НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2026 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: “Вебзастосунок для обміну кулінарними рецептами”

Виконав: студент 4 курсу, групи ТР-21
Кобизький Максим Євгенійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор, д.т.н, професор Сергій ОТРОХ _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент професор каф. ІПЗЕ, д.т.н., професор Олексій НЕДАШКІВСЬКИЙ

_____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль ст.викладач Ольга БЕСПАЛА _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ - 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
(підпис)
“___” _____2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Кобизькому Максиму Євгенійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Вебзастосунок для обміну кулінарними рецептами”
Науковий керівник Отрох Сергій Іванович, д.т.н, професор
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “ 28 ” травня 2026 року № 1992 -с

2. Термін подання студентом роботи 08.06.2026

3. Вихідні дані до роботи мова програмування Python, фреймворк Django, СУБД SQLite, середовище розробки Visual Studio Code

4. Перелік питань, які потрібно розробити _____

1) провести аналіз сучасних вебплатформ та соціальних сервісів для обміну кулінарними рецептами

2) визначити основні функції вебзастосунку, що дозволяє створювати, шукати та обмінюватись рецептами

3) аргументувати вибрані інструменти, алгоритми та технології для реалізації вебсистеми

4) побудувати архітектуру програмного забезпечення та бази даних
5) реалізувати систему реєстрації та авторизації користувачів
6) реалізувати функціонал створення, редагування, видалення та перегляду кулінарних рецептів
7) реалізувати функціонал пошуку за назвами, хештегами, інгредієнтами та користувачами
8) реалізувати механізми взаємодії користувачів за допомогою системи підписок та вподобань
9) створити адаптивний вебінтерфейс користувача та реалізувати підтримку багатомовності
10) представити роботу розробленого вебзастосунку та описати основні сценарії використання

5. Орієнтований перелік ілюстративного матеріалу приклади реалізації основних систем. таблиць бази даних та прикладів роботи з вебзастосунком

6. Дата видачі завдання 15.09.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	15.09.2025	Виконано
2.	Аналіз методів та засобів розв’язання задачі	20.04.2026	Виконано
3.	Розробка архітектури та загальної структури системи	30.04.2026	Виконано
4.	Розробка окремих підсистем	01.05.2026	Виконано
5.	Програмна реалізація системи	04.05.2026	Виконано
6.	Оформлення пояснювальної записки	10.05.2026	Виконано
7.	Захист програмного забезпечення	13.05.2026	Виконано
8.	Передзахист	03.06.2026	Виконано
9.	Захист	15.06.2026	Виконано

Студент _____ Максим КОБИЗЬКИЙ
(підпис) (прізвище та ініціали)

Керівник _____ Сергій ОТРОХ
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

В наш час існує велика кількість соціальних мереж, які з часом впроваджуються новий трендовий функціонал. Найбільш популярним жанром є блог, де люди записують відео різного характеру або фотографують моменти зі свого життя, обмінюючись вони отримують як досвід, так і розвагу. Проте, окрім розваг важливим є саме досвід, а особливо в приготуванні корисної та смачної їжі.

Актуальність теми полягає у створенні саме української соціальної мережі для обміну та пошуку рецептів. На сьогоднішній стан існує декілька сайтів для публікації рецептів від якогось шефа, що є більше автобіографічним сайтом з власними досягненнями(рецептами). Основна перевага в тому, що це соціальна мережа, де буде можливість отримувати та розвивати свої навички в кулінарії разом з іншими. Дуже частою проблемою є важкість з пошуку рецепту з вже наявних продуктів. Адже іноді немає можливості докупити якийсь товар.

Метою роботи є розробка вебзастосунку для обміну кулінарними рецептами з використанням мови програмування Python та фреймворку Django.

Для успішного виконання цього завдання було проаналізовано предметну область, досліджено існуючі схожі сервіси, визначено засоби розробки для більшої зручності та правильної реалізації. У висновку маємо реалізований вебзастосунок з базовими можливостями соціальної мережі і зручною системою пошуку.

Анотація

Дипломна робота пов’язана зі створенням вебзастосунку для обміну кулінарними рецептами. З сучасним розвитком технологій та можливостей майже будь-хто має свій сайт чи блог. Але для зручного обміну рецептами є лише декілька конкурентно здатних платформ, однак без української локалізації.

У цій роботі було досліджено сучасні платформи для публікації рецептів, визначено їхні переваги та недоліки, а також на основі цього, сформовано вимоги до власної системи. Було спроєктовано архітектуру вебзастосуеку на основі Django фреймворку та розроблено структуру бази даних для зберігання інформації про користувачів, рецепти та теги.

Розроблений вебзастосунок має систему реєстрації та авторизації користувачів, ведення персональних профілів з можливістю їх кастомізації, публікацію та редагування рецептів, пошук за назвами, інгредієнтами, тегами і користувачами. Також, реалізовано механізми підписок, вподобань, адаптивний інтерфейс та підтримку багатомовності.

Для реалізації проєкту використано мову програмування Python, фреймворк Django, систему керування базами даних SQLite, а також технології HTML, CSS, Bootstrap і Jinja. Отриманий результат являє собою функціональний вебзастосунок, який може слугувати основою для подальшого розвитку та розширення можливостей платформи.

Ключові слова: вебзастосунок, кулінарні рецепти, Python, Django, SQLite, база даних, таблиця, веброзробка, пошук рецептів, соціальна мережа, локалізація.

Abstract

The bachelor's thesis is related to the development of a web application for sharing culinary recipes. With the modern development of technologies and opportunities, almost anyone can have their own website or blog. However, there are only a few competitive platforms for convenient recipe sharing, and they do not provide Ukrainian localization.

This work investigates modern platforms for publishing recipes, identifies their advantages and disadvantages, and, based on this analysis, formulates the requirements for the developed system. The architecture of the web application was designed using the Django framework, and a database structure was developed for storing information about users, recipes, and tags.

The developed web application includes a user registration and authentication system, personal profiles with customization capabilities, recipe publishing and editing functions, and search by titles, ingredients, tags, and users. In addition, subscription and like mechanisms, a responsive user interface, and multilingual support were implemented.

The project was developed using the Python programming language, the Django framework, the SQLite database management system, as well as HTML, CSS, Bootstrap, and Jinja technologies. The resulting product is a functional web application that can serve as a foundation for further development and expansion of the platform's capabilities.

Keywords: web application, culinary recipes, Python, Django, SQLite, database, table, web development, recipe search, social network, localization.

1 АНАЛІЗ АКТУАЛЬНИХ ВЕБПЛАТФОРМ ДЛЯ ОБМІНУ КУЛІНАРНИМИ РЕЦЕПТАМИ

У мережі інтернету найпоширенішим способом для обміну різного роду інформації слугують соціальні мережі. Вони популярні серед будь-якої вікової категорії та швидко розповсюджують будь-які дані.

1.1 Вебзастосунків для обміну кулінарними рецептами

Сервіси на тему кулінарії мають високий попит адже люди мають бажання їсти смачно, готувати швидко та не витрачати багато грошей і часу на відвідини ресторанів або на навчання методом спроб та помилок. Вебзастосунки дозволяють користуватись зручними функціями для створення публікацій у вигляді рецептів, шукати рецепти за назвою та за категоріями. Взаємодіяти між собою, за допомогою функцій підписки та вподобань.

Існуючі платформи мають базові функції та поєднують невелику частину інформаційної системи та соціальної мережі. Користувачі можуть створювати власні рецепти, заповнюючи необхідну інформацію, та взаємодіяти з іншими, підписуючись та оцінювати публікації. Також, створювати і оформлювати свої профілі. На основі певних факторів кожен користувач буде мати свою живу стрічку, де будуть розміщені можливо цікаві публікації.

Розробка вебзастосунків є дуже актуальною, адже майже кожен сервіс чи виробництво хоче мати свій зручний і логічний застосунок. У сфері кулінарії вебзастосунки є найзручнішим варіантом для швидкого обміну, взаємодії та пошуку.

1.2 Аналіз існуючих вебплатформ

Звичайно, перед розробкою було проаналізовано існуючи вебзастосунки та вебсервіси на тематику обміну рецептів. Було помічено тенденцію, що користувачі обирають зручний інтерфейс ніж тематичні вебзастосунки.

Першими конкурентами є соціальні мережі такі як, Instagram, Tiktok, Youtube. Їх люди обирають, бо вони дуже зручні та не мають зайвого і є майже універсальними. Вони вже мають свою систему підписок, хештегів та інтуїтивно простий інтерфейс. Деякі з цих функцій було досліджено та проаналізовано власне для власного вебзастосунку

Наступним сервісом-конкурентом буде Pinterest, що правда на ньому відсутня повноцінна взаємодія. Він має схожий функціонал та велику адаптивну стрічку постів.

Найголовнішими конкурентами будуть іноземні сервіси Cookpad, Allrecipes. Вони мають схожий задум і створені на тематику кулінарії та мають непоганий функціонал. Що правда вони не завжди добре оптимізовані та мають дуже велику кількість реклами через малу кількість відвідувань платформ. І важливе для нас це відсутність нормальної української локалізації.

1.3 Особливості предметної області

Проаналізувавши загальну інформацію про вебзастосунки та основних конкурентів, було визначено важливі функції та аспекти.

Найголовнішим є інтуїтивна простота, адже користувачі можуть довго адаптовуватись і звикати, що може значно їх відштовхнути від використання застосунку. Тому було обрано важливий і простий функціонал, а саме: взаємодія користувачів, профілі, публікації, стрічка, система підписок та вподобайок, пошуки за назвами, хештегами, продуктами та користувачами (таблиця 1.1).

Таблиця 1.1 — Призначення таблиць бази даних

Сервіс	Пошук	Профілі	Локалізація	Підписки
Cookpad	+	+	-	+
Allrecipes	+	+	-	-
Pinterest	Частково	+	+	+
Instagram	Частково	+	+	+
Розроблений продукт	+	+	+	+

Як бачимо з таблиці, деякі сервіси мають потрібний функціонал, але якщо брати Pinterest та Instagram то вони є більше соціальними мережами, які містять не тільки рецепти, а й власні блоги людей або якийсь не потрібний функціонал для нашої задумки. Отже, було досліджено їхні переваги та взято в урахування при розробці.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ОБМІНУ КУЛІНАРНИМИ РЕЦЕПТАМИ

У наш час такої української і зручної системи, аналогічної до розробленої, немає. Але, схожий результат можна отримати, якщо використовувати додаткові розширення для браузерів і перекладачі.

2.1 Задача розробки вебзастосунку

Метою розробки є створення вебзастосунку для обміну кулінарними рецептами. Вебзастосунок буде мати функції, які дозволять користувачам публікувати власні рецепти, оформлювати свій профіль, взаємодіяти з іншими користувачами та виконувати швидкі пошуки, за тегами та інгредієнтами.

Отже, для отримання такого результату необхідно реалізувати відповідні задачі з реалізації систем та інтерфейсів.

2.2 Архітектура вебзастосунку

Для реалізації використовувався фреймворк Django. Він використовує архітектуру MVT (model-view-template). Архітектура системи складається з: моделей, представлень, HTML-шаблонів, бази даних та системи маршрутизації (рисунок 2.1).

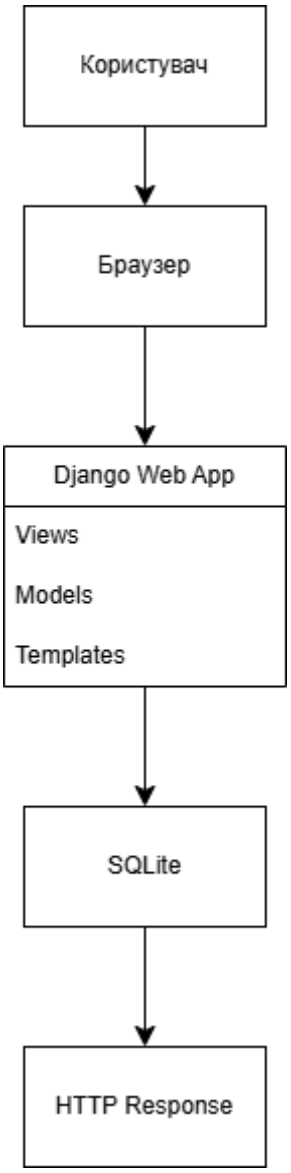

Рисунок 2.1 — Схема архітектури вебзастосунку

2.2.1 Моделі Django

Моделі – це опис бази даних та роботи з ними, за допомогою ORM Django. У проєкті використовуються моделі для профілю користувачів (Profile), для публікацій (Post). Для взаємодії користувачів між собою використовуються моделі Followers та LikePost. Для швидкого і зручного пошуку Tag та Ingredient.

Моделі по суті це таблиці бази даних, вони полегшують роботу з SQL-запитами і створюють таблиці автоматично.

2.2.2 Представлення Views

У представленнях знаходиться весь функціонал та програмна логіка. Було реалізовано представлення для домашньої сторінки чи стрічки, де відображаються пости профілів, на які підписаний користувач. Далі, було описано представлення для реєстрації, авторизації та перегляду і налаштування профілів. Для перегляду, створення, редагування реалізовано відповідні представлення. Була описана логіка з тегами для профілів, де якщо профіль публікує 5 рецептів з один і тим самим тегом чи тегами то на акаунт присвоюється цей тег. Нарешті, для пошуку було реалізовано представлення, яке передбачає пошук профілів, рецептів, тегів та інгредієнтів.

2.2.3 HTML-шаблони

HTML шаблони використовуються для формування інтерфейсу користувача. За допомогою шаблонної системи Django реалізовано динамічне відображення даних, отриманих із бази даних, зокрема рецептів, профілів користувачів та результатів пошуку. Також, за допомогою Jinja, підтримуються використання змінних, циклів та умовних конструкцій для гнучкого формування вмісту сторінок

Шаблони ще використуються для роботи зі статичними файлами та реалізації багатомовності інтерфейсу. Це дозволяє відокремити логіку застосунку від його зовнішнього вигляду. Такий підхід спрощує підтримку та можливість подальшого росту.

2.2.4 Маршрутизація URL

Маршрутизація URL використовується для обробки HTTP-запитів користувачів та зв’язування URL-адрес із відповідними представленнями Django. Вона забезпечує перехід між сторінками та враховує мову, яку встановив користувач у вебзастосунку.

Маршрути прописуються у файлі urls.py, де кожній адресі відповідає певне представлення. Завдяки цьому ми можемо централізовано керувати навігацією застосунку.

2.3 Структура вебзастосунку

Система програми складається з декількох модулів, які працюють між собою. Кожен з них відповідає за реалізацію певної частини функціоналу вебзастосунку. Такий підхід забезпечує зручністю розробки, підтримки та подальшого розширення програмного забезпечення. В Django фреймворку є багато вбудованих та корисних функціоналів, які полегшують реалізацію та роботу вебзастосунку.

2.3.1 Модуль авторизації користувачів

За допомогою вбудованої системи Django Authentication System, реалізований модуль авторизації користувачів. Головною задачею модулю є забезпечення безпечного доступу користувачів до вебзастосунку. Модуль надає можливість реєстрації нових користувачів, входу до системи, за допомогою логіна та пароля, а також виходу з облікового запису. Права доступу користувачів до сторінок та функцій системи перевіряє механізм автентифікації Django.

2.3.2 Модуль авторизації користувачів

Модуль профілів користувачів зберігає та керує інформацією про зареєстрованих користувачів. Після створення облікового запису для кожного користувача формується персональний профіль. Користувач може редагувати особисті дані, додати біографію, локацію, змінювати фотографію профілю та

змінювати колір сторінки. Також цей модуль відображає статистику користувача, його рецептів.

2.3.3 Модуль рецептів

Центральним компонентом вебзастосунку є модуль рецептів. Рецепти оформленні як публікації користувачів. Користувач може створювати, редагувати, переглядати, вподобати, перейти на детальну сторінку рецепту та видаляти публікацію. Кожна публікація має свою назву, фотографію страви, опис процесу приготування, набір тегів, інгредієнти, автора, час приготування, дату створення та кількість вподобань. Усі дані зберігаються у базі даних та використовуються при необхідності іншими модулями.

2.3.4 Модуль пошуку

Модуль пошуку призначений для швидкого знаходження потрібної інформації в системі. Для обробки пошукових запитів використовується можливості ORM Django. Він забезпечує ефективну взаємодію з базою даних та швидке отримання результатів навіть при збільшенні кількості записів. Реалізований модуль може виконувати пошук за списком інгредієнтів, тегами, назвами та для пошуку користувачів.

2.3.5 Модуль взаємодії користувачів

Соціальні можливості вебзастосунку реалізовані у модулі взаємодії користувачів. До функцій цього модулю належить система підписок на інших користувачів та механізмом вподобань для оцінювання рецептів. Завдяки цим функціям користувачі можуть стежити за новими та цікавими їм публікаціям від інших користувачів. Підписки та вподобання допомагають формувати домашню стрічку користувача та підвищення активності користувачів у системі.

2.4 Проєктування бази даних

Для проєктування структури бази даних було побудовано ER-діаграму, яка відображає основні сутності системи та зв’язки між ними (рисунок 2.2).

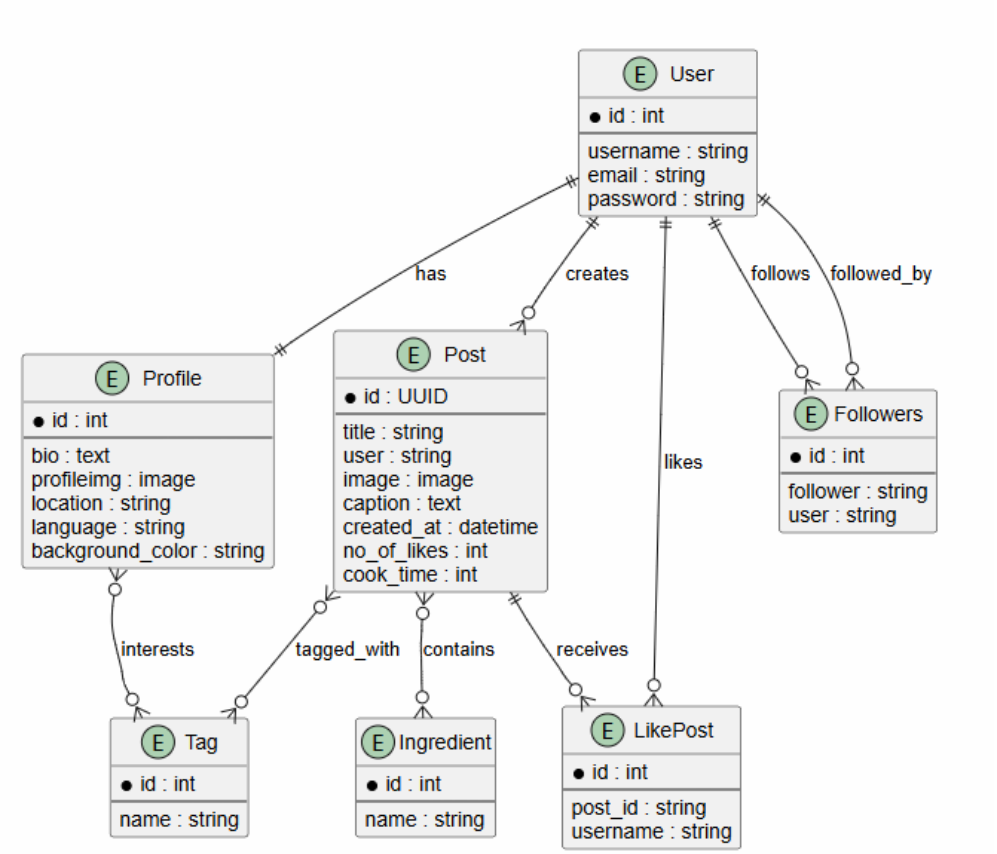

Рисунок 2.2 — ER-діаграма бази даних вебзастосунку

Центральними є Profile та Post. Сутність Profile зберігає інформацію про користувачів, зокрема опис профілю, зображення, місцезнаходження, мовні налаштування. Сутність Post містить інформацію про опубліковані рецепти, включаючи назву, зображення, опис, дату створення, кількість вподобань та час приготування.

Між сутностями Profile та User встановлено зв’язок типу «один до одного», що забезпечує наявність лише одного профілю для кожного користувача. Сутність Post пов’язана з користувачем зв’язком типу «один до багатьох», оскільки один користувач може створювати декілька рецептів.

Для класифікації, структурування та пошуку рецептів використовуються сутності Tag та Ingredient. Один рецепт може містити декілька тегів і декілька інгредієнтів, а кожен тег або інгредієнт може використовуватися в багатьох рецептах. Тому між сутностями Post і Tag, а також між Post і Ingredient реалізовано зв’язки типу «багато до багатьох». Крім того, сутність Profile також має зв’язок типу «багато до багатьох» із сутністю Tag, що дозволяє зберігати інтереси та вподобання користувачів.

Для реалізації соціальних функцій вебзастосунку використовуються сутності Followers та LikePost. Сутність Followers зберігає інформацію про підписки між користувачами, а сутність LikePost – дані про вподобання, поставленні користувачами до рецептів. Ці сутності дозволяють реалізувати функції для взаємодії користувачів та зручного використання вебзастосунку.

2.5 Реалізація локалізації

Так як основною ідеєю є розробко вебзастосунку для обміну кулінарними рецептами, що розрахована на українських користувачів було створено український переклад вебзастосунку. Продукт розроблявся англійською і було прийнято рішення залишити для можливого росту та залучення більшої кількості користувачів.

Було використано i18n, файли перекладу .po та .mo і механізм gettext. Було вручну зроблено переклад для усіх слів для кращої адаптації та вигляду.

3 ОБҐРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ ВЕБЗАСТОСУНКУ

Середовищем розробки було обрано Visual Studio Code через свою зручність та підтримку усіх необхідних функцій. Visual Studio Code має велику кількість доповнень, серед яких можна знайти і Python з Django фреймворк. Інтегрований термінал для зручності, а при використанні Django термінал використовується дуже часто.

3.1 Вибір мови програмування Python

Python – це одна з найпоширеніших, простих завдяки синтаксису мов програмування. Для веброзробки це ідеальний вибір, бо швидко, просто і легко.

Python має велику базу користувачів, які в свою чергу створюють велику кількість бібліотек, фреймворків та додатків. Тому це є великою перевагою, а особливо для веброзробок є Django. Сам Python має вбудовану роботу з віртуальним середовищем, базою даних та читанням файлів, що робить його ще зручнішим. Python використовувався для написання основної логіки та механізмів системи.[1]

3.2 Фреймворк Django

Django – це фреймворк, з відкритим вихідним кодом, який використовувався в даному проєкті для реалізації серверної частини вебзастосунку. Це високорівневий фреймворк, написаний мовою програмування Python. Він має багато готових компонентів, які допомагають при створенні

вебзастосунків, щоб не витрачати час на реалізацію великої системи. Отже, основною метою Django є спрощення та прискорення процесу розробки вебзастосунків завдяки наданню великої кількості готових компонентів [2].

Однією з головних переваг Django є дотримання DRY (Don’t Repeat Yourself), який спрямований на мінімізацію дублювання коду. Завдяки цьому можна зосередитись на реалізації бізнес-логіки застосунку, а не на створенні допоміжних механізмів [2].

У проєкті окрім реалізації серверної частини вебзастосунку, Django обробляє HTTP-запити, взаємодії з базою даних, керування користувачами та формуванням динамічних вебсторінок. Фреймворк має готові інструменти для роботи з маршрутами, шаблонами, моделями даних, локалізацією та автентифікацією користувачів.

Ще однією важливою перевагою при виборі Django було зазначено наявність вбудованої адміністративної панелі, яка автоматично генерується на основі створених моделей. Для початкової системи це дуже зручно та спрощує адміністрування системи і керування даними без необхідності створення нового інтерфейсу чи сторінки. Права доступу до адміністративної панелі регулюються та захищаються самим Django.

Безпека забезпечується механізмами Django. Вони можуть захистити від найбільш поширених вебзагроз, серед яких SQL-ін’єкції, міжсайтовий скриптинг, міжсайтові підробки запитів та інші типи атак. Саме тому фреймворк широко використовується для створення сучасних вебзастосунків [2].

3.2.1 Архітектура MVT

Django має архітектурний шаблон MVT (model-view-template), що є забезпечує зручність та швидку роботу. Серед основних переваг є система маршрутизації URL, ORM шаблони для роботи з базою даних, систему моделей, шаблонів та представлень, механізми авторизації та адміністрування, підтримка локалізації та надання можливості роботи з статичними файлами [2].

3.2.2 ORM Django

ORM (Object-Relational Mapping) – це механізм для контакту через Python-об’єкти з базою даних, без написання власноруч SQL-запитів. ORM дозволяє роботи майже все що можна роботи з базами даних, тобто створювати, фільтрувати, шукати, видаляти, створювати різного виду зв’язки. Після написання моделей та встановлення зв’язків необхідно зробити міграцію та мігрувати. Це створить нашу базу даних та таблиці так як ми це прописали [2].

3.2.3 Система шаблонів

Django має можливість формувати шаблони HTML-сторінок і передавати дані з серверу, використовувати цикли та умови серед HTML-коду, підключати статичні файли та легко реалізовувати локалізацію. Під час розробки було використано Jinja, яка має простий синтаксис та легка в застосування.

3.2.4 Система міграцій

Система міграцій використовується для автоматичного керування структурою бази даних на основі описаних моделей у вигляді коду мовою Python.

Після створення або зміни моделей необхідно виконати генерацію міграції, яка фіксує всі зміни. Наступним кроком буде застосування міграції. Структура бази даних автоматично оновлюється відповідно до описаних моделей і робиться запис на випадок не правильної міграції, щоб потім була можливість повернутись до минулої версії. Крім цього, міграції легко переносять застосунок між різними середовищами розробки та розгортання без втрат цілісності даних. У проєкті було використано міграції для поступового створення таблиць та оновлення даних і їх властивостей (рисунок 3.1).

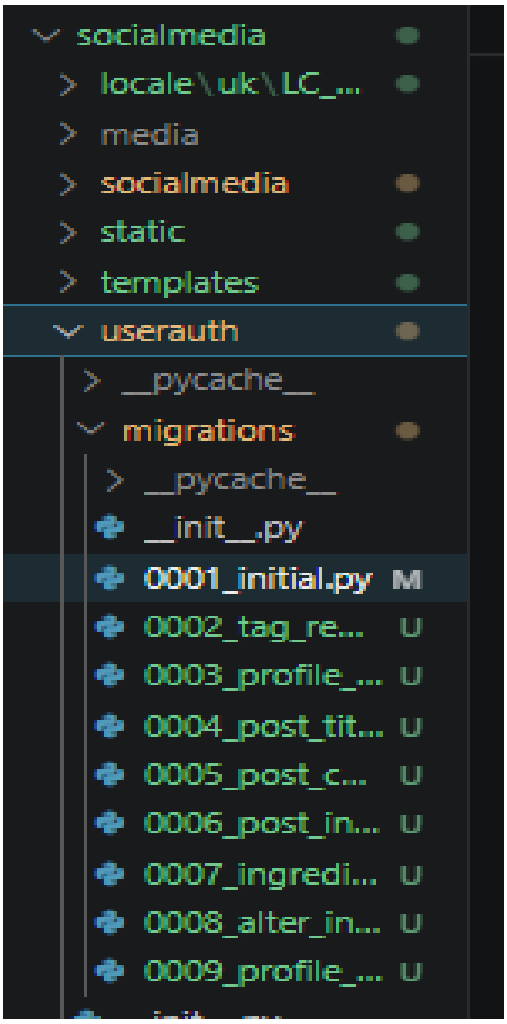

Рисунок 3.1 — Записи про міграції проєкту

Загалом було створено такі таблиці: користувачів, профілів, публікацій, тегів, інгредієнтів, підписок та вподобань. Ці таблиці зберігають відповідні дані кожної сутності.

3.2.5 Засоби безпеки та адміністрування

Для безпеки вебзастосунку Django використовує комплекс вбудованих механізмів захисту. Одним із них є система автентифікації користувачів, яка забезпечує шифрування унікальним ключем паролів, а далі безпечне зберігання [2].

Фреймворк автоматично виконує екранування даних у шаблонах, що зменшує ризик виникнення атак типу XSS. Для захисту від підробки міжсайтових запитів використовується механізм CSRF-токенів, який додається

до форм та перевіряється під час їх відправлення. Також важливо зазначити, що під час створення вебзастосунку для нього створюється його унікальний SECRET_KEY. Тому при заливанні продукту на GitHub чи на інші платформи потрібно прибирати та не передавати нікому цей ключ. Адже з ним можна отримати доступ до всього.

Наявність адміністративної панелі є неабиякою перевагою. Адміністратор системи може переглядати, створювати, редагувати та видаляти записи в базі даних через зручний вебінтерфейс (рисунок 3.2).

Рисунок 3.2 — Фрагмент адміністративної панелі розробленого продукту

Адміністративна панель автоматично працює з усіма моделями, зареєстрованими в системі, що значно спрощує підтримку вебзастосунку.

В подальшому розвитку продукту розглядається налаштування адміністративної панелі та додавання ще більш кращого захисту вебзастосунку.

3.3 Використання SQLite

База даних, як відомо, використовувалась для збереження даних. SQLite вбудований в Django, тому не потребує окремого з'єднання та налаштування. Він простий, вже інтегрований та зручно тестується [5]. Тому на початку реалізації вебзастосунку використовується саме ця база даних, бо має багато переваг (таблиця 3.1).

Таблиця 3.1 – Основні переваги SQLite

Перевага	Опис
Простота	З використання Django взагалі нічого не потрібно робити напряму з SQLite
Компактність	Уся база даних зберігається в одному файлі
Швидкодія	Забезпечує швидкий доступ до даних
Інтеграція	Повністю підтримується Django і не потребує додаткових налаштувань
Доступність	Не потребує ліцензії (безкоштовний).

В подальшому розвитку буде замінений на іншу базу даних PostgreSQL. Адже при зростанні навантаження і кількості записів краще використовувати більш надійну та потужнішу СУБД.

3.4 Реалізація користувацького інтерфейсу

Вебзастосунок використовує базові інструменти для створення користувацького інтерфейсу: HTML, CSS та Bootstrap. Їхній функціонал буде поданий у вигляді таблиці (таблиця 3.2).

Таблиця 3.2 – Опис функціоналу інструментів

Інструмент	Функціонал
HTML	Створення структури сторінок, формування елементів інтерфейсу та створення форм для роботи з даними
CSS	Створення стилю, налаштування кольорів та тексту, реалізація адаптації інтерфейсу до розмірів вікна.
Bootstrap	Швидко створює адаптивні сторінки, використовує готові стилі та легко оптимізовує сторінки для мобільних пристроїв.

Отже, завдяки використанню HTML та Jinja ми маємо структуру сторінок та форми для роботи з даними. CSS та Bootstrap дають нам стиль, адаптивність.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

В цьому розділі буде розглянуто детальніше реалізацію вебзастосунку. Буде описано саму структуру Django, реалізації систем авторизації, публікації рецептів, пошуку, системи підписок та вподобань.

4.1 Структура Django фреймворку

Використовуючи Django, маємо розділений функціонал системи на окремі компоненти, що дозволяє швидко та легко налаштовувати функціонал сайту.

Почнемо з головного компоненту settings.py в цьому файлі знаходяться основні налаштування, токени та всі сервіси які підтримує сайт (рисунок 4.1).

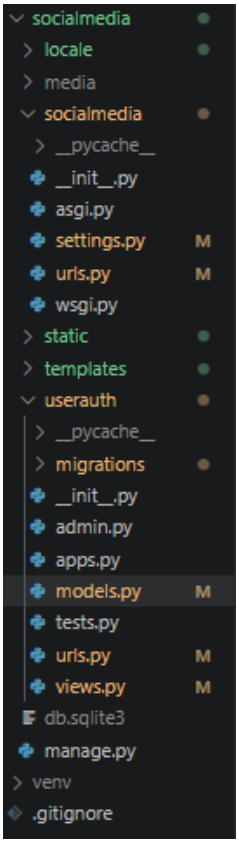

Рисунок 4.1 — Структура проєкту

Тут ми налаштовуємо, де будуть знаходитись статичні файли, медіа файли, підключаємо базу даних. І окремо підключаємо можливість додавання локалізації.

В наступному модулі ми маємо URLs тобто наші посилання, адреси до сторінок. Ми налаштовуємо розділення для різних мов, та в основному файлі URL прописуємо потрібні нам адреси для сторінок профілів, рецептів, пошуку і так далі.

Наступний компонент структури це моделі. За допомогою ORM Django, можна сказати, що моделі це наші таблички в базі даних. Розглянемо, як приклад модель профілю.

Отже, ми маємо таблицю «Profile», вона містить стовпчики «user», «bio», «profileimg» «location» «language» «background_color» та «tags». Для кожного стовпчику ми встановлюємо тип даних, там де можна максимальний розмір, значення за замовчуванням та якщо нам потрібно то зв’язки.

З компонентом моделей та маршрутизації пов’язаний наступний — модуль представлень views.py. Цей модуль можна назвати нашим мозком. В ньому ми прописуємо весь функціонал та описуємо як будуть витягатись дані, що буде з ними відбуватись і які дії може виконувати користувач.

Останнім, але не менш важливим компонентом є HTML-шаблони. В цих шаблонах і є наш інтерфейс та зовнішній вигляд сторінок.

4.2 База даних

Для зберігання інформації ми використовуємо SQLite. У системі використовується моделі ORM Django. Тобто, ми описуємо у вигляді коду на Python, якими мають бути таблиці бази даних. Ці моделі автоматично створюють таблички та дозволять працювати з базою даних у вигляді Python коду.

4.2.1 Модель Profile

Модель Profile використовується для збереження даних про користувача. Користувач може зберігати свій унікальний username, біографічні дані, картинку профіля, місце розташування, мову, колір шапки профілю та теги. Модель пов’язана з вбудованою моделлю user через зв’язок «один до одного».

Було створено таблицю для кращого розуміння полів та їх типів даних таблиці Profile (таблиця 4.1).

Таблиця 4.1 – Опис полів таблиці Profile

Поле	Тип
User	OneToOneField
Bio	TextField
Profileimg	ImageField
Location	CharField
Language	CharField
Background_color	CharField
Tags	ManyToManyField

Отже, для отримання даних для поля user ми використовуємо зв’язок «один до одного» з вбудованої таблиці user. Далі для поля біографії ми використовуємо текстовий тип даних, для поля фотографії використовуємо тип даних зображення, для решти окрім Tags маємо тип даних для малих рядків. І для отримання Tags маємо зв’язок з таблицею відповідної назви «багато до багатьох».

4.2.2 Модель Post

Модель пост зберігає назву публікації, унікальний номер, автора, картинку, опис, час публікації, кількість вподобань, теги, інгредієнти та час приготування. Як і з усіма таблицями ми описували таблицю Post у вигляді

програмного коду. Розглянемо таблицю для кращого розуміння таблиці Post (таблиця 4.2).

Таблиця 4.2 – Опис полів таблиці Post

Поле	Тип
Title	CharField
Id	UUIDField
User	CharField
Image	ImageField
Caption	TextField
Created_at	DateTimeField
No_of_likes	IntegerField
Tags	ManyToManyField
Ingredients	ManyToManyField
Cook_time	PositiveIntegerField

Отже, для назви публікації, ім’я автора публікації було використано тип даних короткого рядка. Далі для поля фотографії використовуємо тип даних зображення, для унікального номера використовуємо тип UUIDField. Так як опис приготування рецепту буде мати велику кількість тексту то було використано тип даних для великого рядка. Для зберігання дати створення посту використовується тип даних часу. Для кількості лайків та часу приготування використовується тип даних натурального числа. І для отримання Tags та Ingredients маємо зв'язки з таблицями відповідної назви «багато до багатьох».

4.2.3 Моделі Tag, Ingredient

Моделі тегів та інгредієнтів зберігає лише їхню назву. Отже, таблиці для тегів та інгредієнтів дуже прості адже мають давати тільки назви цих самих тегів

та інгредієнтів. Було створено одну таблицю для кращого розуміння таблиць Tag, Ingredient (таблиця 4.3).

Таблиця 4.3 – Опис полів таблиць Tag, Ingredient

Поле	Тип
name	CharField

Отже, через схожість програмної реалізації було зроблено одну таблицю, адже, що в одній таблиці, що в іншій ми маємо одну й ту саму назву поля. Для поля назви тегу чи інгредієнта було використано тип даних короткого рядка. Тег та інгредієнт не має мати велику назву тому обраний тип аргументований.

4.2.4 Моделі LikePost та Followers

Модель LikePost зберігає унікальний номер публікації та користувача, який вподобає публікацію. Модель Followers зберігає користувача та його кількість підписників.

Отже, маємо невеликі таблиці, які містять коротку програмну реалізацію. Було створено одну таблицю для кращого розуміння таблиць LikePost, Followers (таблиця 4.4).

Таблиця 4.4 – Опис полів таблиць LikePost, Followers

Поле	Тип
Post_id	CharField
Username	CharField
Follower	CharField
User	CharField

Отже, ім’я користувача, унікальний номер та підписника було описано типом даних короткого рядка, адже вони не мають містити великої кількості даних.

4.3 Реалізація систем вебзастосунку

Завдання, які має виконувати вебзастосунок було розбито на декілька окремих систем. Системи мають реалізовувати свій функціонал та вільно взаємодіяти між собою.

4.3.1 Реалізація системи авторизації

Для реалізації системи авторизації використовувалась вбудована система Django Authentication System. Було розроблено реєстрацію. Дані для реєстрації користувача передаються з форми, яку заповнює сам користувач на відповідній сторінці. Записується ім’я, електронна пошта, пароль та створюється користувач. Далі, відбувається перенаправлення на домашню сторінку. Але у випадку якщо користувач вже існує то не зареєстрований користувач отримає помилку та буде мати спробу ще раз зареєструватись.

Переходимо до опису входу та виходу у вже існуючий профіль. За допомогою вбудованої системи авторизації, ми передаємо дані, які ввів не

авторизований користувач та перевіряємо відповідність даних. Після успішної перевірки користувача авторизує та перенаправляє на домашню сторінку. У випадку невідповідності паролю до користувача буде виведене помилка на екран та надана можливість спробувати ще раз увійти в профіль.

Якщо ж користувач з будь-якої сторінки сайту захоче вийти з профілю то він здійснюється відразу через використання вбудованої системи автентифікації Django. І для надійності було додано перевірку на авторизацію, щоб використати вихід з сайту, це зроблено на випадок збереження файлів cookie або якщо сторінку з самим рецептом було поширено для не зареєстрованого користувача.

4.3.2 Реалізація системи публікацій

Кожен профіль може створювати свої публікації з рецептами, де має вказати картинку, назву публікації, опис, теги, інгредієнти та час приготування.

Після обробки дані зберігаються в базі даних та одразу з’являються в стрічці автора, користувачів, які підписані на автора, та в загальній сторінці огляду. Окрім додавання вручну тегів до публікації, додати тег при зміні профілю не можна вручну. Було зроблено систему додавання тегу до профілю, щоб досягти цього потрібно створити мінімум п’ять публікацій з одним й тим самим тегом і після цього до профілю автоматично присвоїться цей тег і при пошуку він буде висвічуватись користувачам.

Для того щоб відкрити публікацію на всю сторінку необхідно натиснути на назву посту і після цього, користувачу буде доступний детальний перегляд рецепту. Функція з детальною інформацією публікації відкриває публікацію якщо отримує її унікальний номер або помилку 404. Далі підвантажуються теги та інгредієнти і важливо зазначити, що на самій сторінці HTML є перевірка чи перейшов автор на цю сторінку зі своїм рецептом.

При переході автора на свою публікацію він може редагувати її та змінювати всю інформацію про неї. Для редагування публікації створена окрема функція. При зміні даних, нова інформація про публікацію оновлюється в

таблиці бази даних. При додаванні нових тегів чи інгредієнтів буде таке саме розбиття на окремі теги та інгредієнти.

Для видалення публікації було реалізовано окрему функцію. Публікацію може видалити відповідно лише автор і це відбувається, за допомогою вбудованої системи post в Django.

4.3.3 Реалізація системи профілів

Після авторизації кожен користувач може змінювати дані профілю: додавати якусь інформацію або змінювати наяву. Також, змінювати картинку та колір шапки профілю, для цього всього було створено відповідну функцію профілю. Ще на сторінці профілю можна переглянути свої публікації та теги. Теги користувач не може сам собі вписати. Для того, щоб тег присвоївся до користувача, користувач має створити щонайменше 5 постів, функції для чого було описано вище.

Для домашньої сторінки, де розміщена персоналізована стрічка було також реалізовано свою функцію. На домашній сторінці застосовується фільтр для відображення і своїх публікацій, і публікацій користувачів, на яких підписаний авторизований користувач. Фільтр відображає згори найновіші публікації, а нижче йдуть публікації пізніше по даті створення.

Також, схожий фільтр та відображення усіх публікацій працює на окремій сторінці «Огляд». На цій сторінці відображаються просто картинки постів та є можливість перейти на сторінку детальної публікації. Окрім цього, для зміни кольору обгортки на сторінці профілю було додано окрему модальну сторінку та її функцію. Після того як користувач обере новий колір він зберігається та одразу оновлюється на сторінці профілю та відповідне поле у таблиці бази даних.

4.3.4 Реалізація системи пошуку

Для пошуку рецептів, користувачів використовується ORM Django та об’єкти Q. Реалізований пошук дозволяє шукати теги, інгредієнти, користувачів

та публікації. На будь-якій сторінці, прописані теги можна на них натиснути і перейти одразу на пошук за цим тегом.

4.3.5 Реалізація системи стеження та вподобань

Користувач може натиснути кнопку вподобання і автоматично створиться запис у таблиці і дані про публікацію оновляться і буде додано кількість вподань публікації. Майже та сама логіка з підписниками. Користувач може зайти на профіль іншого користувача та натиснути кнопку стеження після чого виконається запис у таблицю і оновиться кількість підписок та підписників у обох користувачів відповідно. А також після підписки публікації користувача, на якого було виконане стеження будуть з’являтись на домашній сторінці користувача, який почав стежити. Функція для стеження виконує перевірку користувача та виконує дію стеження методом створення об’єктів підписки і підписника.

Функція для системи вподобань перевіряє чи правильно підвантажує публікацію і виконує операцію вподобання. Також, виконується перевірка чи не поставив користувач вподобання і якщо кнопка натиснута повторно то функція викликається і вподобання одразу прибирається.

4.4 Реалізація шаблонів Django

Для вебзастосунку було реалізовано такі шаблони як: головна сторінка, сторінка реєстрації та входу, налаштування, пошук, створення публікацій та редагування, сторінка профілю та редагування та огляд публікації (рисунок 4.2).

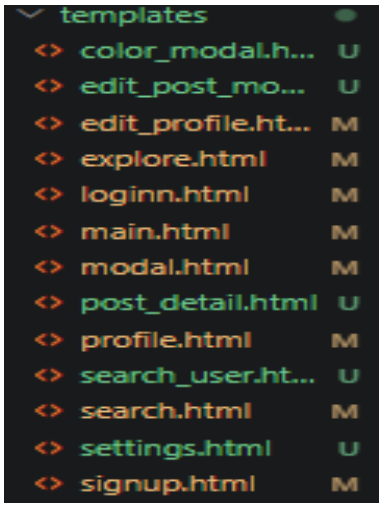

Рисунок 4.2 — Структура сторінок

Сторінки виконанні з використанням HTML, CSS, Bootstrap та Jinja. Jinja необхідна для використання статичних файлів, які необхідні нам, щоб створити панелі для створення та зміни інформації, а також, для реалізації локалізації сайту. Локалізація сайту виконана, за допомогою Django Internationalization.

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З ВЕБЗАСТОСУНКОМ

У користуванні вебзастосунком немає ніяких вимог, щодо встановлення будь-яких файлів. При проєктуванні проєкту було враховано усі важливі та зайві механізми з сайтів конкурентів.

5.1 Системні вимоги

Розроблений вебзастосунок не потребує окремої інсталяції. Він працює через браузер, а отже, користувачу необхідно мати браузери з підтримкою HTML5, CSS3. Такими браузерами можуть бути Google Chrome, Mozilla Firefox, Microsoft Edge, Opera. Вебзастосунок може використовуватись на комп’ютерах, ноутбуках, планшетах та телефонах. Серверна частина працює з використанням Python, Django, SQLite, Jinja.

5.2 Запуск вебзастосунку

Запуск вебзастосунку має здійснювати адміністратор через консоль (рисунок 5.1).

```
(venv) C:\Users\keteray\Desktop\Template\socialmedia\socialmedia>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 01, 2026 - 22:57:13
Django version 6.0.5, using settings 'socialmedia.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 5.1 — Запуск локального серверу

Прописуючи команду запуску та отримуючи локальну адресу. В подальшому розвитку розглядається запуск на хост та адміністрування вебзастосунку.

5.3 Реєстрація та авторизація

Для реєстрації користувач має ввести ім’я для профілю, який далі буде використовуватись, електронну пошту та пароль (рисунок 5.2).

Рисунок 5.2 — Приклад реєстрації

В подальшому для входу користувач має ввести на сторінці входу ім’я користувача та пароль (рисунок 5.3).

Рисунок 5.3 — Приклад входу

У випадку не співпадіння паролю з ім’ям користувача на сторінці входу буде виведено повідомлення про неправильні дані для входу до профілю (рисунок 5.4).

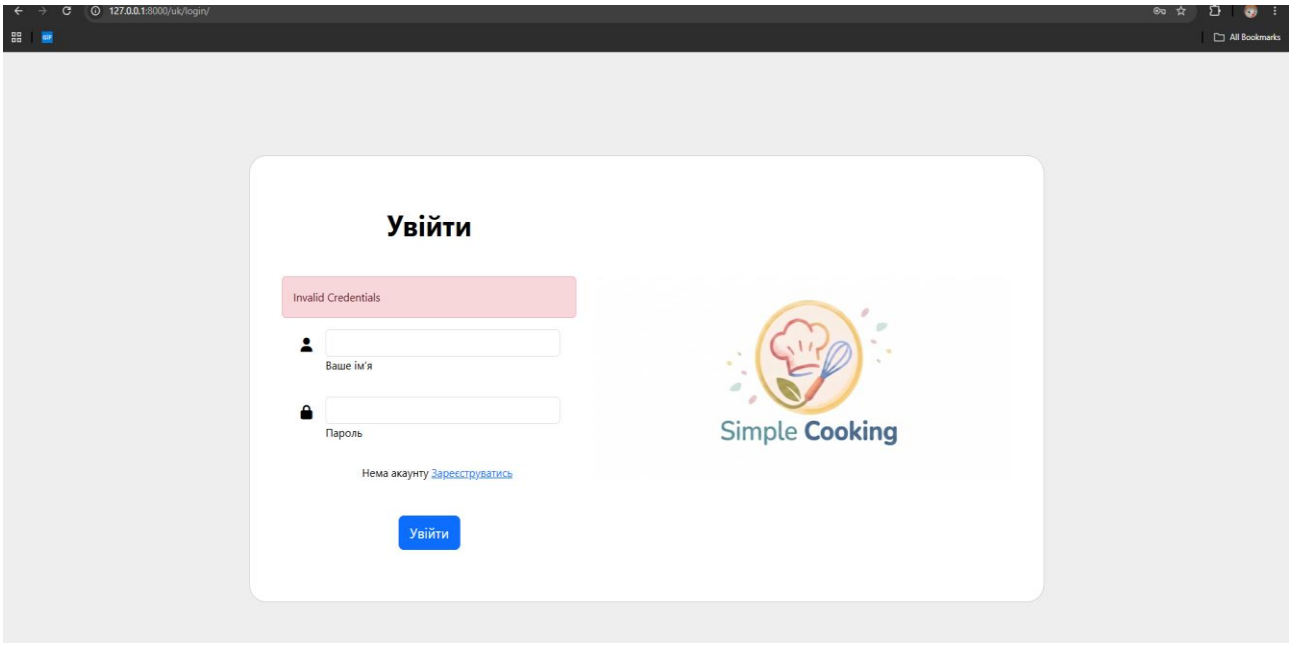

Рисунок 5.4 — Не правильно введений пароль

5.4 Головна сторінка та sidebar

Після реєстрації чи входу користувача перекине на головну сторінку, де буде знаходитись його стрічка, поки що порожня. Користувач має навігаційну дошку з лівої сторони, де він може перейти в огляд, свій профіль, створити пост, перейти в налаштування та відповідно вийти з профіля (рисунок 5.5).

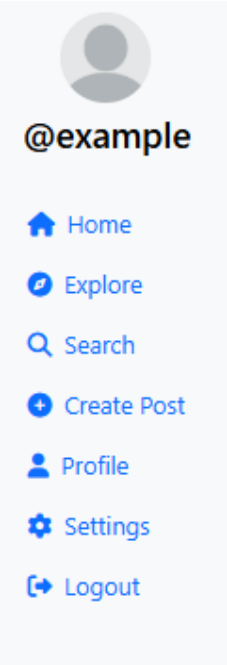

Рисунок 5.5 — Приклад домашньої сторінки

В подальшому на головній сторінці будуть додаватись публікації, якщо користувач створить власну публікацію або підпишеться на іншого користувача.

5.5 Робота з профілем користувача

Перейшовши на сторінку свого профіля, користувач може кастомізувати свій профіль так як захоче. Змінити колір обгортки профілю профілю, за допомогою модального вікна форми (рисунок 5.6).

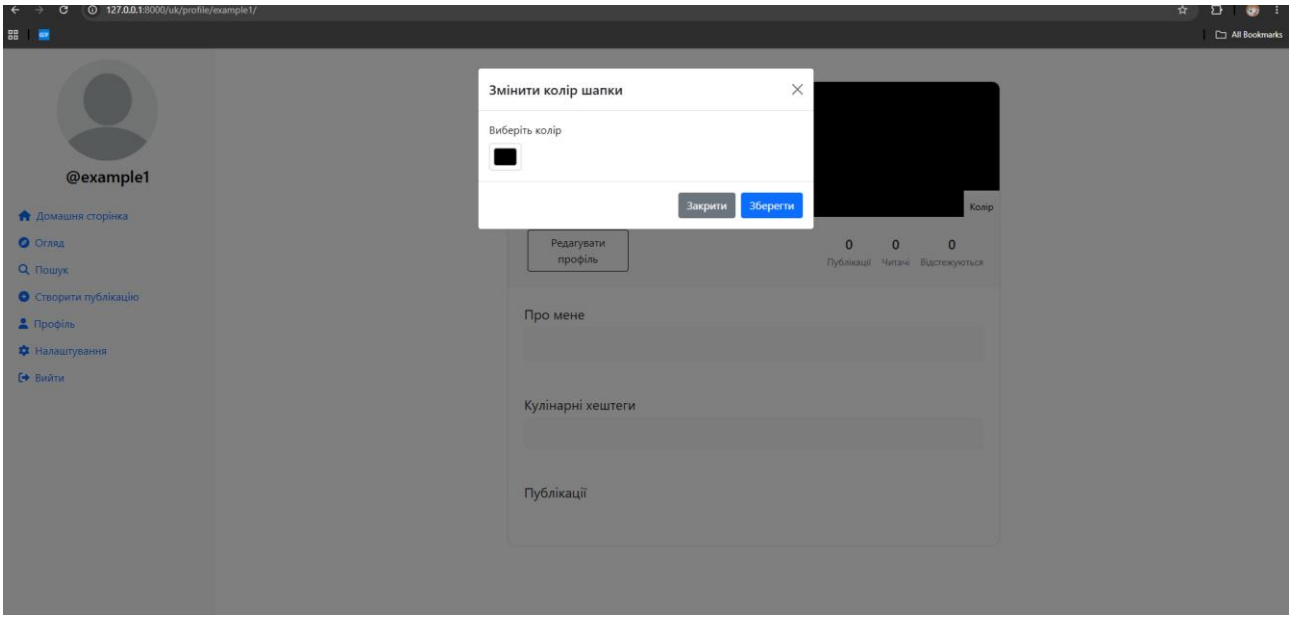

Рисунок 5.6 — Вікно зміни кольору обгортки профілю

Також, користувач може змінювати картинку профілю, біографію, місце розташування але у окремому модальному вікні після натискання кнопки «Редагувати профіль» (рисунок 5.7).

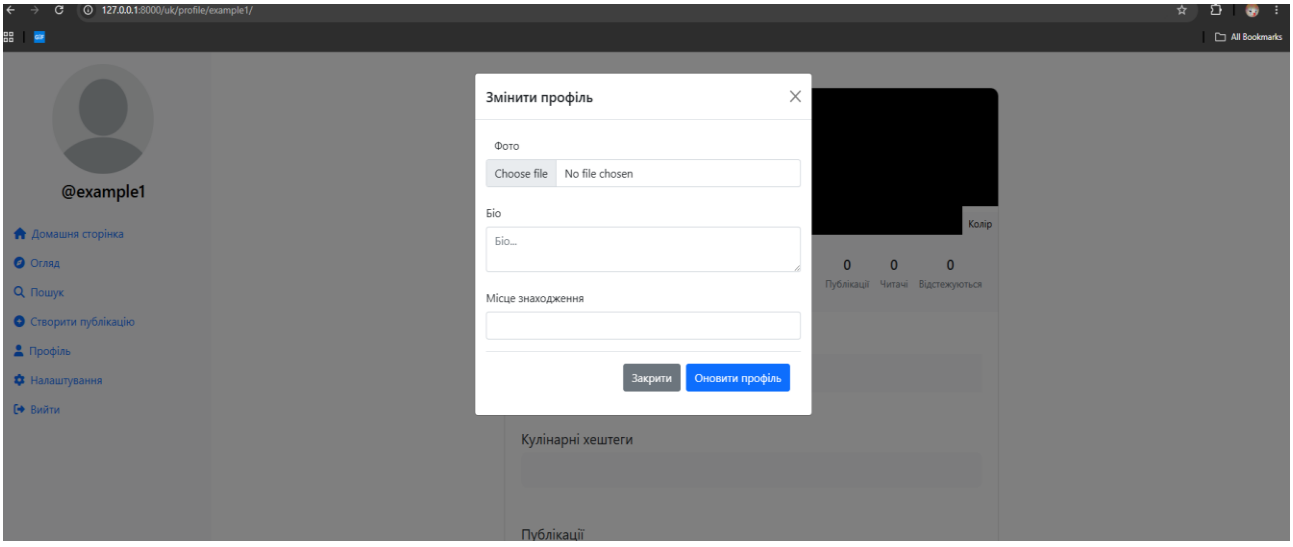

Рисунок 5.7 — Модальна форма зміни інформації профілю

Для демонстрації було змінено колір обгортки на жовтий, змінено фотографію користувача, заповнено розділ «Про мене». Також, можна побачити статистику, скільки користувачів підписані на нього або скільки вподобань мають його пости (рисунок 5.8).

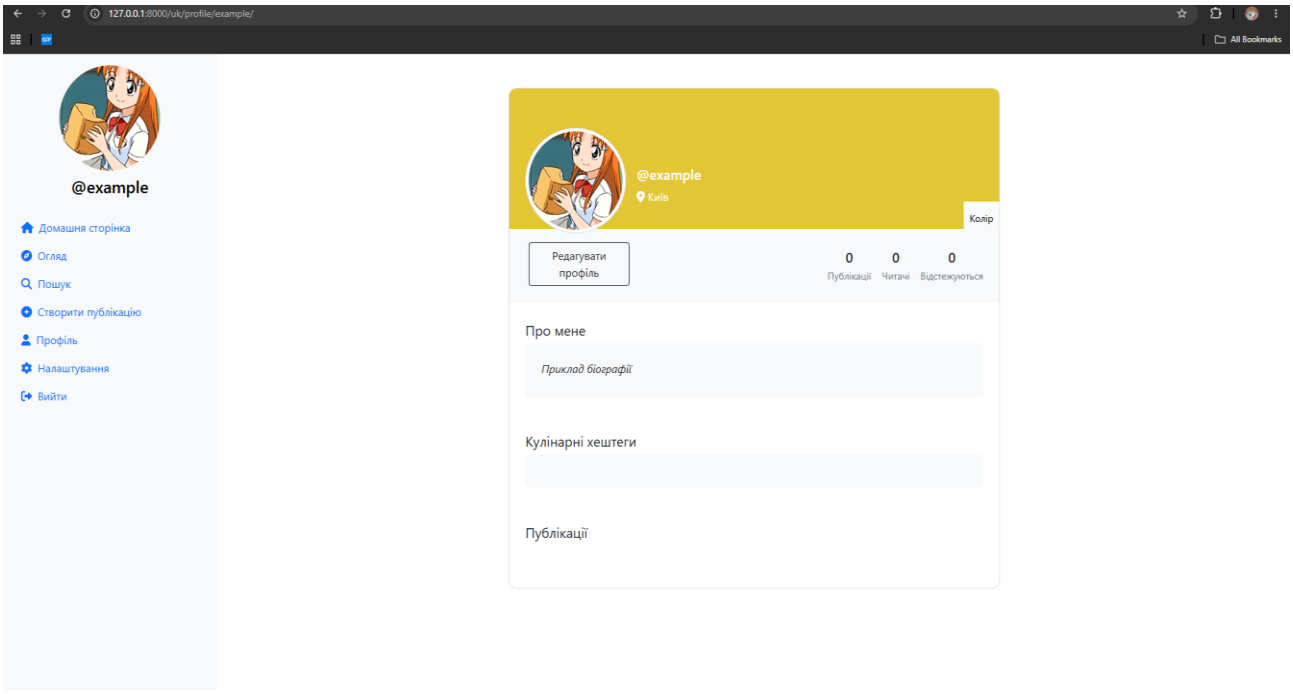

Рисунок 5.8 — Результат зміни інформації профілю

Далі перед користувачем постає завдання з пошуку рецепту. Якщо ж користувач хоче просто пошукати по картинкам або передивитись картинки усіх рецептів, з метою знайти іншого цікавого користувача то він може перейти на сторінку огляду з бокової панелі (рисунок 5.9).

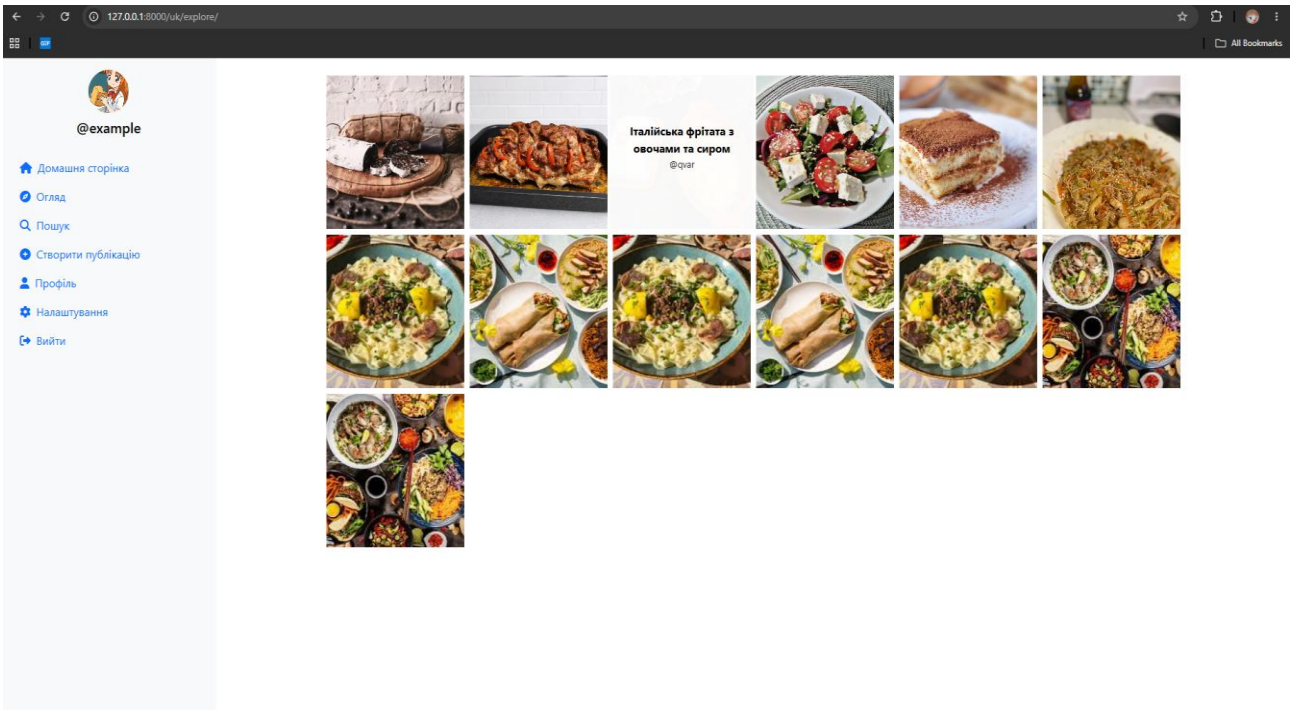

Рисунок 5.9 — Сторінка огляду

Користувач може зайти на публікацію, натиснувши на назву публікації або, натиснувши на ім’я користувача, перейти на сторінку профілю автора публікації (рисунок 5.10).

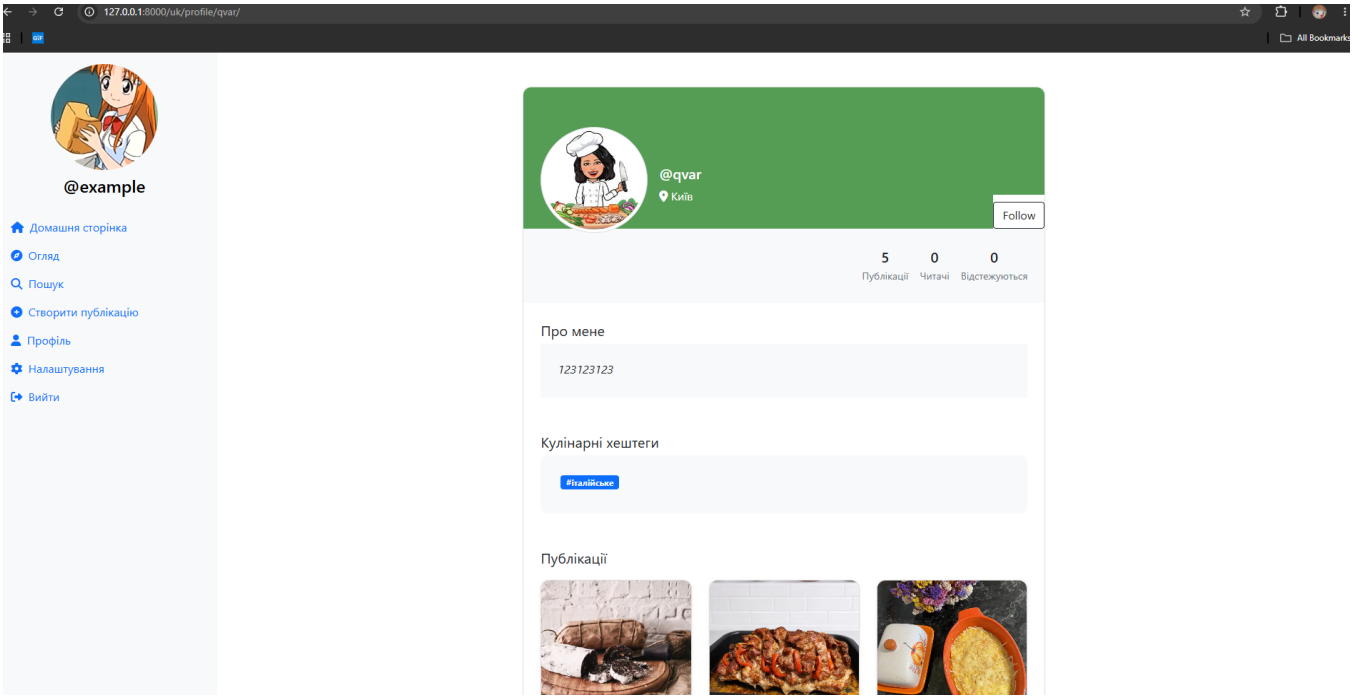

Рисунок 5.10 — Сторінка іншого користувача

Як бачимо, на сторінці іншого користувача ми не можемо змінити колір, замість тієї кнопки є кнопка для підписання на цього користувача. Також, відсутня кнопка редагування профілю. Користувач має багато рецептів з один і тим самим тегом, а отже система надала цей тег на його профіль.

Якщо натиснути на назва публікації, яка нас цікавить, то нас перекине на іншу сторінку. На сторінці буде лише необхідна публікація та її дані (рисунок 5.11).

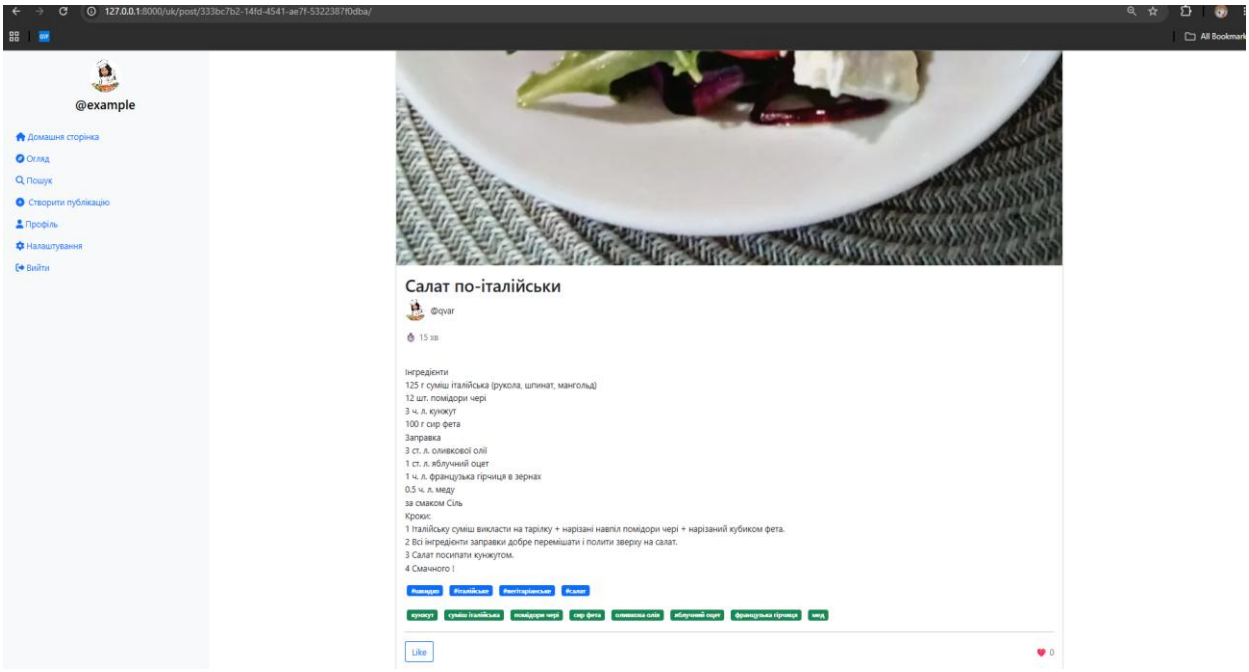

Рисунок 5.11 — Сторінка рецепту іншого користувача

Тепер розглянемо створення своєї публікації. Нам необхідно завантажити фото, описати приготування, ввести теги та інгредієнти. Теги потрібно вписувати самостійно, в майбутньому розглядається автозаповнення з вже існуючих тегів. Інгредієнти потрібно ввести усі які були використанні, а далі в описі приготування ввести грамування кожного інгредієнту (рисунок 5.12).

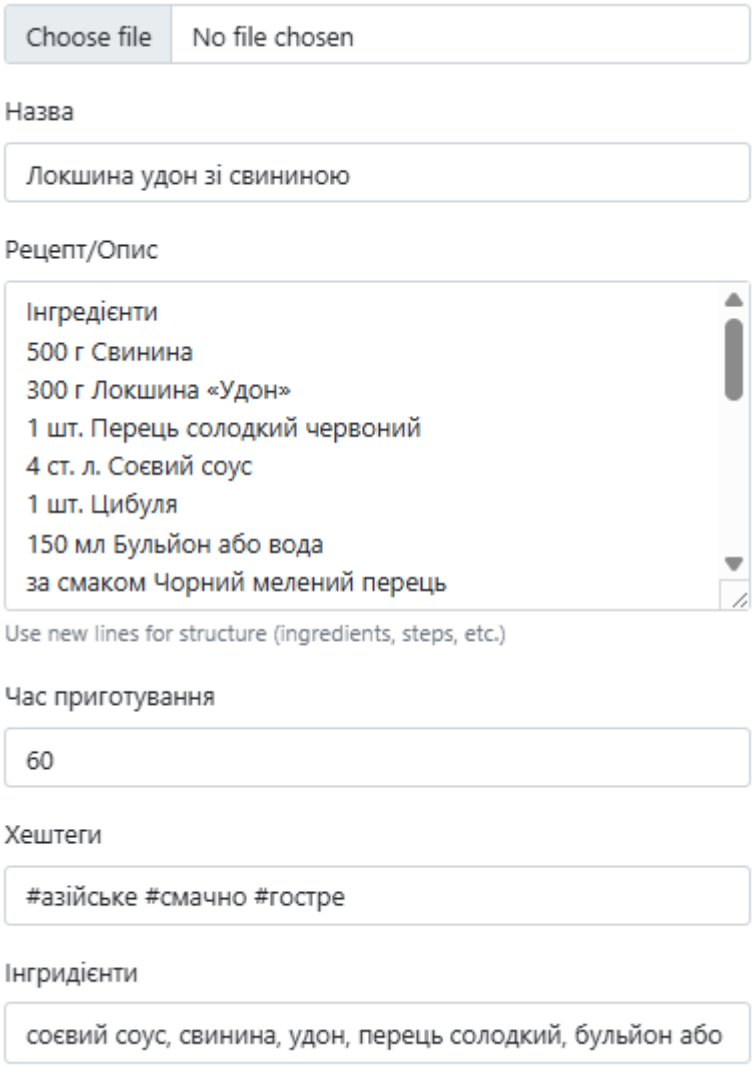

Рисунок 5.12 — Створення власного рецепту

Після створення рецепт автоматично додається на сторінку огляду, профіль та домашню сторінку. Якщо ж автор публікації захоче його редагувати, то він має так само перейти на детальний огляд і йому буде доступна кнопка редагування (рисунок 5.13).

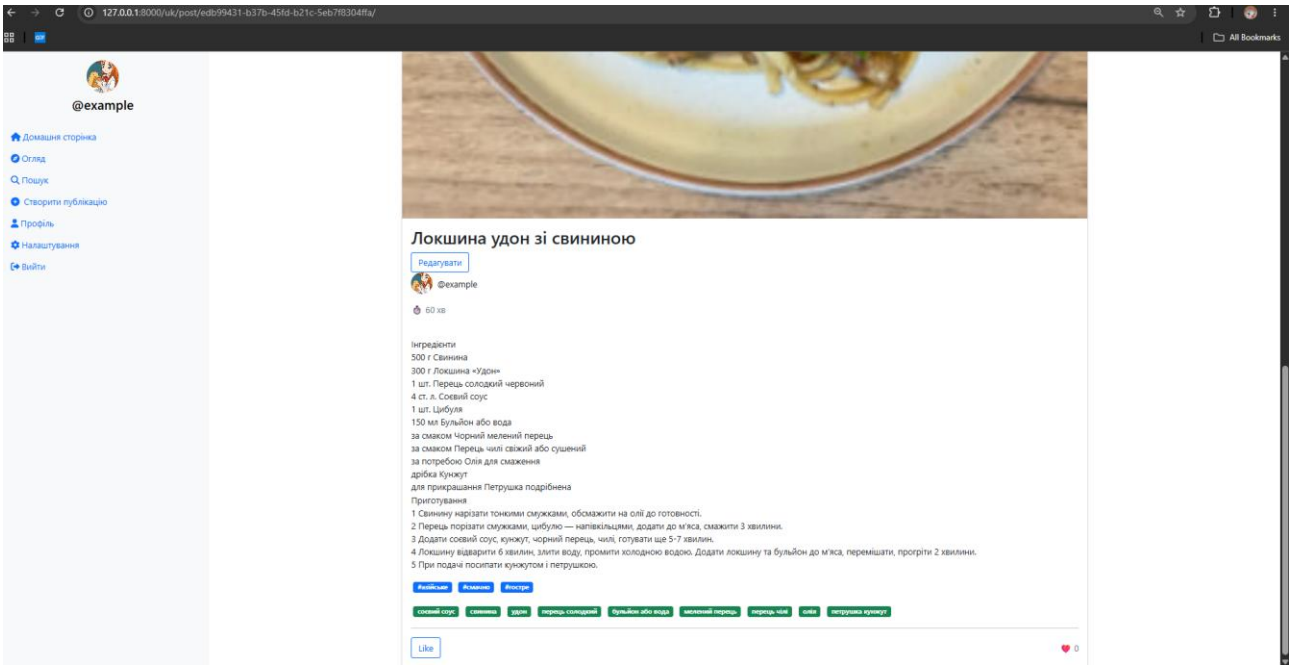

Рисунок 5.13 — Перегляд власного рецепту

На сторінці, окрім редагування, можна подивитись статистику, скільки вподобань має пост.

Отже, маємо простий функціонал для перегляду та редагування рецептів, профілів та взаємодії з користувачами.

5.6 Пошук рецептів

Для пошуку чогось конкретного користувач на боковій панелі може перейти на сторінку пошуку та ввести в пошук необхідні дані. Але якщо користувач знайшов якийсь рецепт і йому сподобався тег чи якийсь інгредієнт він може натиснути прямо на нього та шукати одразу в одне натискання. Основний механізм – це пошук за наявними інгредієнтами (рисунок 5.14).

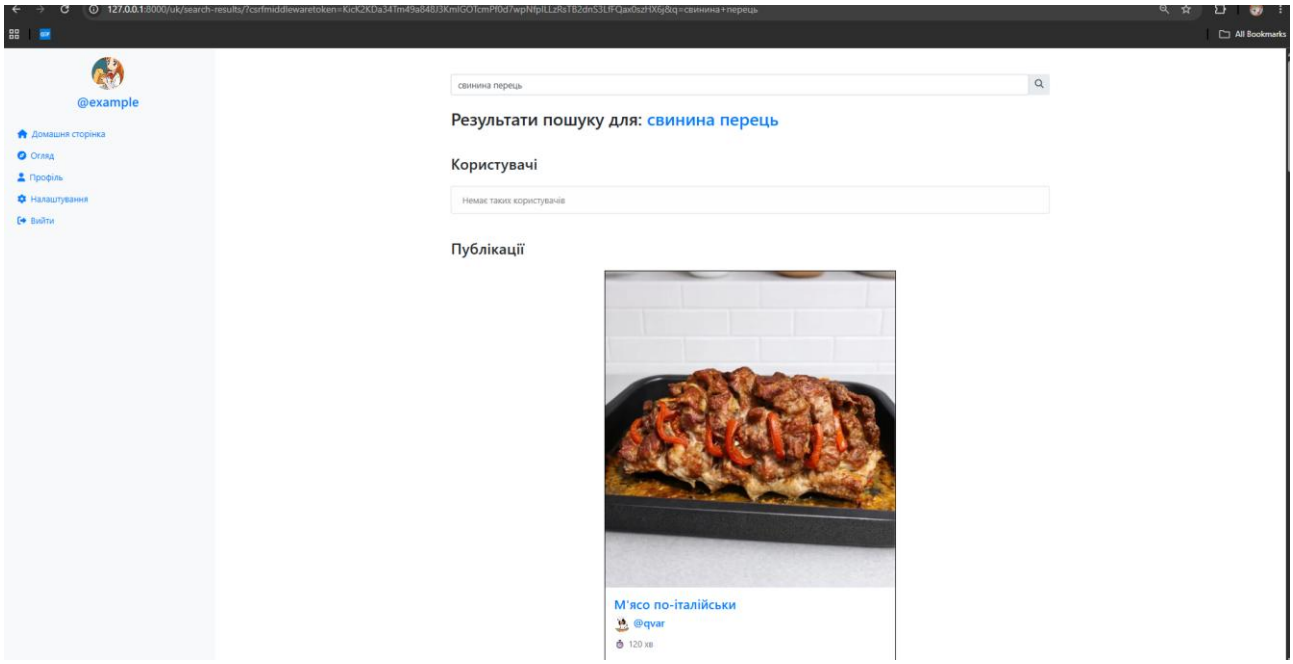

Рисунок 5.14 — Сторінка результатів пошуку за інгредієнтами

А отже, користувач може через пробіл ввести продукти, які є в нього вдома та побачити найбільш схожі рецепти з його набором інгредієнтів.

Важливо розглянути пошук за тегами. Якщо ми будемо шукати за тегом, і якийсь користувач буде завантажувати багато рецептів з цим тегом то він буде виділений згори перед рецептами (рисунок 5.15).

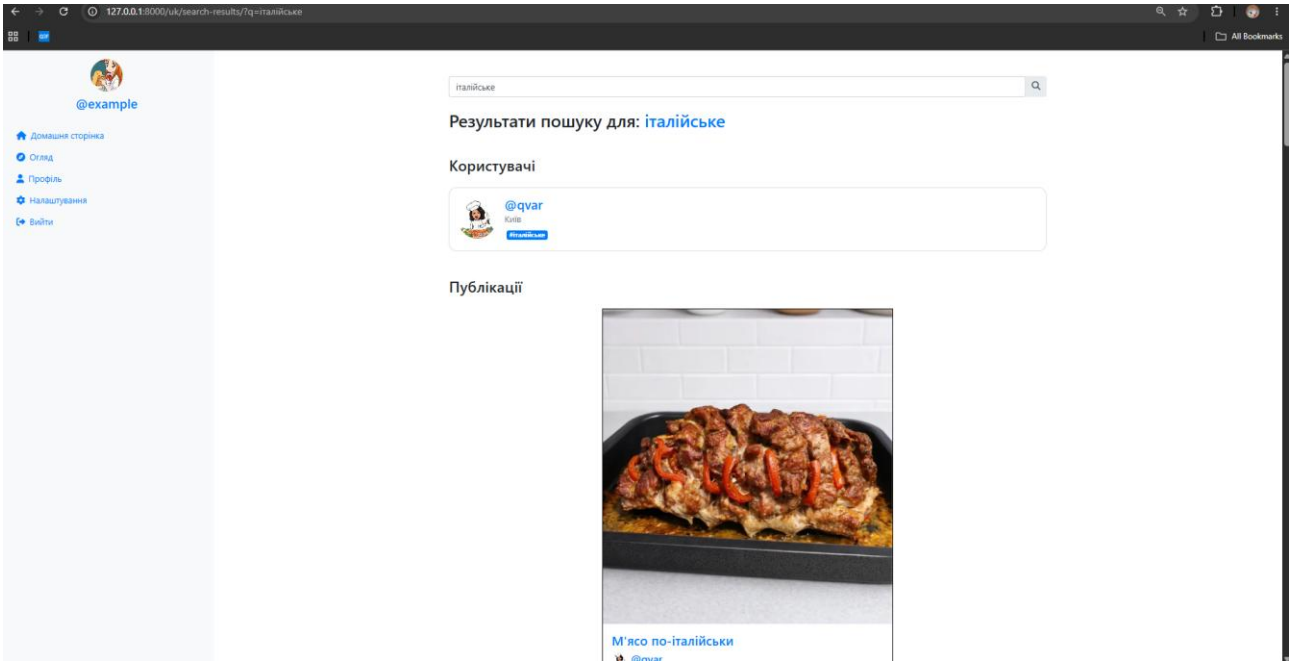

Рисунок 5.15 — Сторінка результатів пошуку за тегом

5.7 Локалізація вебзастосунку

Локалізація була розміщена в налаштуваннях на боковій панелі, де відповідно можна і змінити мову сайту. При зміні мови сайт одразу оновиться та змінить свою мову (рисунок 5.16).

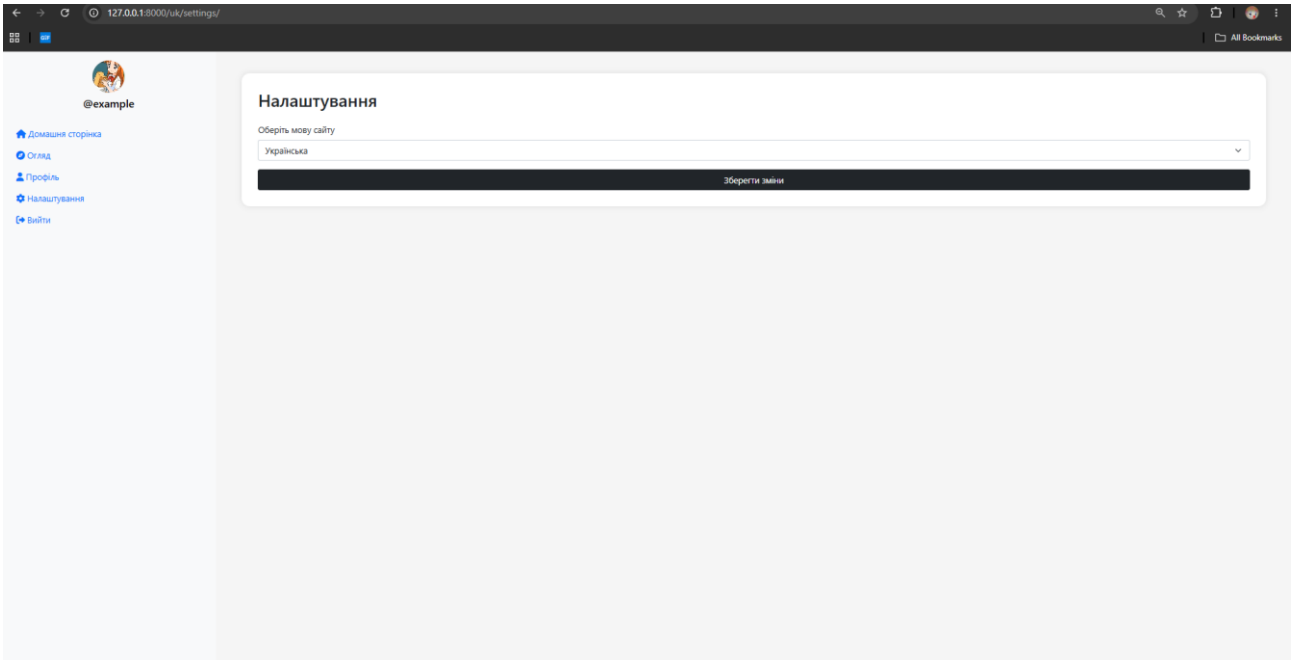

Рисунок 5.16 — Сторінка налаштування мови

Доступні мови: українська та англійська. Після зміни користувач має натиснути на кнопку після чого, зміни одразу застосуються і мова сайту зміниться. В подальшому розвитку вебзастосунку можна буде додати більшу підтримку функціоналу та відповідно мов інтерфейсу.

Таким чином, інтерфейс вебзастосунку включає усі поставленні в завданні дії, а саме: реєстрацію, авторизацію, редагування профілю, створення та редагування рецептів, персоналізацію стрічки, пошук рецептів за тегами та інгредієнтами, видалення рецептів та перемикання мови інтерфейсу. Отже, маємо функціонуючу вебплатформу для обміну кулінарними рецептами.

ВИСНОВКИ

Під час проходження переддипломної практики було виконано аналіз предметної області, поставлено задачі та відповідно розроблено вебзастосунок для обміну кулінарними рецептами за допомогою мови програмування Python та фреймворку Django.

Під час роботи було досліджено вебплатформи, які вже існують. Було виділено їхні переваги та недоліки, для врахування в розробці власної платформи. Найважливішим було відмічено простий інтерфейс, відсутність зайвих функцій, легкість у використанні.

У ході виконання роботи було проаналізовано та спроєктовано структуру вебзастосунку на базі Django. Реалізовано систему реєстрації та авторизації користувачів. Створено функціонал для профілів та публікацій. Створено систему підписок та вподобань. Створено зручну та легку систему пошуку. Додано локалізацію та адаптивний користувацький інтерфейс.

Для реалізації системи було використано Python, Django, SQLite, Jinja, HTML, CSS, Bootsrap.

У результаті маємо бажаний вебзастосунок з усім необхідним для зручності та легкості у використанні. Користувачі можуть легко створювати, шукати та взаємодіяти у цьому вебзастосунку. Отриману систему можна покращувати в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python Documentation: вебсайт. URL: https://www.python.org/ (дата звернення: 26.10.2025).

2. Django Documentation: вебсайт. URL: https://www.djangoproject.com/ (дата звернення: 26.10.2025).

3. Bootstrap Documentation: вебсайт. URL: https://getbootstrap.com/ (дата звернення: 26.10.2025).

4. MDN Web Docs: вебсайт. URL: https://www.mozilla.org/ (дата звернення: 26.10.2025).

5. SQLite Documentation: вебсайт. URL: https://sqlite.org/ (дата звернення: 26.10.2025).

6. HTML Tutorial W3Schools: вебсайт. URL: https://www.w3schools.com/html/default.asp (дата звернення: 26.10.2025).

7. CSS Tutorial W3Schools: вебсайт. URL: https://www.w3schools.com/css/default.asp (дата звернення: 26.10.2025).

8. Matthes E. Python Crash Course: підручник. San Francisco : No Starch Press, 2023. 552 с.

9. Greenfeld D., Greenfeld A. Two Scoops of Django 3.x: Best Practices for the Django Web Framework: підручник. Los Angeles : Two Scoops Press, 2024. 540 с.

10. Jin B., Sahni S., Shevat A. Designing Web APIs: Building APIs That Developers Love: підручник. Sebastopol : O’Reilly Media, 2022. 232 с.

11. Beaulieu A. Learning SQL: Master SQL Fundamentals: підручник. Sebastopol : O’Reilly Media, 2020. 380 с.

12. Duckett J. HTML and CSS: Design and Build Websites: підручник. Indianapolis : Wiley, 2014. 512 с.

ДОДАТОК А

Тези доповіді

у збірнику матеріалів

3 Міжнародної науково-практичної конференції «Science and Technology: New Horizons of Development»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР_21

Аркушів 3

III Міжнародна науково-практична конференція «Synergy of knowledge: New Horizons in Global Scientific Research»
Секція: Технічні науки

ВЕБЗАСТОСУНОК ДЛЯ ОБМІНУ КУЛІНАРНИМИ РЕЦЕПТАМИ

Кобизький Максим Євгенійович
студент спеціальності 122 «Комп'ютерні науки»
Кафедра цифрових технологій в енергетиці
НТУУ «КПІ ім. Ігоря Сікорського»
e-mail: m.kobyzkyi@gmail.com
Отрох Сергій Іванович
доктор технічних наук, професор
Кафедра автоматизації проєктування енергетичних процесів та систем
НТУУ «КПІ ім. Ігоря Сікорського»
e-mail: 2411197@ukr.net

Сучасні інформаційні технології суттєво змінили способи пошуку, зберігання та обміну інформацією. Одним із популярних напрямів використання вебтехнологій є створення спеціалізованих платформ для обміну кулінарними рецептами. Незважаючи на наявність значної кількості подібних сервісів, більшість із них орієнтовані на міжнародну аудиторію та не враховують повною мірою потреби українських користувачів. Крім того, існуючі рішення часто не забезпечують зручного пошуку рецептів за наявними інгредієнтами, обмежують можливості соціальної взаємодії між користувачами або мають недостатній рівень локалізації. У зв'язку з цим виникає необхідність розробки сучасного вебзастосунку, який поєднуватиме функції кулінарного довідника та соціальної платформи для обміну рецептами, забезпечуючи зручний пошук, персоналізацію контенту та комфортну взаємодію користувачів.

Актуальність теми полягає у створенні українського вебзастосунку для обміну та пошуку кулінарних рецептів. Аналіз існуючих платформ показав, що популярні сервіси Cookpad, Allrecipes, Pinterest та Instagram забезпечують лише частину необхідного функціоналу або не мають повноцінної української локалізації. Особливо актуальною є проблема пошуку рецептів за вже наявними інгредієнтами, коли користувач не має можливості придбати додаткові продукти.

Метою роботи є розробка вебзастосунку для обміну кулінарними рецептами з використанням мови програмування Python та фреймворку Django. Для досягнення поставленої мети було проведено аналіз предметної області,

досліджено існуючі рішення, спроєктовано архітектуру системи та реалізовано основні функціональні модулі вебзастосунку.

Таблиця 1. Порівняння функціональних можливостей існуючих сервісів

Сервіс	Пошук	Профілі	Локалізація	Підписки
Cookpad	+	+	-	+
Allrecipes	+	+	-	-
Pinterest	Частково	+	+	+
Instagram	Частково	+	+	+
Розроблений продукт	+	+	+	+

Для реалізації системи було використано фреймворк Django, який базується на архітектурі MVT (Model–View–Template). Архітектура системи включає моделі даних, представлення, HTML-шаблони, систему маршрутизації та базу даних SQLite. Для взаємодії з базою даних використано Django ORM, що спрощує розробку та підтримку програмного забезпечення [1-4].

У вебзастосунку реалізовано систему реєстрації та авторизації користувачів, персональні профілі з можливістю налаштування, створення та редагування рецептів, пошук за назвами, інгредієнтами, тегами та користувачами. Додатково впроваджено механізми підписок та вподобань, що забезпечують соціальну взаємодію між користувачами. Особливістю системи є пошук рецептів за наявними інгредієнтами, що дозволяє швидко знаходити страви на основі продуктів, які вже є у користувача.

Під час розробки використано Python, Django, SQLite, HTML, CSS, Bootstrap та Jinja. Для забезпечення зручності користування реалізовано адаптивний інтерфейс та підтримку української й англійської мов. Вбудовані механізми безпеки Django забезпечують захист від SQL-ін’єкцій, CSRF-атак та інших поширених загроз (рис. 1).

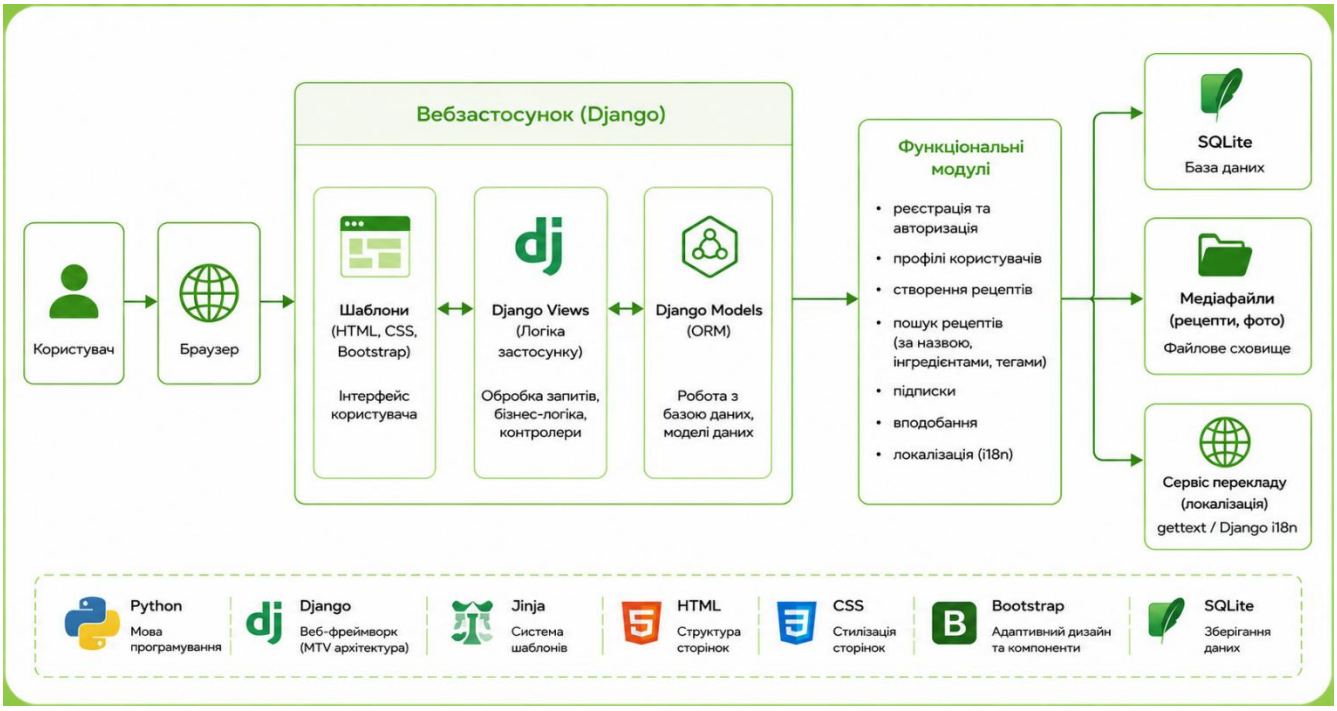

Рисунок 1. Архітектура вебзастосунку для обміну кулінарними рецептами

У результаті виконання роботи було створено функціональний вебзастосунок, який поєднує можливості інформаційної системи та соціальної мережі. Користувачі можуть створювати власні рецепти, взаємодіяти між собою, формувати персоналізовану стрічку та здійснювати швидкий пошук необхідної інформації. Розроблене програмне забезпечення може бути основою для подальшого розвитку платформи, впровадження рекомендаційних алгоритмів та переходу на промислові системи керування базами даних.

Список використаних джерел

1. Python Documentation : вебсайт. URL: https://www.python.org/ (дата звернення: 05.04.2026).
2. Django Documentation : вебсайт. URL: https://www.djangoproject.com/ (дата звернення: 05.04.2026).
3. Bootstrap Documentation : вебсайт. URL: https://getbootstrap.com/ (дата звернення: 05.04.2026).
4. SQLite Documentation : вебсайт. URL: https://www.sqlite.org/ (дата звернення: 05.04.2026).

ДОДАТОК Б

Фрагменти програмного коду розробленого вебзастосунку
«Вебзастосунок для обміну кулінарними рецептами»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР_21

Мова програмування — Python

Аркушів 16

Лістинг програмного коду розробленого вебзастосунку

==========
=
ЧАСТИНА 1. Внутрішня частина вебзастосунку. Реалізація представлень (Python, Django framework)

Файл: socialmedia\userauth\views.py

```
import re
from collections import Counter

from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.db.models import Q
from django.views.decorators.http import require_POST
from django.contrib.auth.decorators import login_required
from django.urls import reverse

from .models import Profile, Post, LikePost, Followers, Tag, Ingredient
import json


def update_user_tags(user):
    profile = Profile.objects.get(user=user)

    posts = Post.objects.filter(user=user.username)

    counter = Counter()

    for post in posts:
        counter.update([t.name for t in post.tags.all()])

    popular_tags = []

    for name, count in counter.items():
        if count >= 5:
            try:
                tag_obj = Tag.objects.get(name=name)
                popular_tags.append(tag_obj)
            except Tag.DoesNotExist:
```

```
            pass

    profile.tags.set(popular_tags)
    profile.save()
@login_required
@require_POST
def update_background_color(request):
    profile = Profile.objects.get(user=request.user)

    color = request.POST.get('background_color')

    if color:
        profile.background_color = color
        profile.save()

    return redirect(request.META.get('HTTP_REFERER', '/'))

def signup(request):
    try:
        if request.method == 'POST':
            fnm = request.POST.get('fnm')
            emailid = request.POST.get('emailid')
            pwd = request.POST.get('pwd')

            user = User.objects.create_user(
                username=fnm,
                email=emailid,
                password=pwd
            )

            Profile.objects.create(user=user)

            login(request, user)
            return redirect('/')

    except:
        return render(request, 'signup.html', {'invalid': 'User already exists'})

    return render(request, 'signup.html')
```

```
def loginn(request):
    if request.method == 'POST':
        fnm = request.POST.get('fnm')
        pwd = request.POST.get('pwd')

        user = authenticate(request, username=fnm, password=pwd)

        if user:
            login(request, user)
            return redirect('/')

        return render(request, 'loginn.html', {'invalid': 'Invalid Credentials'})

    return render(request, 'loginn.html')

@login_required
def logoutt(request):
    logout(request)
    return redirect('/login')


@login_required
@require_POST
def upload(request):
    caption = request.POST.get('caption', '')
    title = request.POST.get('title', '')
    raw_tags = request.POST.get('hashtags', '')
    cook_time = request.POST.get('cook_time', '')
    ingredients_raw = request.POST.get('ingredients', '')

    post = Post.objects.create(
        user=request.user.username,
        image=request.FILES.get('image_upload'),
        caption=caption,
        title=title,
        cook_time=cook_time if cook_time else None
    )

    tags = re.findall(r"#(\w+)", raw_tags)
    for t in tags:
        tag_obj, _ = Tag.objects.get_or_create(name=t.lower())
```

```
            post.tags.add(tag_obj)

        if ingredients_raw:
            ingredients = [
                i.strip().lower()
                for i in ingredients_raw.split(',')
                if i.strip()
            ]

            for name in ingredients:
                ing_obj, _ = Ingredient.objects.get_or_create(name=name)
                post.ingredients.add(ing_obj)

        post.save()

        update_user_tags(request.user)

        return redirect(request.META.get('HTTP_REFERER', '/'))

    @login_required
    def home(request):
        following_users = Followers.objects.filter(
            follower=request.user.username
        ).values_list('user', flat=True)

        posts = Post.objects.filter(
            Q(user=request.user.username) |
            Q(user__in=following_users)
        ).order_by('-created_at')

        for p in posts:
            p.profile = Profile.objects.filter(user__username=p.user).first()
            p.tags_list = [t.name for t in p.tags.all()]
            p.ingredients_list = [i.name for i in p.ingredients.all()]

        profile = Profile.objects.get_or_create(user=request.user)[0]

        return render(request, 'main.html', {
            'post': posts,
            'profile': profile
        })
```

```
@login_required
def likes(request, id):
    post = get_object_or_404(Post, id=id)
    username = request.user.username

    like = LikePost.objects.filter(post_id=id, username=username)

    if like.exists():
        like.delete()
        post.no_of_likes -= 1
    else:
        LikePost.objects.create(post_id=id, username=username)
        post.no_of_likes += 1

    post.save()

    return redirect(request.META.get('HTTP_REFERER', '/'))



def post_detail(request, id):
    post = get_object_or_404(Post, id=id)

    post.tags_list = [t.name for t in post.tags.all()]
    post.ingredients_list = [i.name for i in post.ingredients.all()]

    profile = Profile.objects.filter(
        user__username=post.user
    ).first()

    return render(request, 'post_detail.html', {
        'post': post,
        'profile': profile
    })

@login_required
def explore(request):
```

```
    posts = Post.objects.all().order_by('-created_at')

    for p in posts:
        p.tags_list = [t.name for t in p.tags.all()]

    profile = Profile.objects.get_or_create(user=request.user)[0]

    return render(request, 'explore.html', {
        'post': posts,
        'profile': profile
    })


@login_required
def profile(request, id_user):
    user_object = get_object_or_404(User, username=id_user)

    user_profile = Profile.objects.get_or_create(user=user_object)[0]
    profile = Profile.objects.get_or_create(user=request.user)[0]


    if request.method == 'POST':
        image = request.FILES.get('image')
        bio = request.POST.get('bio')
        location = request.POST.get('location')

        if image:
            profile.profileimg = image

        profile.bio = bio
        profile.location = location
        profile.save()
        return redirect('profile', id_user=request.user.username)

    user_posts = Post.objects.filter(user=id_user).order_by('-created_at')

    follow_unfollow = "Unfollow" if Followers.objects.filter(
        follower=request.user.username,
        user=id_user
    ).exists() else "Follow"
```

```
    return render(request, 'profile.html', {
        'user_object': user_object,
        'user_profile': user_profile,
        'user_posts': user_posts,
        'profile': profile,
        'follow_unfollow': follow_unfollow,
        'user_followers': Followers.objects.filter(user=id_user).count(),
        'user_following': Followers.objects.filter(follower=id_user).count(),
        'user_post_length': user_posts.count(),
        'hashtags_list': user_profile.tags.all()
    })


@login_required
def follow(request):
    if request.method == 'POST':
        follower = request.POST.get('follower')
        user = request.POST.get('user')

        obj = Followers.objects.filter(follower=follower, user=user)

        if obj.exists():
            obj.delete()
        else:
            Followers.objects.create(follower=follower, user=user)

        return redirect(f'/profile/{user}')

    return redirect('/')


@login_required
def delete(request, id):
    post = get_object_or_404(Post, id=id)
    user = post.user

    post.delete()

    user_obj = User.objects.get(username=user)
    update_user_tags(user_obj)
```

```
    return redirect('/profile/' + request.user.username)


@login_required
def search_results(request):
    query = request.GET.get('q', '')
    query = query.strip().lstrip('#').lower()

    if not query:
        return redirect('home')

    profile = Profile.objects.get_or_create(user=request.user)[0]

    keywords = [
        w.strip().lower()
        for w in query.replace(',', ' ').split()
        if w.strip()
    ]

    users = Profile.objects.filter(
        Q(user__username__icontains=query) |
        Q(tags__name__icontains=query)
    ).distinct()
    base_posts = Post.objects.filter(
        Q(tags__name__in=keywords) |
        Q(ingredients__name__in=keywords) |
        Q(title__icontains=query) |
        Q(caption__icontains=query) |
        Q(user__icontains=query)
    ).distinct()

    for post in base_posts:
        post.profile = Profile.objects.filter(user__username=post.user).first()

        post.tags_list = list(post.tags.values_list('name', flat=True))
        post.ingredients_list = list(post.ingredients.values_list('name', flat=True))

    return render(request, 'search_user.html', {
        'query': query,
        'users': users,
        'posts': base_posts,
```

```
        'profile': profile
    })

@login_required
def settings(request):
    profile = Profile.objects.get_or_create(user=request.user)[0]

    if request.method == 'POST':
        image = request.FILES.get('image')
        bio = request.POST.get('bio')
        location = request.POST.get('location')

        if image:
            profile.profileimg = image

        profile.bio = bio
        profile.location = location
        profile.save()

        return redirect('settings')

    return render(request, 'settings.html', {'profile': profile})

def edit_post(request, id):
    post = get_object_or_404(Post, id=id)

    if request.method == "POST":

        post.title = request.POST.get('title')
        post.caption = request.POST.get('caption')
        post.cook_time = request.POST.get('cook_time')
        if post.cook_time:
            post.cook_time = int(post.cook_time)
        else:
            post.cook_time = None
        if request.FILES.get('image'):
            post.image = request.FILES['image']

        raw_tags = request.POST.get('hashtags', '')
        tags = re.findall(r"#(\w+)", raw_tags)
```

```
        post.tags.clear()
        for t in tags:
            tag_obj, _ = Tag.objects.get_or_create(name=t.lower())
            post.tags.add(tag_obj)

        raw_ing = request.POST.get('ingredients', '')
        ingredients = [i.strip().lower() for i in raw_ing.split(',') if i.strip()]

        post.ingredients.clear()
        for name in ingredients:
            ing_obj, _ = Ingredient.objects.get_or_create(name=name)
            post.ingredients.add(ing_obj)

        post.save()

        return redirect('post_detail', id=post.id)

    return redirect('post_detail', id=post.id)
```

==========
=

ЧАСТИНА 2. Внутрішня частина вебзастосунку. Реалізація моделей (Python, Django framework)

Файл: socialmedia\userauth\models.py

```
from django.db import models
from django.contrib.auth.models import User
import uuid
from datetime import datetime



class Tag(models.Model):
    name = models.CharField(max_length=50, unique=True)

    def __str__(self):
        return self.name


class Ingredient(models.Model):
```

```
    name = models.CharField(max_length=100, unique=True)

    def __str__(self):
        return self.name


class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(blank=True, default='')
    profileimg = models.ImageField(upload_to='profile_images', default='blank-
profile-picture.png')
    location = models.CharField(max_length=100, blank=True, default='')
    language = models.CharField(max_length=10, default='en')
    background_color = models.CharField(max_length=20, default="#000000")
    tags = models.ManyToManyField(Tag, blank=True)

    def __str__(self):
        return self.user.username



class Post(models.Model):
    title = models.CharField(max_length=255, blank=True)
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    user = models.CharField(max_length=100)
    image = models.ImageField(upload_to='post_images')
    caption = models.TextField()
    created_at = models.DateTimeField(default=datetime.now)
    no_of_likes = models.IntegerField(default=0)

    tags = models.ManyToManyField(Tag, blank=True)
    ingredients = models.ManyToManyField(Ingredient, blank=True)

    cook_time = models.PositiveIntegerField(null=True, blank=True)

    def __str__(self):
        return self.title or self.user



class LikePost(models.Model):
    post_id = models.CharField(max_length=500)
```

```
    username = models.CharField(max_length=100)

    def __str__(self):
        return self.username

class Followers(models.Model):
    follower = models.CharField(max_length=100)
    user = models.CharField(max_length=100)

    def __str__(self):
        return self.user
```

==========
=

ЧАСТИНА 3. Внутрішня частина вебзастосунку. Налаштування маршрутизації (Python, Django framework)

Файл: socialmedia\userauth\urls.py

```
from django.contrib import admin
from django.urls import path
from django.conf.urls.static import static
from django.conf import settings
from userauth import views


urlpatterns = [
    path('', views.home, name='home'),
    path('signup/', views.signup, name='signup'),
    path('login/', views.loginn, name='login'),
    path('logout/', views.logoutt, name='logout'),
    path('upload/', views.upload, name='upload'),
    path('like-post/<str:id>', views.likes, name='like-post'),
    path('explore/', views.explore, name='explore'),
    path('profile/color/', views.update_background_color, name='update_color'),
    path('profile/<str:id_user>/', views.profile, name='profile'),
    path('follow', views.follow, name='follow'),
    path('delete/<str:id>', views.delete, name='delete'),
    path('search-results/', views.search_results, name='search_results'),
    path('settings/', views.settings, name='settings'),
    path('post/<uuid:id>/', views.post_detail, name='post_detail'),
    path('post/<uuid:id>/edit/', views.edit_post, name='edit_post'),
```

]

==========
=
ЧАСТИНА 4. Зовнішня частина вебзастосунку. Домашня сторінка (HTML, CSS, Jinja, Bootstrap)

Файл: socialmedia\templates\main.html

```
{% load static %}
{% load i18n %}

<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
   <script src="https://kit.fontawesome.com/cb792c0850.js"
crossorigin="anonymous"></script>
   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
      integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLA
SjC" crossorigin="anonymous">

   <title>Simple Cooking</title>
   <link rel="stylesheet" href="{% static 'css/app.css' %}">
   <style>
      .sidebar {
         background-color: #f8f9fa;
         height: 100vh;
         color: black;
      }

      .profile-pic {
         width: 60px;
```

```
            height: 60px;
            border-radius: 50%;
            object-fit: cover;
        }

        a {
            color: black;
        }
        .post-image {
            max-height: 500px;
            object-fit: cover;
            border-radius: 8px;
        }
    </style>
</head>

<body>
    <!-- this a modal code -->
<div class="container-fluid">
    <div class="row">
        <nav class="col-md-2 d-none d-md-block sidebar" style="position: fixed;">
            <div class="text-center mt-3">
                <img src="{{profile.profileimg.url}}" alt="Profile Picture" class="profile-
pic">
                <h4 class="mt-2"><a href="/profile/{{user}}" style="text-
decoration:none;">@{{user}}</a></h4>
            </div>
            <ul class="nav flex-column mt-4">
                <li class="nav-item">
                    <a class="nav-link active" href="{% url 'home' %}">
                        <i class="fa-solid fa-house mr-1"></i> {% trans "Home" %}
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'explore' %}">
                        <i class="fa-solid fa-compass mr-1"></i> {% trans "Explore" %}
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#" data-bs-toggle="modal" data-bs-
target="#exampleModal3">
```

```
                <i class="fa-solid fa-magnifying-glass mr-1"></i> {% trans "Search"
%}
              </a>
            </li>

            <li class="nav-item">
              <a class="nav-link" href="#" data-bs-toggle="modal" data-bs-
target="#exampleModal"
                data-bs-whatever="@getbootstrap">
                <i class="fa-solid fa-circle-plus mr-1"></i> {% trans "Create Post"
%}
              </a>


            </li>
            <li class="nav-item">
              <a class="nav-link" href="{% url 'profile' user %}">
                <i class="fa-solid fa-user mr-1"></i> {% trans "Profile" %}
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{% url 'settings' %}">
                <i class="fa-solid fa-gear mr-1"></i> {% trans "Settings" %}
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{% url 'logout' %}">
                <i class="fa-solid fa-right-from-bracket mr-1"></i> {% trans "Logout"
%}
              </a>
            </li>
          </ul>
        </nav>
        <!-- this is the main content code -->
        <main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-md-4">
          <!-- Your main content goes here -->

          {% include "modal.html" %}
          {% include "search.html" %}

        <!-- home posts code -->
```

```
        <div>{% for pos in post %}
            <div class="container" id="{{pos.id}}">
                <div class="row">
                    <div class="col-md-6 mx-auto">
                        <div class="post-card"style="border: 1px solid black;margin:
50px;">
                            <img src="{{pos.image.url}}" alt="Post Image" class="img-fluid
w-100 post-image">
                            <div class="post-content"style="margin-left: 19px;">
                                <h4>
                                    <a href="{% url 'post_detail' pos.id %}" style="text-
decoration:none;">
                                        {{ pos.title }}
                                    </a>
                                </h4>
                                <h5>
                                    {% if pos.profile %}
                                        <img src="{{ pos.profile.profileimg.url }}"
                                            style="width:30px;height:30px;border-
radius:50%;object-fit:cover;">
                                    {% endif %}
                                    <a href="/profile/{{pos.user}}" style="text-decoration:
none;">
                                        @{{pos.user}}
                                    </a>
                                </h5>
                                <p class="text-muted">
                                        {{ pos.cook_time }} {% trans "min" %}
                                </p>
                                <p class="caption-text">{{ pos.caption|linebreaks }}</p>
                                <div class="mt-2">
                                    {% for tag in pos.tags_list %}
                                        {% if tag %}
                                            <a href="{% url 'search_results' %}?q={{ tag }}"
class="badge bg-primary m-1">
                                                #{{ tag }}
                                            </a>
                                        {% endif %}
                                    {% endfor %}
                                </div>
                                <div class="mt-2">
```

```
                        {% for ingredient in pos.ingredients_list %}
                            {% if ingredient %}
                                <a href="{% url 'search_results' %}?q={{ ingredient
}}"
                                class="badge bg-success m-1">
                                    {{ ingredient }}
                                </a>
                            {% endif %}
                        {% endfor %}
                    </div>
                    <p class="text-muted">{{pos.created_at}}</p>
                    <hr>
                    <div class="d-flex justify-content-between" style="margin-
bottom: 8px; margin-right:10px;">
                        {% if pos.id %}
                        <a href="{% url 'like-post' pos.id %}#{{pos.id}}"
class="btn btn-outline-primary">Like</a>
                        {% else %}
                        <button class="btn btn-secondary" disabled>No
ID</button>
                        {% endif %}


                        {% if pos.no_of_likes == 0 %}
                        <p>0</p>
                        {% elif pos.no_of_likes == 1 %}
                        <p>Liked by {{pos.no_of_likes}} person</p>
                        {% else %}
                        <p>Liked by {{pos.no_of_likes}} people</p>
                        {% endif %}
                    </div>
                </div>
            </div>
        </div>
    </div>
    {% endfor %}

</div>
```

```
        </main>


    </div>
</div>

    <script src="{% static 'js/app.js'%}"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIax
VXM"
        crossorigin="anonymous"></script>
</body>

</html>
```