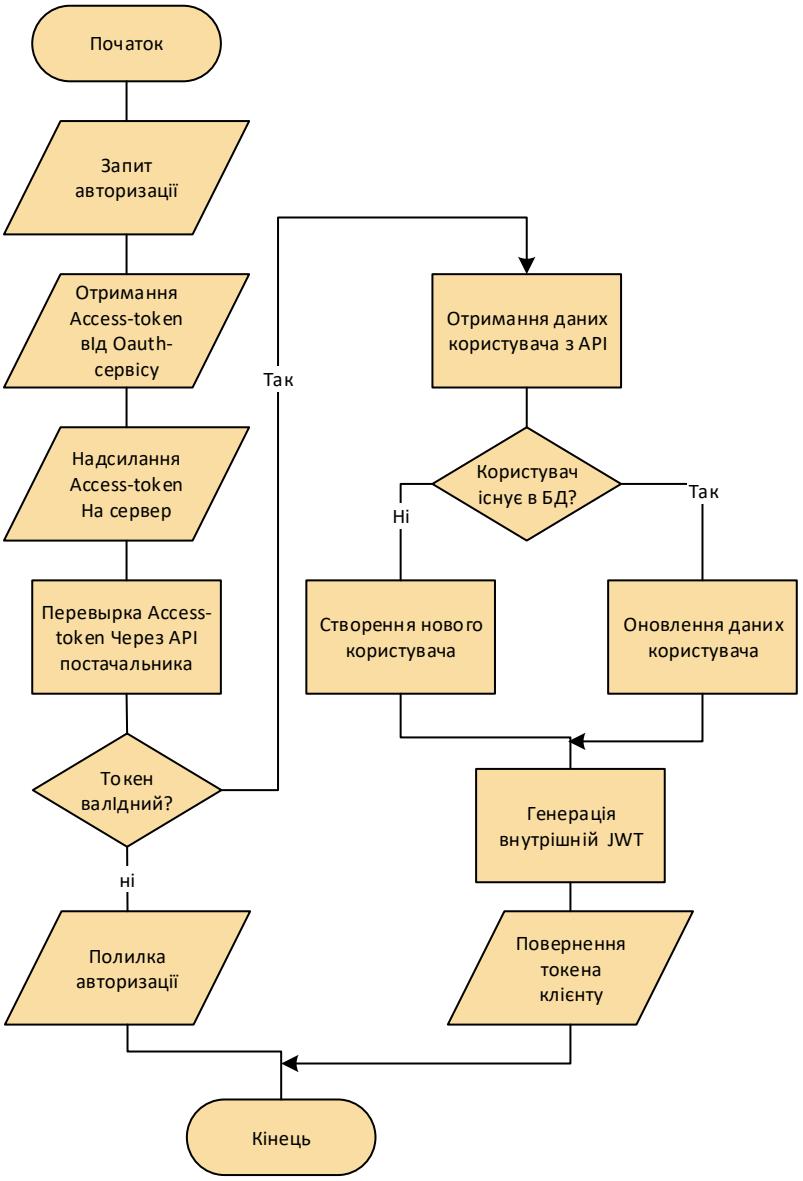

Рисунок 2.15 – Блок-схема алгоритму авторизації

Використання технології OAuth 2.0 у системі кулінарних рецептів забезпечує високий рівень безпеки, підвищує зручність для користувачів, а також відкриває можливості для масштабування і інтеграції з іншими сервісами. Це оптимальне рішення для авторизації у сучасному веб-застосунку.

Висновки до розділу

В розділі здійснено комплексне технічне обґрунтування, проєктування та реалізацію основних компонентів інформаційно-довідкової системи кулінарних рецептів, яка розроблялася в межах дипломного проєкту. Робота

охоплювала повний цикл інженерного створення програмного продукту: від вибору архітектури та технологічного стека до побудови інтерфейсу та налаштування механізмів безпечної авторизації користувачів.

На етапі вибору архітектури було обґрунтовано доцільність використання трирівневої клієнт-серверної моделі, що дозволяє ефективно розмежовувати логіку взаємодії з користувачем, обробку даних і доступ до бази даних. Така архітектура сприяє масштабованості системи та забезпечує можливість подальшого її розширення без суттєвих змін у структурі.

Проєктування бази даних виконувалося з урахуванням особливостей предметної області та функціональних вимог до системи. Була розроблена нормалізована модель даних, що включає сутності для збереження інформації про рецепти, інгредієнти, користувачів, категорії страв, оцінки та коментарі. На основі моделі побудовано ER-діаграму, яка чітко відображає логіку зв’язків між таблицями, що забезпечує цілісність і зручність доступу до даних у процесі експлуатації системи.

У рамках підрозділу розробки візуального інтерфейсу користувача створено адаптивні макети основних сторінок системи — головної сторінки, пошуку рецептів, профілю користувача, сторінки перегляду рецепту, авторизації та додавання нових матеріалів. Особливу увагу приділено адаптації інтерфейсу під мобільні пристрої та планшети, дотриманню принципів UX/UI-дизайну та забезпеченню логічної навігації між модулями.

Програмна реалізація системи здійснювалася з використанням сучасного інструментарію. Для серверної частини застосовано мову Python у зв’язці з фреймворком Django REST Framework, що забезпечує швидку побудову API, зручну інтеграцію з базою даних і високий рівень безпеки. На клієнтській стороні реалізовано верстку інтерфейсів із використанням HTML, CSS (Tailwind), JavaScript, а також бібліотек для інтерактивної взаємодії з бекендом. Структура проєкту, організована у вигляді Django-додатків, дозволяє легко масштабувати функціональність та обслуговувати код.

Важливою частиною розробки стало впровадження механізму автентифікації користувачів. Для цього використано протокол OAuth 2.0, який забезпечує безпечну авторизацію через зовнішні постачальники ідентифікації — Google, Facebook, Twitter. Такий підхід дозволяє уникнути прямої роботи з паролями, скорочує час реєстрації та підвищує довіру користувачів до системи. Була реалізована повна схема авторизації, обробки токенів, створення або оновлення облікових записів, що інтегрується у загальну архітектуру сервісу.

Таким чином, у другому розділі дипломного проєкту послідовно реалізовано всі ключові етапи проєктування та створення повнофункціональної інформаційно-довідкової системи. Вибір інструментів, архітектурних рішень і реалізаційних підходів є обґрунтованим і відповідає сучасним вимогам до надійності, масштабованості та зручності веб-систем. Отриманий результат забезпечує готовність системи до подальшого тестування, розгортання та впровадження для реального використання.

3 ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

3.1 Тестування базового функціоналу системи та механізму авторизації користувачів

Після завершення основного етапу розробки системи було проведено її комплексне тестування. Мета тестування полягала у перевірці працездатності, коректності реалізації основного функціоналу, а також у забезпеченні якісного відображення інтерфейсу на різних пристроях із різними розмірами екрану. Для цього тестування проводилося в умовах, наближених до реального розгортання.

Першим етапом стало розгортання веб-застосунку на локальному сервері розробника. Використовувалося вбудоване середовище запуску Django (python manage.py runserver), що забезпечувало швидкий запуск та тестування без потреби у сторонньому хостингу. База даних MySQL була попередньо ініціалізована шляхом виконання міграцій Django, що забезпечило створення всіх необхідних таблиць згідно з моделями. Для збереження медіафайлів (зображень страв, фотографій профілю користувачів) було налаштовано локальне сховище в папці media/, і доступ до нього забезпечено через URL-шлях, прописаний у файлі settings.py.

Після запуску системи було виконано функціональне тестування REST API. Основна увага приділялась коректності відповідей на HTTP-запити до таких ендпоінтів, як /api/recipes/, /api/comments/, /api/users/, /api/ratings/. Для цього використовувався інструмент Postman, за допомогою якого надсилались запити на створення, отримання, оновлення і видалення об’єктів. Перевірялися також реакції API на некоректні дані, відсутність токена авторизації, порушення прав доступу тощо. Усі відповіді перевірялись на відповідність форматам JSON, а також на відповідність очікуваній структурі, валідації і статус-кодам (200, 201, 400, 401, 403, 404).

Окремим блоком тестування було забезпечено перевірку коректності реалізації автентифікації через OAuth 2.0. (див. рис. 3.1) Для цього

використовувався реальний Google API-токен, згенерований через Google Identity Platform. Було протестовано повний цикл входу користувача: отримання токена на фронтенді, надсилання токена на бекенд, отримання профілю користувача, створення облікового запису та повернення авторизованої сесії. У разі невалідного токена система коректно повертала повідомлення про помилку авторизації.

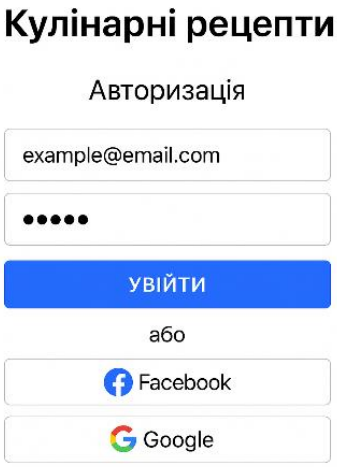

Рисунок 3.1 – Тестування роботи протоколу OAuth 2.0.

Проведене тестування підтвердило коректність і надійність функціонування розробленої системи в умовах, наближених до її реального використання. Усі ключові компоненти — від взаємодії з базою даних до логіки REST API та системи авторизації — пройшли перевірку на відповідність функціональним вимогам. Особливу увагу було приділено перевірці стійкості до некоректних запитів, обробці виключних ситуацій, коректності валідації вхідних даних, дотриманню політик доступу та надійності обробки авторизації через сторонні сервіси. Отримані результати свідчать про високий рівень стабільності та функціональності системи, що дає підстави для її подальшого розгортання на продуктивному сервері та використання в реальному середовищі з реальними користувачами. У процесі

тестування не виявлено критичних помилок, що підтверджує готовність системи до впровадження та відповідність поставленим технічним вимогам.

3.2 Тестування адаптивного інтерфейсу

Наступним етап тестування полягало в перевірці адаптивності інтерфейсу. Метою цього етапу було забезпечити, щоб сторінки інтерфейсу системи коректо відображались на різних типах пристроїв: мобільних телефонах, планшетах, десктопах (див. рис. 3.2). Для перевірки використовувався браузер Google Chrome у режимі інструментів розробника (DevTools → Responsive mode). У цьому режимі тестувалися шаблони головної сторінки, сторінки пошуку (див. рис. 3.3), форми входу, перегляду рецепта та додавання коментарів (див. рис. 3.4), створення/редагування рецепту (див. рис. 3.7), налаштування користувача системи (див. рис. 3.8). Усі блоки, такі як поля пошуку, кнопки, рейтинги, зображення та текстові поля, змінювали свої розміри та положення відповідно до ширини вікна браузера (див. рисунок 3.2).

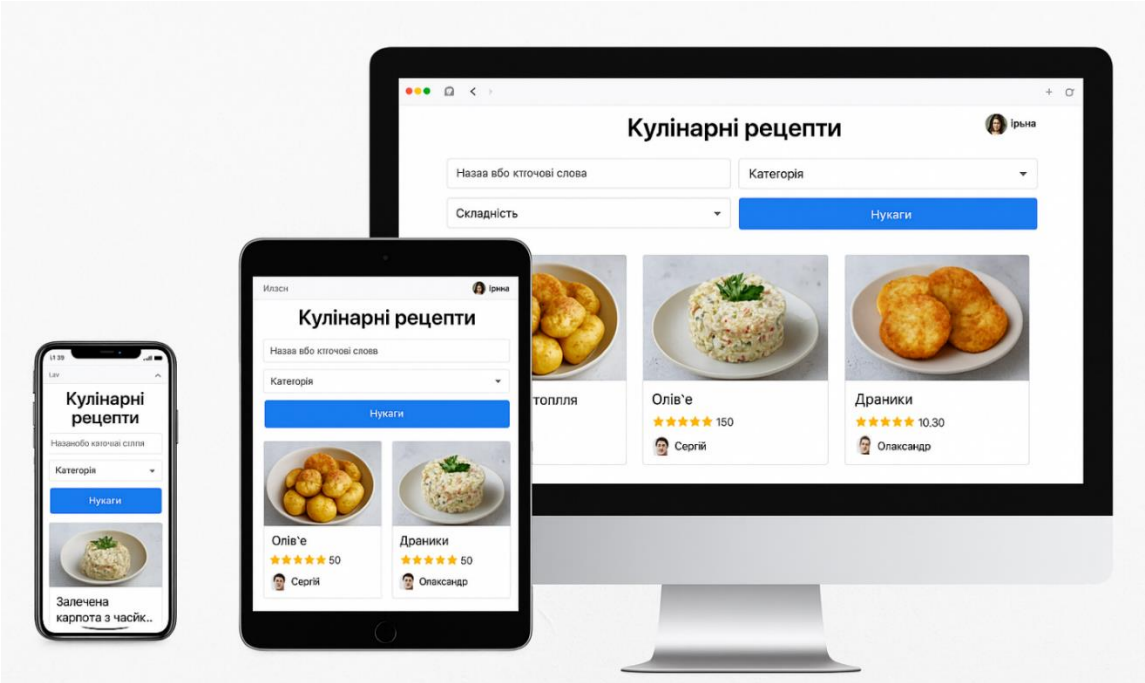

Рисунок 3.2 – Відображення адаптивних сторінок системи

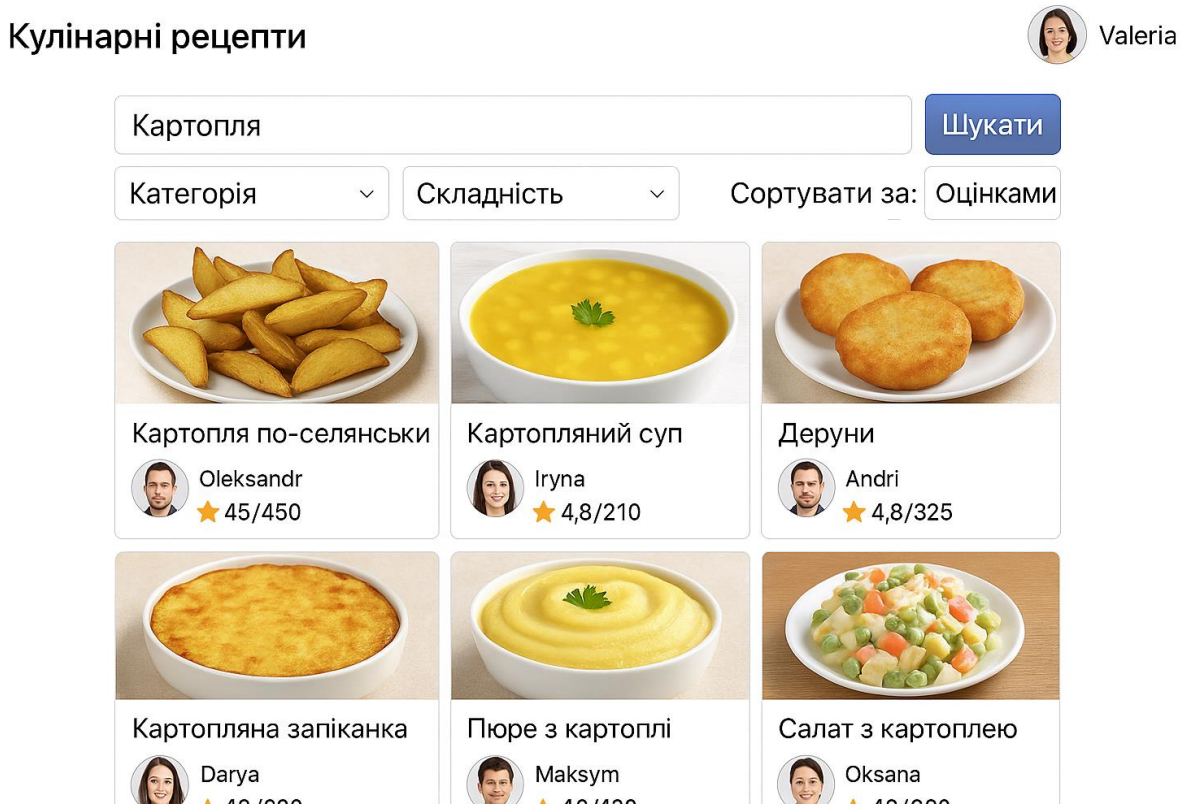

Рисунок 3.3 – Веб-сторінка пошуку

Сторінки успішно адаптувалися до різних роздільних здатностей, без горизонтальної прокрутки та з дотриманням відступів і логічної структури відображення.

Факт проведення тестування було задокументовано скріншотами основних сторінок веб-застосунку у трьох адаптивних форматах: смартфон (360x640), планшет (768x1024) та ПК (1920x1080). Зображення було отримано як для неавторизованого користувача, так і після входу через Google. Вони ілюструють коректне функціонування інтерфейсу та підтверджують наявність персоналізації (фото та ім’я користувача у шапці сайту), динамічне оновлення даних та успішне завантаження зображень страв.

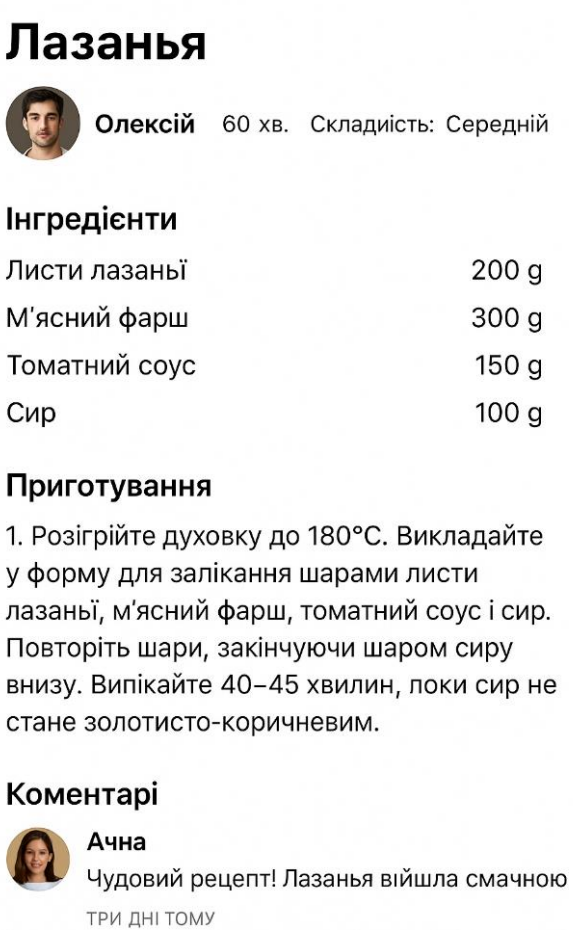

Рисунок 3.4 – Веб-сторінка перегляду рецепту на мобільному пристрої

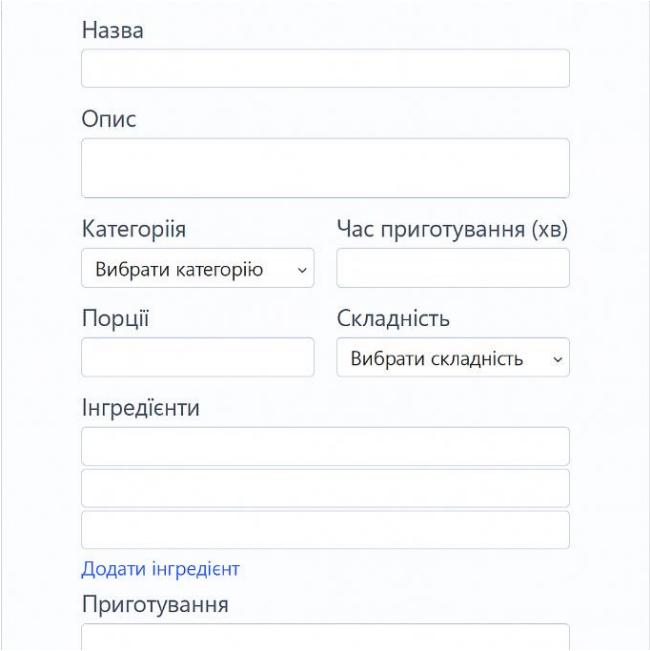

Рисунок 3.5 – Веб-сторінка створення рецепту на екрані планшету

Рис. 3.6 – Веб-сторінка створення рецепту на екрані планшету

У результаті тестування було підтверджено стабільну роботу ключових модулів, високу якість адаптивного інтерфейсу та коректну взаємодію з API авторизації. Зібрані матеріали засвідчують, що програмна система відповідає заявленим функціональним вимогам і готова до публічного використання.

Висновки до розділу

У ході тестування було здійснено всебічну перевірку працездатності всіх модулів системи, зокрема авторизації, створення та редагування рецептів, фільтрації за інгредієнтами, перегляду повної інформації про рецепт, динамічного підрахунку інгредієнтів відповідно до обраної кількості порцій, а також функцій оцінювання, коментування і переходу до соціального профілю автора.

Особливу увагу приділено коректності відображення веб-інтерфейсу на різних типах пристроїв, що перевірялося шляхом ручного тестування в сучасних браузерах на стаціонарному ПК, планшеті та смартфоні. Результати

підтвердили адаптивність макетів, відповідність інтерфейсних рішень принципам UX/UI, а також збереження повної функціональності незалежно від роздільної здатності екрана. Всі динамічні елементи коректно переміщуються або приховуються відповідно до розміру пристрою, а критично важливі функції (наприклад, кнопки пошуку, авторизації, публікації рецепту) завжди залишаються доступними для користувача.

Тестування модулів авторизації через соціальні мережі (OAuth 2.0) показало високу стабільність функціонування. Авторизовані користувачі отримували доступ до персонального профілю, могли додавати власні рецепти, редагувати особисту інформацію та залишати коментарі. Було перевірено правильність обробки помилок при авторизації, а також поведінку системи при спробі неавторизованого доступу до функцій, які потребують реєстрації.

Було виконано тестування базових сценаріїв взаємодії з системою — пошук за ключовими словами, категоріями та інгредієнтами. Результати показали, що реалізований алгоритм пошуку повертає релевантні результати, включаючи рецепти з частковим збігом ключових слів та суміжних категорій. Перевірка фільтрації за складністю та кількістю порцій також підтвердила коректну роботу механізмів обробки параметрів користувача на стороні сервера.

У процесі тестування були виявлені незначні візуальні зауваження, що були оперативно усунені. Загальна продуктивність системи знаходиться на прийнятному рівні: сторінки завантажуються швидко, інтерактивні елементи реагують без затримки, навігація є логічною і не викликає труднощів у користувача.

Також підтверджено збереження консистентності даних у базі, цілісність зв’язків між таблицями, коректність додавання, редагування та видалення записів, пов’язаних із користувачами, рецептами, коментарями та оцінками. Всі операції успішно виконуються з відповідною валідацією введених даних.

Загалом результати тестування свідчать про високий рівень реалізації функціональних вимог, що були визначені на етапі проектування. Система демонструє стабільну роботу, відповідає принципам адаптивності, має зручний та інтуїтивно зрозумілий інтерфейс, забезпечує достатній рівень безпеки та продуктивності.

Отже, на основі проведених тестувань можна зробити висновок, що розроблений веб-застосунок кулінарних рецептів є повністю працездатним, готовим до подальшого розгортання на хостинговій платформі та використання кінцевими користувачами. Усі ключові модулі системи функціонують згідно з поставленими завданнями, що підтверджує якість розробки та успішну реалізацію проекту в рамках дипломної роботи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено повнофункціональну інформаційно-дискусійну розподілену-систему для зберігання, пошуку, обговорення та додавання кулінарних рецептів. Весь процес розробки охоплював послідовні етапи аналізу, проектування, реалізації та тестування програмного продукту, що відповідає принципам інженерії програмного забезпечення.

На першому етапі було здійснено аналіз предметної області, вивчено потреби майбутніх користувачів, визначено основні функціональні та нефункціональні вимоги до системи. Зокрема, були сформульовані вимоги до реалізації авторизації через соціальні мережі, пошуку за інгредієнтами, адаптивного інтерфейсу для мобільних пристроїв, а також можливості коментування та оцінювання рецептів.

Наступним етапом стало проектування структури даних і архітектури системи. Розроблено концептуальну модель бази даних у вигляді ER-діаграми та створено SQL-структуру таблиць для зберігання інформації про користувачів, рецепти, інгредієнти, коментарі, рейтинги та категорії страв. Також було побудовано UML-діаграми переходів між сторінками, що забезпечили логічне структурування інтерфейсу користувача та зв’язків між основними елементами системи.

Під час розробки інтерфейсу користувача значну увагу було приділено адаптивності дизайну. Були створені шаблони (wireframes) та макети сторінок для трьох типів пристроїв: мобільних телефонів, планшетів і десктопів. Реалізовано інтерфейс у мінімалістичному стилі з використанням сучасних фреймворків верстки, зокрема Tailwind CSS. Ключові сторінки — пошуку, перегляду рецепта, додавання нового рецепта, авторизації та редагування профілю — були протестовані в реальному середовищі та адаптовані для різних роздільних здатностей екрану.

Серверна частина системи була реалізована за допомогою мови програмування Python та фреймворку Django у поєднанні з Django REST

Framework. Було створено REST API, який дозволяє працювати з рецептами, користувачами, оцінками та коментарями. Для зберігання даних використовувалася реляційна СКБД MySQL. Особливу увагу приділено безпеці — реалізовано авторизацію користувачів через протокол OAuth 2.0 із використанням Google та Facebook як постачальників ідентифікації. Також налаштовано збереження медіафайлів користувачів та страв у локальному сховищі з перспективою переходу на хмарне сховище.

Після завершення розробки було проведено комплексне тестування системи, яке включало перевірку функціональності REST API, коректності обробки запитів, безпеки авторизації та адаптивності інтерфейсу. Було задокументовано результати тестування за допомогою скріншотів і діаграм, що підтверджують працездатність системи в умовах, наближених до експлуатаційних.

Таким чином, розроблена система повністю відповідає поставленим вимогам кваліфікаційної роботи бакалавра. Вона забезпечує зручну платформу для обміну кулінарними знаннями, підтримує багатофункціональність і безпечну взаємодію з користувачем, має сучасний адаптивний дизайн та відповідає принципам масштабованої архітектури. Отриманий результат демонструє успішне застосування знань і навичок у сфері програмної інженерії та здатність реалізовувати повноцінні інформаційні системи від аналізу до впровадження.

ПЕРЕЛІК ПОСИЛАНЬ

1. ISO/IEC 25010:2011 — Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models;

2. Django Software Foundation. Django Documentation. [Електронний ресурс] — Режим доступу: https://docs.djangoproject.com/

3. Django REST Framework. [Електронний ресурс] — Режим доступу: https://www.django-rest-framework.org/

4. MySQL Reference Manual. Oracle. [Електронний ресурс] — Режим доступу: https://dev.mysql.com/doc/

5. OAuth 2.0 Authorization Framework. Internet Engineering Task Force (IETF). [Електронний ресурс] — https://datatracker.ietf.org/doc/html/rfc6749

6. Google Identity Platform. [Електронний ресурс] — Режим доступу: https://developers.google.com/identity

7. Facebook for Developers: Login. [Електронний ресурс] — Режим доступу: https://developers.facebook.com/docs/facebook-login/

8. W3C. HTML5 Specification. World Wide Web Consortium. [Електронний ресурс] — Режим доступу: https://html.spec.whatwg.org/

9. Mozilla Developer Network (MDN). CSS, HTML, JavaScript Docs. [Електронний ресурс] — https://developer.mozilla.org/

10. Tailwind Labs. Tailwind CSS Documentation. [Електронний ресурс] — https://tailwindcss.com/docs

11. Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.

12. Nielsen, J. (1999). Designing Web Usability: The Practice of Simplicity. New Riders Publishing.

13. Sommerville, I. (2016). Software Engineering (10th Edition). Pearson Education Limited.

14. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

15. Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). Head First Design Patterns. O'Reilly Media.

16. Martin, R.C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

17. Bąk, P. (2021). Modern API Development with Django and DRF. Packt Publishing.

18. Welling, L., & Thomson, L. (2005). PHP and MySQL Web Development. Addison Wesley.

19. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.

20. Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner’s Approach. McGraw-Hill Education.

21. Kruchten, P. (2004). The Rational Unified Process: An Introduction (3rd Edition). Addison-Wesley.

Додаток А – Код програми

```
# manage.py
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'culinary_platform.settings')
    try:
        from          django.core.management          import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't  import  Django.  Are  you  sure  it's
installed and "
            "available  on  your  PYTHONPATH  environment
variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

# culinary_platform/settings.py

import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY  =  os.getenv('DJANGO_SECRET_KEY',  'replace-this-
with-real-key')
DEBUG = True
ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'users',
    'recipes',
    'comments',
    'ratings',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
```

```
        'django.middleware.common.CommonMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
        'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
    ]

    ROOT_URLCONF = 'culinary_platform.urls'

    TEMPLATES = [
        {
            'BACKEND':
'django.template.backends.django.DjangoTemplates',
            'DIRS': [BASE_DIR / 'templates'],
            'APP_DIRS': True,
            'OPTIONS': {
                'context_processors': [
                    'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
                ],
            },
        },
    ]

    WSGI_APPLICATION = 'culinary_platform.wsgi.application'

    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': os.getenv('DB_NAME', 'culinary_db'),
            'USER': os.getenv('DB_USER', 'root'),
            'PASSWORD': os.getenv('DB_PASSWORD', ''),
            'HOST': os.getenv('DB_HOST', 'localhost'),
            'PORT': os.getenv('DB_PORT', '3306'),
        }
    }

    AUTH_PASSWORD_VALIDATORS = []

    LANGUAGE_CODE = 'uk'
    TIME_ZONE = 'Europe/Kyiv'
    USE_I18N = True
    USE_L10N = True
    USE_TZ = True

    STATIC_URL = '/static/'
```

```
STATICFILES_DIRS = [BASE_DIR / 'static']
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# culinary_platform/urls.py

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/users/', include('users.urls')),
    path('api/recipes/', include('recipes.urls')),
    path('api/comments/', include('comments.urls')),
    path('api/ratings/', include('ratings.urls')),
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)

# users/models.py

from django.db import models

class User(models.Model):
    SOCIAL_TYPES = [
        ('google', 'Google'),
        ('facebook', 'Facebook'),
        ('twitter', 'Twitter'),
    ]
    name = models.CharField(max_length=100)
    email = models.EmailField(unique=True, null=True,
blank=True)
    social_id = models.CharField(max_length=100,
unique=True)
    social_type = models.CharField(max_length=10,
choices=SOCIAL_TYPES)
    avatar_url = models.URLField(null=True, blank=True)
    profile_url = models.URLField(null=True, blank=True)
    is_blocked = models.BooleanField(default=False)
    date_joined = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return f"{self.name} ({self.social_type})"
    class Meta:
        db_table = 'users'

# users/serializers.py

from rest_framework import serializers
from .models import User
```

```
    class UserSerializer(serializers.ModelSerializer):
        class Meta:
            model = User
            fields                                                  =
['id','name','email','social_id','social_type','avatar_url','pro
file_url','is_blocked','date_joined']
            read_only_fields = ['id','date_joined']

    # users/views.py

    import requests
    from django.conf import settings
    from rest_framework.views import APIView
    from rest_framework.response import Response
    from rest_framework import status, permissions, generics
    from .models import User
    from .serializers import UserSerializer

    class GoogleAuthView(APIView):
        permission_classes = [permissions.AllowAny]
        def post(self, request):
            token = request.data.get('token')
            if not token:
                return   Response({'error':'Token   required'},
status=400)
            user_info_url                                           =
'https://www.googleapis.com/oauth2/v2/userinfo'
            r           =           requests.get(user_info_url,
headers={'Authorization':f'Bearer {token}'})
            if r.status_code != 200:
                return   Response({'error':'Invalid   token'},
status=401)
            data = r.json()
            user, created = User.objects.get_or_create(
                social_id=data['id'],social_type='google',

defaults={'name':data.get('name'),'email':data.get('email'),'ava
tar_url':data.get('picture')}
            )
            serializer = UserSerializer(user)
            msg = 'Created' if created else 'Authenticated'
            return
Response({'message':msg,'user':serializer.data})

    class UserListView(generics.ListAPIView):
        queryset = User.objects.all()
        serializer_class = UserSerializer
        permission_classes = [permissions.IsAdminUser]

    class UserDetailView(generics.RetrieveAPIView):
        queryset = User.objects.all()
        serializer_class = UserSerializer
        permission_classes = [permissions.IsAuthenticated]
```

```
    # users/urls.py

    from django.urls import path
    from .views import GoogleAuthView, UserListView,
UserDetailView

    urlpatterns = [
        path('auth/google/', GoogleAuthView.as_view(),
name='google-auth'),
        path('', UserListView.as_view(), name='user-list'),
        path('<int:pk>/', UserDetailView.as_view(), name='user-
detail'),
    ]

    # recipes/models.py

    from django.db import models
    from users.models import User

    class Category(models.Model):
        name = models.CharField(max_length=100)
        def __str__(self): return self.name

    class Recipe(models.Model):
        title = models.CharField(max_length=255)
        description = models.TextField(blank=True)
        steps = models.TextField()
        default_portions = models.IntegerField(default=1)
        prep_time = models.IntegerField()
        difficulty = models.CharField(max_length=10,
choices=[('легка','легка'),('середня','середня'),('складна','скл
адна')])
        user =
models.ForeignKey(User,null=True,on_delete=models.SET_NULL)
        category =
models.ForeignKey(Category,null=True,on_delete=models.SET_NULL)
        views = models.IntegerField(default=0)
        created_at = models.DateTimeField(auto_now_add=True)
        def __str__(self): return self.title

    class Ingredient(models.Model):
        name = models.CharField(max_length=100)
        def __str__(self): return self.name

    class RecipeIngredient(models.Model):
        recipe =
models.ForeignKey(Recipe,on_delete=models.CASCADE)
        ingredient =
models.ForeignKey(Ingredient,on_delete=models.CASCADE)
        amount =
models.DecimalField(max_digits=6,decimal_places=2)
```

```
        unit                                                =
models.CharField(max_length=5,choices=[('г','г'),('мл','мл'),('ш
т','шт')])
        per_portion = models.BooleanField(default=True)

    # recipes/serializers.py

    from rest_framework import serializers
    from                    .models                    import
Recipe,Category,Ingredient,RecipeIngredient

    class IngredientSerializer(serializers.ModelSerializer):
        class Meta: model = Ingredient; fields=['id','name']

    class
RecipeIngredientSerializer(serializers.ModelSerializer):
        ingredient = IngredientSerializer()
        class     Meta:     model     =     RecipeIngredient;
fields=['ingredient','amount','unit']

    class RecipeSerializer(serializers.ModelSerializer):
        ingredients                                         =
RecipeIngredientSerializer(source='recipeingredient_set',
many=True)
        class Meta:
            model = Recipe
            fields                                          =
['id','title','description','steps','default_portions','prep_tim
e','difficulty','user','category','ingredients','views','created
_at']

    # recipes/views.py

    from rest_framework import generics, permissions, filters
    from .models import Recipe,Category
    from .serializers import RecipeSerializer

    class RecipeListView(generics.ListCreateAPIView):
        queryset = Recipe.objects.all()
        serializer_class = RecipeSerializer
        permission_classes                                  =
[permissions.IsAuthenticatedOrReadOnly]
        filter_backends                                     =
[filters.SearchFilter,filters.OrderingFilter]
        search_fields                                       =
['title','description','recipeingredient__ingredient__name']
        ordering_fields = ['created_at','views']
        def          perform_create(self,          serializer):
serializer.save(user=self.request.user)

    class
RecipeDetailView(generics.RetrieveUpdateDestroyAPIView):
        queryset = Recipe.objects.all()
```

```
        serializer_class = RecipeSerializer
        permission_classes =
[permissions.IsAuthenticatedOrReadOnly]

    # recipes/urls.py

    from django.urls import path
    from .views import RecipeListView, RecipeDetailView

    urlpatterns = [
        path('', RecipeListView.as_view(), name='recipe-list'),
        path('<int:pk>/',           RecipeDetailView.as_view(),
name='recipe-detail'),
    ]

    # comments/models.py

    from django.db import models
    from recipes.models import Recipe
    from users.models import User

    class Comment(models.Model):
        recipe =
models.ForeignKey(Recipe,on_delete=models.CASCADE)
        user =
models.ForeignKey(User,null=True,on_delete=models.SET_NULL)
        content = models.TextField()
        image_url = models.URLField(blank=True)
        created_at = models.DateTimeField(auto_now_add=True)

    # comments/serializers.py

    from rest_framework import serializers
    from .models import Comment

    class CommentSerializer(serializers.ModelSerializer):
        class Meta: model = Comment; fields='__all__'

    # comments/views.py

    from rest_framework import generics, permissions
    from .models import Comment
    from .serializers import CommentSerializer

    class CommentListView(generics.ListCreateAPIView):
        queryset = Comment.objects.all()
        serializer_class = CommentSerializer
        permission_classes =
[permissions.IsAuthenticatedOrReadOnly]
        def          perform_create(self,          serializer):
serializer.save(user=self.request.user)

    # comments/urls.py
```

```
    from django.urls import path
    from .views import CommentListView
    urlpatterns    =    [path('',    CommentListView.as_view(),
name='comment-list')]

    # ratings/models.py

    from django.db import models
    from recipes.models import Recipe
    from users.models import User

    class Rating(models.Model):
        recipe                                                  =
models.ForeignKey(Recipe,on_delete=models.CASCADE)
        user = models.ForeignKey(User,on_delete=models.CASCADE)
        value                                                   =
models.DecimalField(max_digits=2,decimal_places=1)

    # ratings/serializers.py

    from rest_framework import serializers
    from .models import Rating

    class RatingSerializer(serializers.ModelSerializer):
        class Meta: model = Rating; fields='__all__'

    # ratings/views.py

    from rest_framework import generics, permissions
    from .models import Rating
    from .serializers import RatingSerializer

    class RatingListView(generics.ListCreateAPIView):
        queryset = Rating.objects.all()
        serializer_class = RatingSerializer
        permission_classes                                      =
[permissions.IsAuthenticatedOrReadOnly]

    # ratings/urls.py

    from django.urls import path
    from .views import RatingListView
    urlpatterns    =    [path('',    RatingListView.as_view(),
name='rating-list')]


    <!-- templates/base.html -->
    <!DOCTYPE html>
    <html lang="uk">
    <head>
      <meta charset="UTF-8">
```

```
      <meta     name="viewport"     content="width=device-width,
initial-scale=1.0">
      <title>{%  block  title  %}Кулінарні  рецепти{%  endblock
%}</title>
      <link href="/static/css/tailwind.css" rel="stylesheet">
    </head>
    <body class="bg-gray-100 font-sans">
      <header class="bg-white shadow p-4">
        <div class="container mx-auto flex justify-between items-
center">
          <a   href="/"   class="text-2xl   font-bold">Кулінарні
рецепти</a>
          <nav>
            {% if user.is_authenticated %}
              <span class="mr-4">{{ user.name }}</span>
              <a href="/logout/" class="text-blue-600">Вихід</a>
            {% else %}
              <a href="/login/" class="text-blue-600">Увійти</a>
            {% endif %}
          </nav>
        </div>
      </header>
      <main class="container mx-auto p-4">
        {% block content %}{% endblock %}
      </main>
      <footer class="bg-white shadow p-4 text-center text-gray-
600">
        © 2025 Кулінарні рецепти
      </footer>
    </body>
    </html>

    <!-- templates/recipe_list.html -->
    {% extends 'base.html' %}
    {% block title %}Пошук рецептів{% endblock %}
    {% block content %}
      <h1 class="text-xl mb-4">Результати пошуку</h1>
      <div class="grid grid-cols-1 md:grid-cols-3 gap-4">
        {% for recipe in recipes %}
        <div class="bg-white shadow rounded p-4">
          <img  src="{{  recipe.image_url  }}"  class="w-full  h-32
object-cover rounded mb-2">
          <h2 class="font-bold text-lg">{{ recipe.title }}</h2>
          <p  class="text-sm  text-gray-600">{{  recipe.user.name
}}</p>
          <a href="/recipes/{{  recipe.id  }}/" class="text-blue-
600 mt-2 inline-block">Детальніше</a>
        </div>
        {% endfor %}
      </div>
    {% endblock %}

    <!-- templates/recipe_detail.html -->
```

```
    {% extends 'base.html' %}
    {% block title %}{{ recipe.title }}{% endblock %}
    {% block content %}
      <div class="bg-white shadow rounded p-6">
        <h1  class="text-2xl  font-bold  mb-4">{{  recipe.title
}}</h1>
        <img  src="{{  recipe.image_url  }}"  class="w-full  h-64
object-cover rounded mb-4">
        <div class="mb-4">
          <span class="font-semibold">Порції:</span> {{ portions
}}
          <span class="font-semibold ml-4">Складність:</span> {{
recipe.difficulty }}
        </div>
        <h2 class="font-semibold mb-2">Інгредієнти:</h2>
        <ul class="list-disc list-inside mb-4">
          {% for ing in ingredients %}
          <li>{{   ing.quantity   }}   {{   ing.unit   }}   {{
ing.ingredient.name }}</li>
          {% endfor %}
        </ul>
        <h2 class="font-semibold mb-2">Приготування:</h2>
        <ol class="list-decimal list-inside">
          {% for step in steps %}
          <li class="mb-2">{{ step }}</li>
          {% endfor %}
        </ol>
      </div>
    {% endblock %}

    <!-- templates/add_recipe.html -->
    {% extends 'base.html' %}
    {% block title %}Додати рецепт{% endblock %}
    {% block content %}
      <form     method="post"     enctype="multipart/form-data"
class="max-w-lg bg-white shadow rounded p-6 mx-auto">
        {% csrf_token %}
        <label class="block mb-2">Назва</label>
        <input type="text" name="title" class="w-full p-2 border
rounded mb-4">
        <label class="block mb-2">Опис</label>
        <textarea  name="description"  class="w-full  p-2  border
rounded mb-4"></textarea>
        <label class="block mb-2">Фото</label>
        <input type="file" name="image" class="mb-4">
        <label class="block mb-2">Інгредієнти</label>
        <textarea name="ingredients" placeholder="1 картопля
    2 яйця" class="w-full p-2 border rounded mb-4"></textarea>
        <label class="block mb-2">Кроки приготування</label>
        <textarea name="steps" class="w-full p-2 border rounded
mb-4"></textarea>
        <button type="submit" class="bg-blue-600 text-white px-4
py-2 rounded">Опублікувати</button>
```

```
      </form>
    {% endblock %}

    <!-- templates/login.html -->
    {% extends 'base.html' %}
    {% block title %}Увійти{% endblock %}
    {% block content %}
      <div class="max-w-sm mx-auto bg-white shadow rounded p-6">
        <h1 class="text-xl font-bold mb-4">Увійти через
соцмережі</h1>
        <a href="/api/users/auth/google/" class="block bg-red-
500 text-white text-center py-2 rounded mb-2">Google</a>
        <a href="/api/users/auth/facebook/" class="block bg-
blue-800 text-white text-center py-2 rounded mb-2">Facebook</a>
        <a href="/api/users/auth/twitter/" class="block bg-blue-
400 text-white text-center py-2 rounded">Twitter</a>
      </div>
    {% endblock %}

    <!-- templates/profile.html -->
    {% extends 'base.html' %}
    {% block title %}Профіль{% endblock %}
    {% block content %}
      <div class="max-w-md mx-auto bg-white shadow rounded p-6">
        <h1 class="text-2xl font-bold mb-4">{{ user.name }}</h1>
        <img src="{{ user.avatar_url }}" class="w-32 h-32
rounded-full mb-4">
        <p class="mb-2"><strong>Email:</strong> {{ user.email
}}</p>
        <p class="mb-4"><strong>Соцмережа:</strong> {{
user.social_type }}</p>
        <a href="/add-recipe/" class="bg-green-500 text-white px-
4 py-2 rounded">Додати рецепт</a>
      </div>
    {% endblock %}

    # users/views.py
    import requests
    from django.conf import settings
    from rest_framework.views import APIView
    from rest_framework.response import Response
    from rest_framework import status, permissions
    from .models import User
    from .serializers import UserSerializer

    class GoogleAuthView(APIView):
        """
        Обробник авторизації користувача через Google OAuth 2.0.
        Клієнт надсилає ID token, сервер перевіряє його
дійсність,
        отримує дані профілю та створює або оновлює запис
користувача.
        """
```

```
        permission_classes = [permissions.AllowAny]

        def post(self, request):
            token = request.data.get('token')
            if not token:
                return Response({'error': 'Token is required'},
status=status.HTTP_400_BAD_REQUEST)

            # Отримання даних користувача з Google API
            user_info_url                                    =
'https://www.googleapis.com/oauth2/v2/userinfo'
            headers = {'Authorization': f'Bearer {token}'}
            google_response    =    requests.get(user_info_url,
headers=headers)

            if google_response.status_code != 200:
                return   Response({'error':   'Invalid   token'},
status=status.HTTP_401_UNAUTHORIZED)

            data = google_response.json()
            social_id = data.get('id')
            name = data.get('name')
            email = data.get('email')
            avatar = data.get('picture')

            # Створення або отримання користувача в базі
            user, created = User.objects.get_or_create(
                social_id=social_id,
                social_type='google',
                defaults={
                    'name': name,
                    'email': email,
                    'avatar_url': avatar
                }
            )

            serializer = UserSerializer(user)
            message = 'Created' if created else 'Authenticated'
            return    Response({'message':    message,    'user':
serializer.data}, status=status.HTTP_200_OK)

    class FacebookAuthView(APIView):
        """
        Обробник авторизації користувача через Facebook OAuth
2.0.
        """
        permission_classes = [permissions.AllowAny]

        def post(self, request):
            token = request.data.get('token')
            if not token:
                return Response({'error': 'Token is required'},
status=status.HTTP_400_BAD_REQUEST)
```

```
            # Отримання даних користувача з Facebook API
            user_info_url                                          =
f'https://graph.facebook.com/me?fields=id,name,email,picture&acc
ess_token={token}'
            fb_response = requests.get(user_info_url)

            if fb_response.status_code != 200:
                return   Response({'error':   'Invalid   token'},
status=status.HTTP_401_UNAUTHORIZED)

            data = fb_response.json()
            social_id = data.get('id')
            name = data.get('name')
            email = data.get('email')
            avatar    =    data.get('picture',    {}).get('data',
{}).get('url')

            user, created = User.objects.get_or_create(
                social_id=social_id,
                social_type='facebook',
                defaults={
                    'name': name,
                    'email': email,
                    'avatar_url': avatar
                }
            )

            serializer = UserSerializer(user)
            message = 'Created' if created else 'Authenticated'
            return    Response({'message':    message,    'user':
serializer.data}, status=status.HTTP_200_OK)

    class TwitterAuthView(APIView):
        """
        Обробник авторизації користувача через Twitter OAuth 2.0.
        """
        permission_classes = [permissions.AllowAny]

        def post(self, request):
            #  Реалізація  авторизації  Twitter  залежить  від
отримання OAuth 1.0a токенів
            return   Response({'error':   'Twitter   OAuth   not
implemented'}, status=status.HTTP_501_NOT_IMPLEMENTED)
```