МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ДЛЯ КУЛІНАРІЇ З ІНТЕРАКТИВНИМ ПОМІЧНИКОМ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 402.22010326

Виконала студентка 4-го курсу, групи 402
_____ Н. М. Шевчук
«18» червня 2024 р.

Керівник: канд. фіз.-мат. наук, доцент
_____І. В. Кулаковська
«18» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студентці групи 402 факультету комп'ютерних наук Шевчук Надії Миколаївні.

1. Тема кваліфікаційної роботи «Вебзастосунок для кулінарії з інтерактивним помічником».

Керівник роботи Кулаковська Інесса Василівна, доцент кафедри, канд. фіз.-мат. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «18» червня 2024 р.

3. Вхідні (початкові) дані до роботи: база даних рецептів, продуктів та спецій.

Очікуваний результат: вебзастосунок для кулінарії з інтерактивним помічником.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- аналіз предметної сфери вебзастосунків для кулінарії. Постановка задачі;
- інформаційні технології для вирішення поставленої задачі. Підготовка до розробки;
- проектування та розробка вебзастосунку для кулінарфї з інтерактивним помічником;
- основні функції та інтерфейс вебзастосунку.

5. Перелік графічного матеріалу: презентація, 32 рисунків, 1 таблиця, 25 використаних джерел та 1 додаток.

6. Завдання до спеціальної частини: «Вимоги до приміщення з комп'ютерним обладнанням. Перевірочний розрахунок природного освітлення для приміщення»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __ доцент кафедри, канд. фіз.-мат. наук, доцент Кулаковська І. В.

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Шевчук Н. М._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок для кулінарії з інтерактивним помічником

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз сучасного стану вебзастосунків для кулінарії з інтерактивним помічником, огляд існуючих технологій, розробка ПЗ	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист КРБ перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	

Розробила студентка __Шевчук Н. М._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ доцент кафедри, канд. фіз.-мат. наук, доцент Кулаковська І. В.
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи студентки групи 402 ЧНУ ім. Петра Могили
Шевчук Надії Миколаївни
Тема: «Вебзастосунок для кулінарії з інтерактивним помічником»

Актуальність розробки визначається потребою у ефективному інструменті, який допомагав би користувачам у процесі приготування їжі, пропонуючи інтерактивну підтримку та рекомендації.

Об’єктом роботи є процес розробки вебзастосунку з інтерактивним помічником.

Предметом роботи є інтеграція штучного інтелекту у вебзастосунок для кулінарії.

Метою даної кваліфікаційної роботи є розробка вебзастосунку для кулінарії, який включає інтерактивного помічника, здатного надавати персоналізовані рекомендації та підтримку. Одним із завдань було розробити архітектуру застосунку, включаючи базу даних та інтеграцію з API OpenAI.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі розглядаються існуючі аналоги вебзастосунків для кулінарії, їх порівняльна характеристика і з’ясовано сучасний стан проблеми та основні методи її вирішення.

У другому розділі обрано інформаційні технології для вирішення поставленої задачі.

У третьому розділі детально розглянуто розроблену архітектуру та функціональність клієнтської та серверної частин системи керування рецептами, включаючи інтеграцію з сервісами для обробки зображень та текстових даних. Ключові аспекти включають розробку інтерактивних модулів чату та рецептів, які забезпечують динамічну взаємодію з користувачами та ефективне управління контентом.

У четвертому розділі представлено основні функції та інтерфейс для використання вебзастосунку для кулінарії з інтерактивним помічником.

У результаті розроблено кулінарний вебзастосунок з інтерактивним помічником, який здатний надавати персоналізовані рекомендації та підтримку. Зокрема, застосунок може розпізнавати страви за фотографіями, надавати рецепти та дозволяє зберігати створені рецепти до власного каталогу за категоріями.

Кваліфікаційна робота містить 79 сторінок, 32 рисунків, 1 таблиця, 25 використаних джерел та 1 додаток.

Ключові слова: вебзастосунок, штучний інтелект, інтерактивний помічник, чат-бот, кулінарія, рецепти.

ABSTRACT

Qualification work of a student of group 402 at Petro Mohyla Black Sea National University

Shevchuk Nadiia

on topic: "Web application for cooking with an interactive assistant "

The relevance of the development is determined by the need for an effective tool that would help users in the process of cooking by offering interactive support and recommendations.

The object of the work is the process of developing a web application with an interactive assistant.

The subject of the work is the integration of artificial intelligence into a cooking web application.

The objective of this thesis is to develop a cooking web application that includes an interactive assistant capable of providing personalized recommendations and support. One of the tasks was to develop the application architecture, including the database and integration with the OpenAI API.

The explanatory note consists of an introduction, four chapters, conclusions and appendices.

The first chapter examines the existing analogs of web applications for cooking, their comparative characteristics, and the current state of the problem and the main methods of solving it are clarified.

In the second section, information technologies are chosen to solve the given problem.

In the third section, the developed architecture and front-end and back-end functionality of the recipe management system, including integration with image and text data processing services, are detailed. Key aspects include the development of interactive chat modules and recipes that provide dynamic user interaction and effective content management.

The fourth chapter presents the main functions and interface for using the cooking web application with an interactive assistant.

The result is a culinary web application with an interactive assistant capable of providing personalized recommendations and support. In particular, the application can recognize dishes from photos, provide recipes and allows you to save created recipes to your own catalog by category.

The qualification work contains 79 pages, 32 figures, 1 table, 25 used sources and 1 appendice.

Key words: web application, artificial intelligence, interactive assistant, chatbot, cooking, recipes.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВММ	–	великі мовні моделі
ПЗ	–	програмне забезпечення
СКБД	–	система управління базами даних

AI	–	Artificial Intelligence
API	–	Application Programming Interface
CLIP	–	Contrastive Language–Image Pre-training
DTO	–	Data Transfer Objects
ER	–	Entity-Relationship
GPT	–	Generative Pre-trained Transformer
HTTPS	–	HyperText Transfer Protocol Secure
ID	–	Identifier
JS	–	JavaScript
JWT	–	JSON Web Token
ORM	–	Object-Relational Mapping

ВСТУП

У сучасному світі кулінарія є не лише необхідністю, але й популярним хобі для мільйонів людей. З розвитком інформаційних технологій кулінарія знайшла своє місце у цифровому просторі, де численні вебзастосунки і платформи допомагають людям готувати смачні страви, ділитися рецептами та вдосконалювати свої кулінарні навички. Актуальність розробки вебзастосунку для кулінарії з інтерактивним помічником визначається потребою у зручному і доступному інструменті, який допомагав би користувачам у процесі приготування їжі, пропонуючи інтерактивну підтримку та рекомендації.

Історично кулінарні рецепти передавались від покоління до покоління у вигляді записів у кулінарних книгах або усних переказів. Однак, із появою інтернету та мобільних технологій, рецепти та кулінарні поради стали більш доступними. Вебзастосунки для кулінарії пройшли еволюцію від простих онлайн-колекцій рецептів до складних інтерактивних платформ, що пропонують персоналізовані рекомендації, відеоуроки та інші корисні функції. Інтерактивний помічник, базований на штучному інтелекті, є логічним наступним кроком у цій еволюції, оскільки він може надати користувачам ще більш індивідуалізовану допомогу та підказки в режимі реального часу.

Розробка вебзастосунку для кулінарії з інтерактивним помічником є важливою не тільки з точки зору технічного прогресу, але й з огляду на соціальні та культурні аспекти. Такий застосунок може допомогти людям з різними рівнями кулінарних навичок – від новачків до досвідчених кухарів, сприяти популяризації здорового харчування, а також зберігати і передавати кулінарні традиції різних культур. Інтерактивний помічник, який використовує технології штучного інтелекту, зокрема моделі OpenAI, може забезпечити користувачам високий рівень зручності та підтримки під час приготування їжі.

Об’єктом роботи є процес розробки вебзастосунку з інтерактивним помічником.

Предметом роботи є інтеграція штучного інтелекту у вебзастосунок для

кулінарії.

Метою даної кваліфікаційної роботи є розробка вебзастосунку для кулінарії, який включає інтерактивного помічника, здатного надавати персоналізовані рекомендації та підтримку.

Для досягнення мети розробки вебзастосунку для кулінарії з інтерактивним помічником необхідно вирішити такі завдання:

– провести аналіз існуючих вебзастосунків та визначити їхні сильні та слабкі сторони;

– обрати відповідні технології для розробки клієнтської та серверної частин вебзастосунку;

– розробити архітектуру, включаючи базу даних та інтеграцію з API OpenAI;

– реалізувати основні функції вебзастосунку, включаючи реєстрацію користувачів, створення та перегляд рецептів, інтерактивний помічник.

У роботі буде детально розглянуто вибір технологій, процес розробки та функціональність застосунку, що зробить його корисним інструментом для широкого кола користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ВЕБЗАСТОСУНКІВ ДЛЯ КУЛІНАРІЇ. ПОСТАНОВКА ЗАДАЧІ

1.1 Основні поняття

Вебзастосунок - це програмне забезпечення, розроблене для роботи через веббраузер, яке надає користувачам доступ до різних функцій та можливостей через Інтернет. Вебзастосунок зазвичай має інтуїтивний інтерфейс, що дозволяє користувачам легко навігувати та використовувати його, а також може бути доступним на різних платформах, включаючи комп'ютери, смартфони та планшети. Це забезпечує зручний та доступний спосіб для користувачів.

Вебзастосунок з кулінарії - це вебзастосунок, спрямований на задоволення потреб користувачів у сфері кулінарії, такий як пошук рецептів, отримання рекомендацій щодо приготування страв тощо.

Інтерактивний помічник - це функціонал вебзастосунку, який взаємодіє з користувачем, надаючи рекомендації, відповіді на запитання та іншу допомогу на основі введених даних.

Чат-бот - це програма, яка автоматично спілкується з користувачем через текстові повідомлення, надаючи йому різноманітну інформацію або виконуючи завдання. В контексті вебзастосунку для кулінарії, чат-бот може діяти як інтерактивний помічник, який надає рекомендації з приготування страв, відповідає на запитання користувачів та надає іншу допомогу пов'язану з кулінарією. Він може використовувати штучний інтелект для аналізу уподобань та обмежень користувача, а також взаємодіяти з користувачами через тексти, фотографії або інші мультимедійні елементи. Чат-бот може бути інтегрований безпосередньо в вебзастосунок, надаючи користувачам зручний спосіб отримання інформації та послуг.

Штучний інтелект - це галузь комп'ютерних наук, яка займається створенням систем, що можуть виконувати завдання, які зазвичай вимагають інтелектуальної людської діяльності. Ці системи використовують алгоритми та

моделі, що дозволяють їм навчатися на основі даних, робити прогнози, приймати рішення та виконувати завдання у різних галузях, включаючи розпізнавання образів, обробку мовлення, планування та прийняття рішень. У вебзастосунках, штучний інтелект може використовуватися для різноманітних цілей, включаючи відповіді на запитання користувачів, рекомендації, персоналізацію вмісту, аналіз даних користувачів та інші.

1.2 Існуючі аналоги та порівняльна характеристика.

У сучасному цифровому світі вебзастосунки з кулінарією стали невід'ємною частиною життя та дозволяють користувачам легко знаходити та експериментувати з новими рецептами, зберігати улюблені страви та ділитися ними з іншими. Однак, перед тим як розробляти власний вебзастосунок, важливо дослідити існуючі аналоги та виявити їхні переваги та недоліки.

Аналіз існуючих аналогів є ключовим етапом у процесі розробки будь-якого нового продукту чи сервісу. Цей етап дозволяє розробникам зрозуміти, що вже існує на ринку, які функції користувачі вважають важливими та що може бути вдосконалено. Крім того, порівняльний аналіз існуючих аналогів надає можливість виявити ніші або прогалини на ринку, які можуть бути заповнені новим проектом.

Запропонований підрозділ розглядає декілька існуючих вебзастосунків у сфері кулінарії, проаналізовуючи їхні основні функції та характеристики і визначаючи, як вони впливають на наше розуміння ринкових потреб та можливостей. В процесі планування роботи було обрано низку найвідоміших в Україні вебресурсів з кулінарії для подальшого порівняльного аналізу. Ці сайти відомі своєю популярністю серед користувачів, вони часто згадуються в соціальних мережах, форумах та кулінарних спільнотах. Вибір цих сайтів не базувався на рейтингах чи оглядах, а був здійснений на основі їхньої широкої відомості та активного зацікавлення аудиторії. Кожен з обраних сайтів представляє різноманітність підходів до представлення рецептів, дизайну та

користувацького досвіду, що робить їх цікавими об'єктами для подальшого аналізу та порівняння.

Розглянемо перший сайт з кулінарії klopotenko.com [18], він має наступний вигляд, який наведено на рис. 1.1.

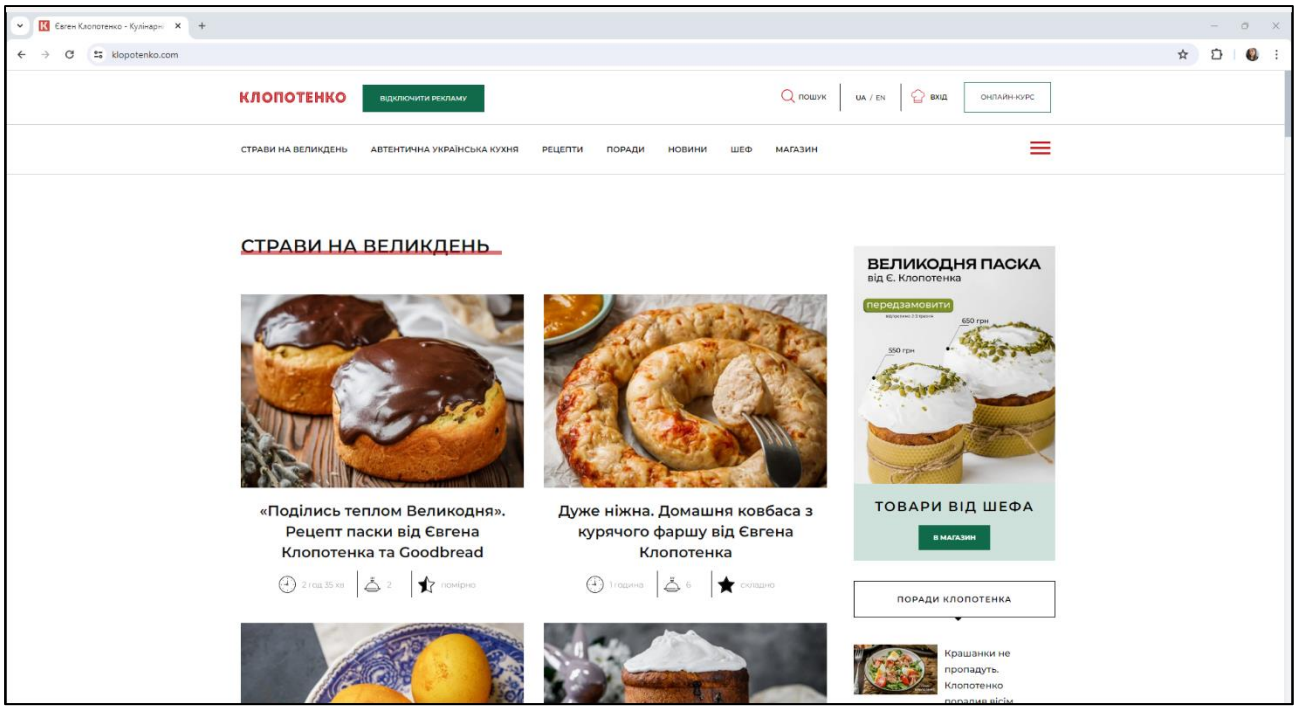

Рисунок 1.1 – Знімок головної сторінки сайту klopotenko.com

Відкривши даний сайт користувач має змогу відразу переглядати його вміст без реєстрації чи авторизації - ці процеси, пов’язані з ідентифікацією користувача, наявні, але не обов’язкові. Сайт має функцію пошуку, яка дозволяє легко знаходити рецепти за назвою страви, замість того, щоб шукати рецепти вручну. Але це також зробити не важко, тому що у шапці даного сайту наведено список з посиланнями на інші сторінки сайту з відповідними підтемами заголовків. Ось наприклад, натискаючи на «Рецепти» вибиває список з наявними категоріями та підкатегоріми, які можна обрати, як наведено на рис. 1.2, що є дуже зручним для користувача в пошуку відповідного рецепту страви.

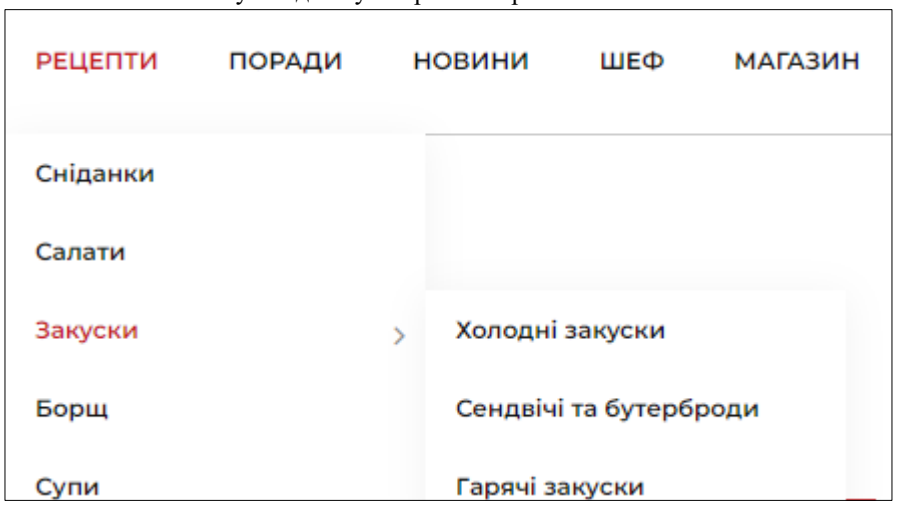

Рисунок 1.2 – Фрагмент категорій рецептів на сайті klopotenko.com

В даному сайті наявна велика кількість різноманітних страв і кожна з них розподілена по притаманним їм категоріям (наприклад, рецепт сирників можна знайти в категорії сніданків). Кулінарні рецепти представлені виключною формою, яка вражає своєю естетикою та простотою використання. Кожен рецепт відображається в зручній формі, де на перший погляд видно основні характеристики страви. Це включає в себе зручний інтерфейс, деталізовану інформацію про час приготування, кількість порцій та рівень складності, як наведено на рис. 1.3.

Рисунок 1.3 – Фрагмент додаткової інформації щодо рецептів на сайті klopotenko.com

Кожен рецепт супроводжується списком необхідних інгредієнтів, докладним описом кроків приготування, один як приклад наведено на рис. 1.4, які легко слідувати, та ілюстраціями з фотографіями, що дозволяють візуально уявити процес та результат. Ці фотографії додатково мотивують користувачів спробувати приготувати страву та гарантують відповідність вигляду готового блюда з

очікуваннями. Завдяки такому детальному та естетичному оформленню кулінарних рецептів, відвідувачі сайту отримують не лише доступ до смачних і різноманітних страв, але й велику порцію натхнення для кулінарних експериментів.

Рисунок 1.4 – Фрагмент покрового рецепту на сайті klopotenko.com

Після кожного рецепту наявні коментарі і відгуки з оцінками інших користувачів, але шоб мати змогу написати їх самому – необхідно зареєструватись або авторизуватись, якщо вже наявний створений акаунт на даному сайті. Наявність відгуків, які можна побачити на рис. 1.5, є значимим елементом застосунку, тому що вони дозволяють іншим користувачам оцінити якість та смак страв, випробуваних іншими людьми. Відгуки можуть надати цінну інформацію щодо того, як рецепт вдався та які можливі модифікації можна внести для поліпшення результату.

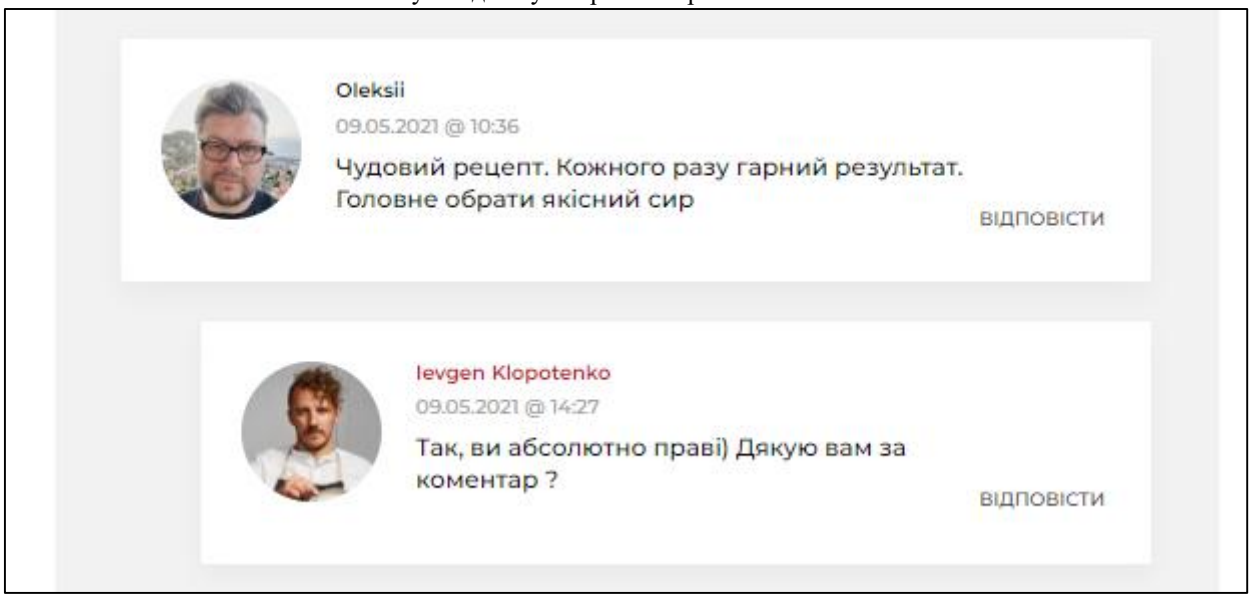

Рисунок 1.5 – Фрагмент відгуків до рецепту на сайті klopotenko.com

Наявність можливості залишати відгуки та коментарі стимулює користувачів до більш активної участі в житті вебресурсу. Вони починають спілкуватись між собою, ділитися враженнями та рецептами, що збагачує контент сайту та робить його більш привабливим для нових користувачів.

Сайт klopotenko.com - це не лише майданчик з кулінарними рецептами, але й ціла система, яка пропонує різноманітний контент для своїх користувачів. Поряд з рецептами тут можна знайти новини та корисні поради з кулінарії, що робить сайт цікавим інформаційним ресурсом. Окрім цього, на сайті присутній інтерактивний магазин, де користувачі можуть зробити замовлення на різні товари. Це може бути все - від предметів побуту до смаколиків, наприклад, паска від Євгена Клопотенка. Крім того, сайт пропонує курси з приготування різноманітної їжі, що дозволяє користувачам розвиватися в кулінарній сфері та поглиблювати свої знання та навички. Такий різноманітний контент робить сайт багатогранним та привабливим для широкого кола аудиторії.

На сайті klopotenko.com можна виявити кілька недоліків.

1. Перш за все, обмежений пошук за інгредієнтами може ускладнити користувачам знаходження рецептів за конкретними складниками, що може бути не зручно для тих, хто шукає страви для використання наявних продуктів у

холодильнику.

2. Крім того, незважаючи на те, що сайт пропонує рецепти без необхідності реєстрації, для залишення відгуків про рецепти потрібно все ж ввести свої дані. Це може бути невигідно для користувачів, які хочуть лише швидко залишити відгук без створення облікового запису на сайті.

3. Ще одним недоліком сайту klopotenko.com є відсутність можливості користувачам додавати власні рецепти. Це обмеження ускладнює співпрацю зі спільнотою та позбавляє користувачів можливості поділитися своїми кулінарними шедеврами з іншими. Така можливість може зробити сайт більш інтерактивним та зацікавити більше людей, що бажають ділитися своїм досвідом та рецептами зі спільнотою.

Наступним аналогом є вебсайт patelnya.com.ua [19] - це український вебресурс з кулінарним контентом, який пропонує широкий вибір рецептів, порад та статей з кулінарії. Головна сторінка наведена на рис. 1.6.

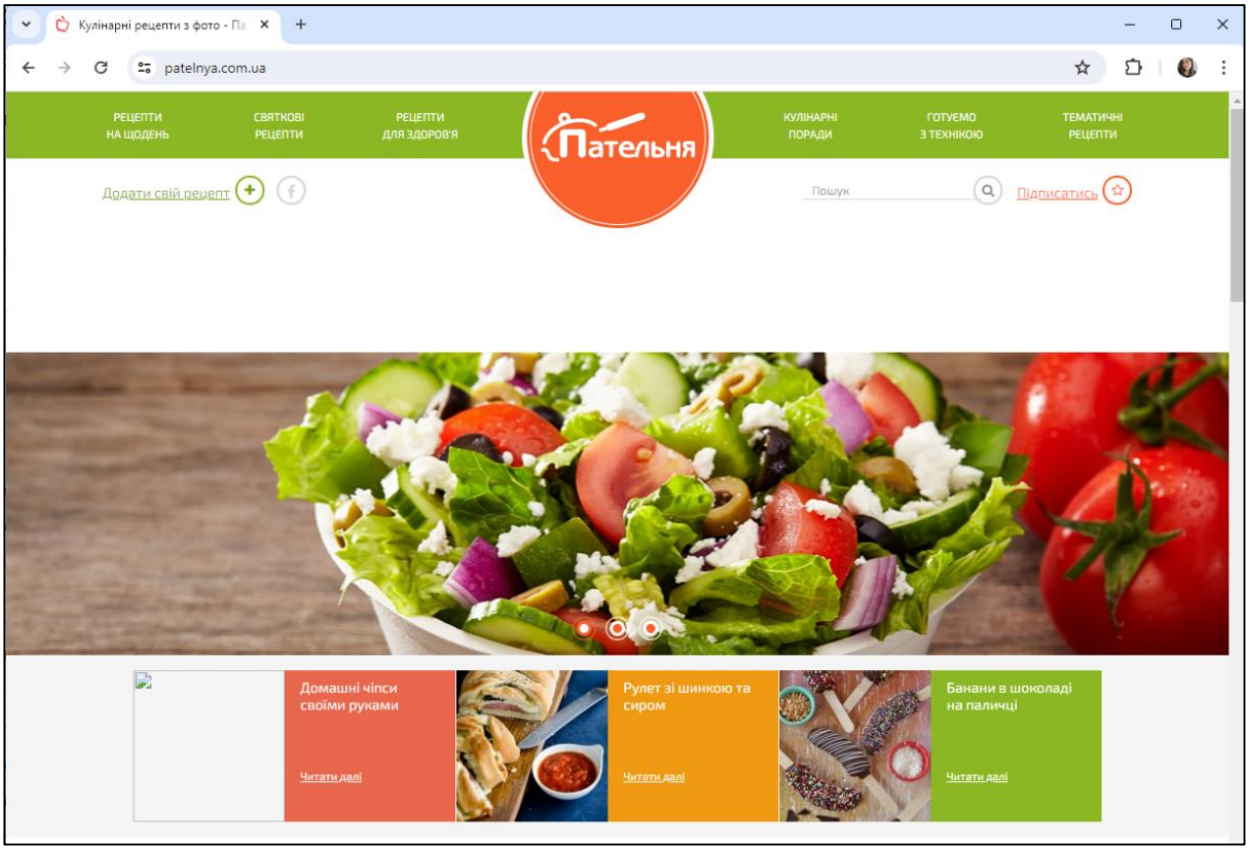

Рисунок 1.6 – Фрагмент головної сторінки сайту patelnya.com.ua

Сайт відомий своєю великою базою рецептів різних кухонь світу, включаючи українську, європейську, азіатську та інші. Кожен рецепт , як і на сайті Клопотенка, зазвичай супроводжується докладним описом приготування, списком інгредієнтів та фотографіями, що дозволяє користувачам легко розібратися в процесі готування.

На сайті Patelnya.com.ua можна знайти не лише багато цікавих рецептів, але й можливість швидкого пошуку страв. Особливістю є те, що користувач може шукати страви за конкретними інгредієнтами, як зображено на рис. 1.7. Проте, мінусом є обмеження - пошук здійснюється тільки за одним інгредієнтом, а не за декількома, що дещо ускладнює процес пошуку для тих, хто шукає страву за декількома специфічними складниками.

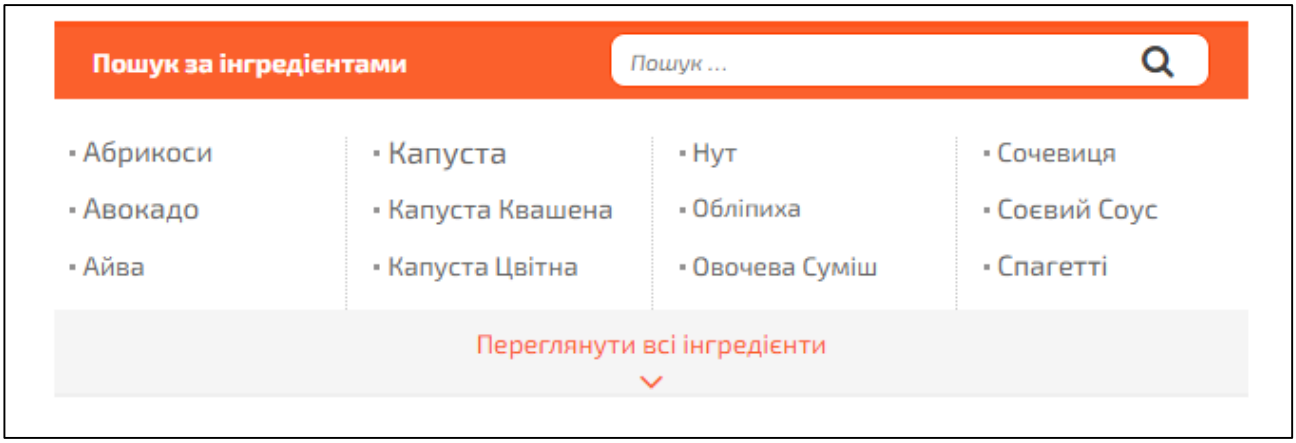

Рисунок 1.7 – Фрагмент пошуку за інгредієнтами на сайті patelnya.com.ua

Варто відзначити, що на сайті не потрібна авторизація чи реєстрація, щоб користуватися його функціоналом. Проте, для того, щоб залишити відгук про рецепт, потрібно ввести свої дані. Це дозволяє забезпечити якість та достовірність відгуків, але, одночасно, не ускладнює процес використання сайту для тих, хто просто шукає рецепти.

Рецепти на сайті також структуровані за категоріями, що спрощує навігацію та пошук необхідної інформації. Однак, серед переваг є особлива категорія "готуємо з технікою", де представлені рецепти, призначені для приготування

страв з використанням певної кулінарної техніки, такої як мультиварка, пароварка чи хлібопічка. Це дає можливість знаходити оптимальні рішення для приготування страв із застосуванням наявної кулінарної техніки та зручно впроваджувати їх у повсякденне кулінарне життя.

Також є ще відмінність від попереднього сайту - на вебресурсі "Patelnya.com.ua" користувачі мають можливість не лише переглядати рецепти, але й додавати власні рецепти страв на сайт. Для розміщення рецепту наявна відповідна форма для заповнення, наведена на рис. 1.8, але сам опис страви буде розміщено на сайті тільки після модерації адміністратором.

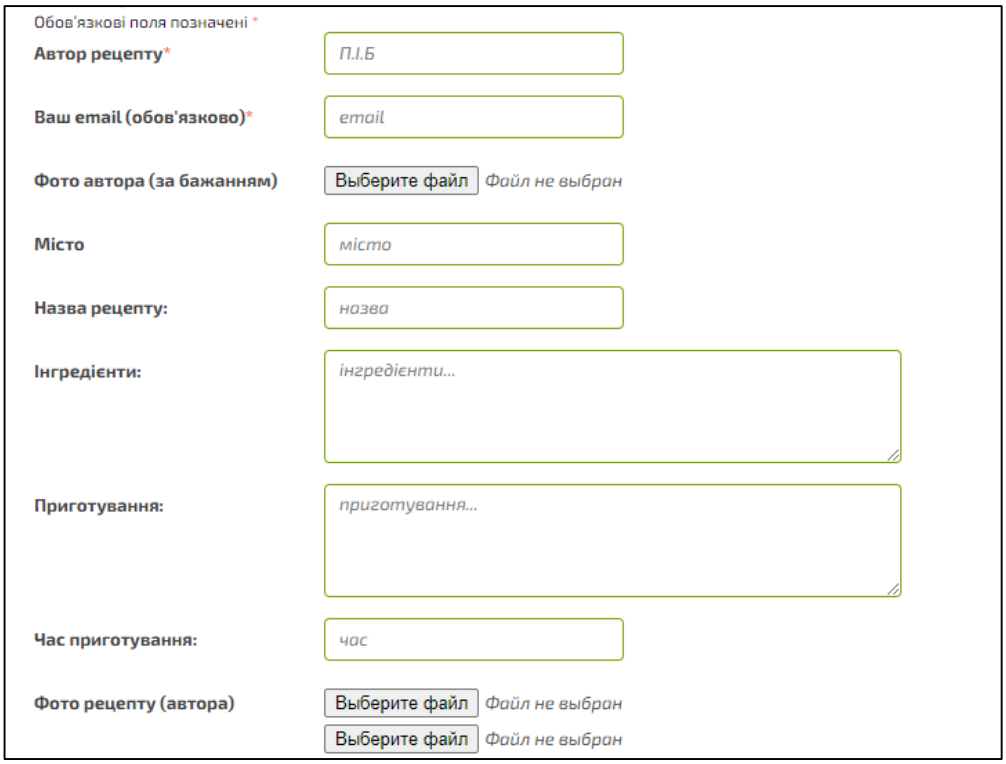

Рисунок 1.8 – Фрагмент форми додавання рецепту на сайті patelnya.com.ua

Обов’язковою умовою публікації рецепту на сайті є наявність зробленого фото готової страви. Також за бажанням можна надсилати покрокові фото та свої знімки з готовою стравою. Ця особливість робить сайт більш інтерактивним та залучає користувачів до активної участі. Кожен охочий може поділитися своїм улюбленим рецептом з іншими користувачами, розширюючи базу кулінарних

знань та надаючи можливість спільноті експериментувати з різноманіттям страв. Такий підхід сприяє взаємному обміну досвідом та творчому розвитку кулінарної спільноти.

Однак, наявність модерації контенту перед публікацією може призводити до затримок у відображенні нових рецептів на сайті, що може обмежувати швидкість оновлення контенту та реакції на запити користувачів.

1.3 Сучасний стан проблеми та основні методи її вирішення

У сучасному світі, де інтернет і соціальні мережі переповнені фотографіями їжі, користувачі часто зіштовхуються з проблемою знаходження рецептів для страв, які вони бачать на зображеннях, але не знають їхньої назви. Ця ситуація породжує потребу в інноваційних інструментах для пошуку та отримання рецептів. Звичайні вебзастосунки для кулінарії, які існують на сьогоднішній день, переважно пропонують лише можливість пошуку рецептів за їхніми назвами або ж інгредієнтами. Вони не мають функції для розпізнавання страв за фотографіями, що створює перешкоди для користувачів, які хочуть знайти рецепт конкретної страви, але не знають її назви. Таким чином, існує потреба у вебзастосунку, який пропонує інноваційну функцію розпізнавання страв за фотографіями і надання відповідних рекомендацій користувачам.

Також у сучасних сайтах з кулінарним контентом відсутня можливість отримати миттєву підтримку та консультації щодо рецептів. Користувачам не доступний механізм задавання певних питань та отримання відповідей безпосередньо в момент їхнього запиту. Зазвичай, їм доводиться чекати на відповідь через коментарі чи відгуки, а цей процес може займати час від годин до декількох днів, залежно від активності інших користувачів або власника сайту. У вебзастосунку, який я пропоную розробити, буде вбудована функція чат-бота, що і є інтерактивним помічником, яка надає користувачам можливість задавати будь-які питання щодо страв і отримувати миттєві відповіді з рекомендаціями та необхідною інформацією, такою як кількість калорій, складність приготування та

будь-що інше.

Одним з ключових рішень для вирішення цієї проблеми є розробка вебзастосунку з чат-ботом, який вміє надавати персоналізовану підтримку та консультацію користувачам щодо приготування страв. Цей чат-бот використовує штучний інтелект для аналізу фотографій страв та автоматичного розпізнавання їх, надаючи користувачам відповідні рекомендації.

Потенційні користувачі цього вебзастосунку можуть включати тих, хто активно використовує соціальні мережі для пошуку мотивації в кулінарії, а також тих, хто шукає зручний спосіб зберігати та використовувати рецепти у повсякденному житті. Результати розробки цього вебзастосунку можуть стати значним кроком у полегшенні процесу приготування їжі та наданні користувачам доступу до широкого асортименту кулінарних рецептів за допомогою сучасних технологій.

1.4 Формулювання завдання

У сучасних сайтах з кулінарним контентом існує відчутна відсутність можливості отримати негайну підтримку та консультації щодо рецептів. Це створює значні перешкоди для користувачів, які шукають негайну відповідь на свої запитання. Замість цього, користувачам доводиться чекати на відповіді через коментарі чи відгуки, що може займати час від годин до декількох днів, в залежності від активності інших користувачів або власника сайту.

У зв’язку з цим, виникає необхідність у створенні вебзастосунку, який надає користувачам миттєвий доступ до підтримки та консультацій з питань приготування страв. Одним з ключових елементів такого вебзастосунку буде вбудований чат-бот, який дозволить користувачам задавати будь-які питання щодо рецептів та отримувати негайні відповіді з рекомендаціями та необхідною інформацією, такою як кількість калорій, складність приготування та будь-якою іншою стосовно страви. Такий чат-бот буде надійним джерелом інформації для користувачів, які шукають експрес-допомогу та індивідуальні консультації у

процесі готування страв.

У вебзастосунку, який буде розроблено, однією з ключових функцій буде можливість розпізнавання страв за фотографіями. Це важлива інноваційна можливість, яка значно полегшить користувачам пошук рецептів і підвищить зручність використання сайту.

Через інтегровану систему штучного інтелекту, додаток здатний автоматично розпізнавати страву на зображенні, навіть якщо користувач не знає її назви. Коли користувач завантажує фотографію страви у додаток, система аналізує зображення і встановлює її ідентичність. Після цього, користувачеві автоматично надаються рекомендації щодо приготування цієї страви, а також інша необхідна інформація, така як список інгредієнтів, кількість калорій, складність приготування та інше. Ця функція дозволяє користувачам легко знаходити рецепти для страв, які вони бачать на зображеннях, але не знають їх назви. Вона стає особливо корисною в умовах сучасного світу, де соціальні мережі переповнені фотографіями їжі без вказання назви. Такий підхід значно спрощує пошук рецептів та зробить процес готування приємнішим та доступнішим для користувачів.

Також у розробленому вебзастосунку користувачі матимуть можливість зберігати сформовані рецепти з чат-боту безпосередньо у додатку, самостійно генеруючи зображення, для подальшого використання. Ця функція дозволить зручно організувати та зберігати улюблені рецепти та меню для майбутнього використання.

Висновки до розділу 1

У розділі «Аналіз предметної сфери вебзастосунків для кулінарії. Постановка задачі» результатом стало ретельне порівняння існуючих аналогів вебзастосунків для кулінарії, включаючи їхню порівняльну характеристику з урахуванням функціональності, ефективності та користувацького досвіду. Було вивчено сучасний стан проблеми в області онлайн-сервісів для кулінарії та

розглянуто основні методи її вирішення. На основі аналізу встановлено, що існуючі вебзастосунки мають недостатню персоналізацію та інтерактивність, що зумовило необхідність створення нового рішення.

Формулювання завдання полягало в розробці та впровадженні інноваційного вебзастосунку для кулінарії з інтерактивним помічником, здатним надавати користувачеві персоналізовані рекомендації щодо приготування страв та забезпечувати можливість зберігати отримані рецепти за окремими категоріями. Крім того, вебзастосунок матиме функціональність, яка дозволятиме користувачам завантажувати фотографії страв у чат, щоб помічник міг автоматично розпізнати їх.

2 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ. ПІДГОТОВКА ДО РОЗРОБКИ

2.1 Моделі OpenAI

Оскільки вебзастосунок для кулінарії включає інтерактивного помічника, цей інтерактивний помічник буде реалізований як чат-бот зі штучним інтелектом, побудованим на технологіях OpenAI. Тому далі розглянемо моделі OpenAI, які будуть використовуватися для створення цього чат-бота, а також пояснимо, як вони працюють, щоб забезпечити ефективну і корисну взаємодію з користувачами.

OpenAI – це провідна дослідницька компанія в галузі штучного інтелекту, яка створює передові моделі машинного навчання для різних застосувань. Одним із основних продуктів OpenAI є мовна модель GPT (Generative Pre-trained Transformer), яка здатна генерувати текст на основі введених даних, відповідаючи на запитання, створюючи описи, генеруючи код тощо [21].

Для створення чат-ботів, які можуть обробляти текст і розпізнавати фото з повідомлень, часто використовують комбінацію кількох моделей. Відповідні моделі OpenAI для цього завдання включають:

1) GPT (Generative Pre-trained Transformer) - для обробки тексту. GPT-3 або GPT-4 можуть бути використані для генерації відповідей на текстові повідомлення, аналізу тексту та інших завдань, пов'язаних з обробкою природної мови [26];

2) CLIP (Contrastive Language–Image Pre-training) - для розпізнавання фото. CLIP може обробляти текстові запити та зіставляти їх з відповідними зображеннями, що дозволяє чат-боту розуміти вміст зображень, надісланих у повідомленнях [17].

2.1.1 Алгоритм роботи великих мовних моделей

Великі мовні моделі працюють, використовуючи методи глибокого

навчання та великі обсяги текстових даних. Ці моделі зазвичай базуються на архітектурі трансформера, наприклад, на генеративному попередньо навченому трансформері, який відмінно справляється з послідовними даними, такими як введення тексту [9].

Модель складається з декількох шарів нейронних мереж, кожен з яких має параметри, що можуть бути точно налаштовані під час навчання і які додатково посилюються за допомогою декількох шарів, відомих як «механізм уваги», що фокусується на певних частинах наборів даних.

Як працюють нейронні мережі?

Ми задаємо вхідні дані, а на виході отримуємо коефіцієнти. Уявимо, що нам треба, щоб наша нейронка розпізнавала що написано в 3×3 пікселях, що наведено на рис. 2.1. У нас буде 1 вхідний рівень, 2 приховані рівні та 1 вихідний.

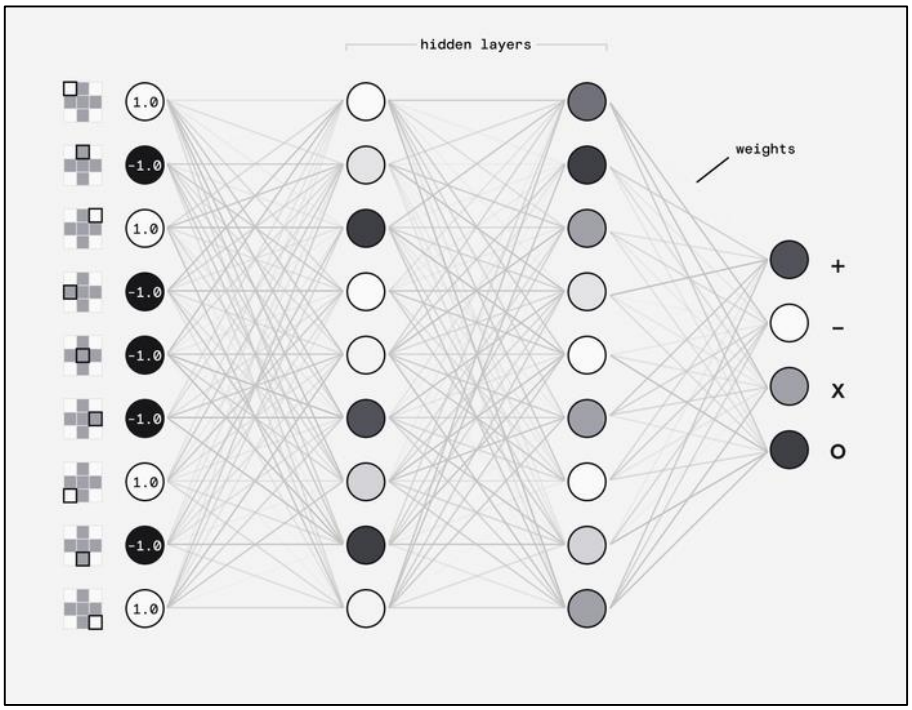

Рисунок 2.1 – Як працюють нейронні мережі [26]

На вході у нас 9 нейронів, які приймають значення від -1 до 1, а на виході у нас 4 нейрони, які матимуть значення імовірності від 0 до 1. В прихованих рівнях нейрони пов’язані та впливають на результат наступного рівня за допомогою ваг

(може бути від -1 до 1). Як правильно розставити ваги? Ну тут ми займаємося навчанням нейронки. Даємо картинку 3×3 дивимося результат, та намагаємося знайти такі ваги, при яких буде найточніший результат. На етапі тренування ми маємо вхідні дані та очікуваний результат. Повторюємо це багато разів і шукаємо той випадок, який дає найкращий результат, відносно очікуваного результату.

У процесі навчання Великі мовні моделі намагаються прогнозувати наступне слово в реченні на основі контексту, наданого попередніми словами. Це досягається шляхом присвоєння ймовірнісних оцінок словам, які були «токенізовані» – розбиті на менші послідовності символів. Потім ці токени перетворюються на вставки, які є числовим відображенням цього контексту .

Вхідний текст розбивається на токени, якими можуть бути слова, знаки, символи та інші частини мови. Усі ці токени зберігаються в багатовимірному векторі, де групуються між собою. Нижче на рис. 2.2, для спрощення, побачимо це на 2-ох вимірній площині.

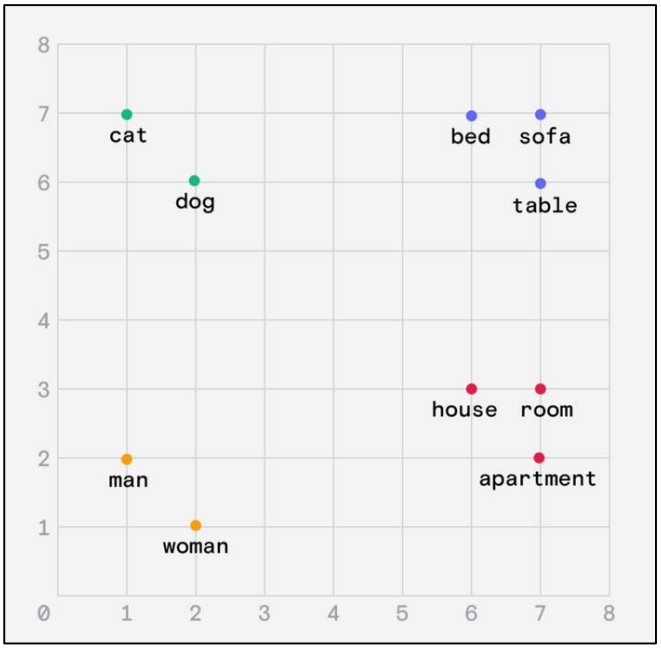

Рисунок 2.2 – Мовні моделі, 2-х вимірна площина [26]

Тобто, якщо нам треба додати слово “цуценя”, то воно буде поруч із “собака” та “кіт“ Мовні моделі намагаються вгадати наступне слово, зважаючи на

контекст, але в той же час, відповідь буде кожного разу різна. Модель не шукає лише те слово, що найбільше підходить, але й бере до уваги на випадковість, адже саме вона дає найкращий результат. Тут така собі функція random().

Для забезпечення точності цей процес передбачає тренування ВММ на великих масивах тексту (в мільярдах сторінок), що дозволяє вчити граматику, семантику та концептуальні відносини через навчання з нульовим ударом та самонавчання. Після цього Великі мовні моделі можуть генерувати текст, автономно прогнозуючи наступне слово на основі вводу, який вони отримують, і використовуючи шаблони та знання.

Результатом є зв’язне та контекстуально відповідне генерування мови, яке можна використовувати для широкого спектра завдань з розуміння природної мови та генерування контенту.

Великі мовні моделі розроблені для того, щоб розуміти та генерувати текст, подібний до людського, а також інші форми контенту, на основі величезної кількості даних, які використовуються для їх навчання. Вони можуть:

– робити висновки з контексту;
– генерувати зв’язні та контекстуально відповідні відповіді;
– перекладати на мови, відмінні від англійської;
– резюмувати текст;
– відповідати на питання (загальна бесіда та часті питання).

Великі мовні моделі можуть робити це завдяки мільярдам параметрів, які дозволяють їм уловлювати складні закономірності мови та виконувати широкий спектр мовних завдань. ВММ здійснюють революцію у додатках у різних галузях: від чат-ботів та віртуальних помічників до створення контенту, допомоги у дослідженнях та мовного перекладу [25-26].

Продовжуючи розвиватися та вдосконалюватися, ВММ готові змінити способи нашої взаємодії з технологіями та доступом до інформації, зробивши їх ключовою частиною сучасного цифрового ландшафту.

2.1.2 Алгоритм роботи CLIP для розпізнавання фото

Модель CLIP (Contrastive Language–Image Pre-training) розроблена OpenAI для одночасного навчання на текстових і візуальних даних. Її основна мета – зіставлення зображень та текстів у спільному векторному просторі, що дозволяє ефективно знаходити відповідності між ними.

Процес роботи CLIP починається з підготовки даних, що складаються з пар зображення і підпису (текстового опису зображення). Модель використовує дві нейронні мережі: одну для обробки зображень (зазвичай це варіант ResNet або Vision Transformer), а іншу для обробки текстів (зазвичай це варіант трансформера, який приймає на вхід один тип послідовностей, а на виході вертає інший, перетворенний по певному алгоритму), як показано на рис. 2.3. Ці мережі перетворюють зображення і тексти у векторні подання, тобто в набори чисел, що відображають основні характеристики вхідних даних.

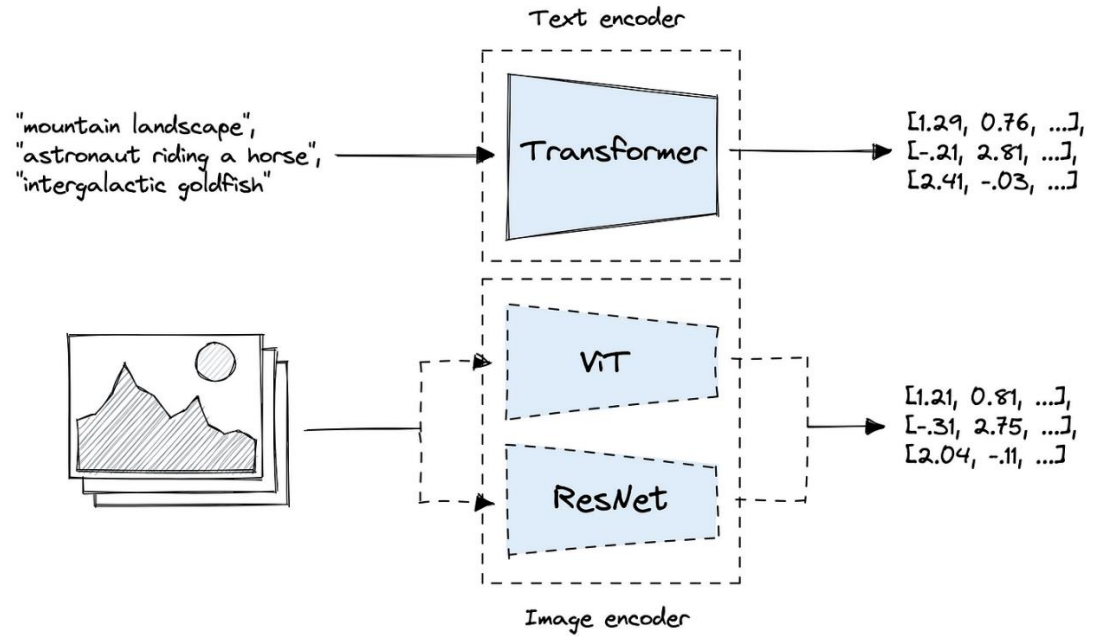

Рисунок 2.3 – Алгоритм роботи CLIP [17]

Під час тренування CLIP одночасно обробляє партії пар "зображення-текст". Кожне зображення і кожен текст перетворюються відповідними мережами

у вектори. Потім модель використовує спеціальний підхід, який називається контрастивним навчанням. Ідея полягає в тому, щоб зменшити відстань у векторному просторі між вектором зображення і вектором його відповідного тексту, а також збільшити відстань між векторами невідповідних пар. Це досягається шляхом обчислення і мінімізації спеціальної функції втрат (loss function), яка спрямована на максимізацію подібності між векторами відповідних пар і мінімізацію подібності між векторами невідповідних пар.

У результаті тренування CLIP отримує два узгоджені векторні простори: один для зображень і один для текстів. Це означає, що відповідні зображення і тексти будуть знаходитися близько один до одного в цьому спільному векторному просторі, що дозволяє легко знаходити відповідності між ними. Після завершення тренування модель може використовуватися для різних задач, таких як пошук зображень за текстовим запитом або генерування текстового опису для зображення. Модель CLIP показує високу ефективність у таких завданнях завдяки своїй здатності навчатися на великих обсягах даних і створювати узгоджені векторні простори для різнорідних даних [17].

2.2 Вибір мови програмування та середовища виконання

У процесі розробки вебзастосунку важливо вибрати відповідні технології, які забезпечать оптимальну продуктивність, гнучкість та зручність підтримки проекту.

Вибір мови програмування є важливим етапом перед безпосередньою розробкою програмного забезпечення. Тому необхідно проаналізувати декілька, щоб зробити кращий вибір.

Для створення вебзастосунку було розглянуто кілька популярних мов програмування, таких як JavaScript, Python та PHP. Кожна з цих мов має свої переваги та недоліки, які варто враховувати при виборі.

Аналіз мов програмування наведено нижче.

JavaScript.

Переваги:

1) завдяки підтримці асинхронного програмування, JavaScript дозволяє створювати високопродуктивні вебзастосунки;

2) динамічна типізація полегшує розробку та робить код гнучким;

3) екосистема – велика кількість бібліотек і фреймворків, таких як React, Angular, Vue.js для фронтенду та Express для бекенду;

4) кросплатформеність – JavaScript підтримується всіма сучасними веббраузерами, що робить його ідеальним вибором для фронтенд-розробки;

5) широке застосування: як на стороні клієнта (фронтенд), так і на стороні сервера (бекенд) за допомогою Node.js.

Недоліки:

1) JavaScript-код може бути вразливим до атак, таких як XSS (cross-site scripting), якщо не вжити належних заходів безпеки;

2) інтенсивні обчислення можуть призвести до проблем з продуктивністю, хоча це можна частково вирішити за допомогою WebAssembly.

Python.

Переваги:

1) читабельний синтаксис, що спрощує розробку;

2) величезна кількість бібліотек і фреймворків, таких як Django та Flask для веброзробки;

3) використовується для різних задач, включаючи веброзробку, аналіз даних, машинне навчання.

Недоліки:

1) в деяких випадках, продуктивність Python нижча у порівнянні з мовами, такими як JavaScript (Node.js) або Java;

2) Python не може працювати безпосередньо в браузері, що робить його менш зручним для фронтенд-розробки.

PHP.

Переваги:

1) популярна мова для серверної частини вебзастосунків;
2) простий синтаксис та велика кількість готових рішень.

Недоліки:

1) PHP відомий своїми історичними проблемами з безпекою, хоча сучасні версії значно покращилися;
2) іноді складніше підтримувати великі проекти у порівнянні з сучаснішими мовами та фреймворками.

Для розробки вебзастосунку було обрано JavaScript з наступних причин.

По перше, використання JavaScript дозволяє створювати як клієнтську, так і серверну частину вебзастосунку, що зменшує необхідність у знанні декількох мов програмування.

По-друге, JavaScript забезпечує високу інтерактивність та динамічність вебзастосунків завдяки таким технологіям, як AJAX та WebSockets.

Також наявність великої кількості фреймворків (React, Angular, Vue.js) та бібліотек значно прискорює розробку та підвищує якість кінцевого продукту.

І також велика кількість ресурсів, документації та спільнот допомагають швидко вирішувати виникаючі проблеми та отримувати нові знання.

Для розробки нашого кулінарного вебзастосунку з інтерактивним помічником обрано Visual Studio Code (VS Code).

Visual Studio Code – це безкоштовний редактор коду, розроблений компанією Microsoft. Він є кросплатформним (працює на Windows, macOS та Linux) і пропонує розширені функціональні можливості, які роблять його зручним та потужним інструментом для розробників. Основні переваги VS Code включають:

– підтримка різних мов програмування;
– розширення та налаштування;
– інтеграція з системами керування версіями;
– дебагінг;
– інтелектуальне автозаповнення та рефакторинг.

VS Code є потужним та зручним інструментом для розробки нашого веб-застосунку, забезпечуючи комфортну роботу з кодом, налагодженням та керуванням проектом.

2.3 Вибір системи керування базою даних

Для розробки додатку «Вебзастосунок для кулінарії з інтерактивним помічником» необхідно вибрати систему керування базою даних (СКБД), яка забезпечує зберігання та доступ до даних. Розглянемо декілька можливих варіантів та порівняємо їх особливості.

MySQL, розроблена компанією Oracle Corporation, є однією з найпопулярніших відкритих реляційних систем управління базами даних (СКБД). Вона відома своєю швидкодією та простотою використання, що робить її популярним вибором для веб-додатків та невеликих підприємств. MySQL підтримує розширення мови SQL, а також має широкий спектр функцій, таких як транзакції, індексація та забезпечення безпеки даних.

Microsoft SQL Server є однією з провідних комерційних СКБД, розроблених компанією Microsoft. Вона надає розширені можливості для обробки та аналізу даних, таких як управління бізнес-інтелектом та інтеграція з іншими продуктами Microsoft, такими як Excel та Power BI. SQL Server підтримує різні рівні ізоляції транзакцій, реплікацію даних та інструменти аналізу для підтримки потужних корпоративних додатків.

PostgreSQL є відкритою реляційною СКБД, яка відома своєю потужністю та розширюваністю. Вона надає багато функцій, включаючи складні типи даних, тригери, роботу з геоданими та повнотекстовий пошук. PostgreSQL активно підтримується великою спільнотою розробників і користувачів, що робить його популярним вибором для проектів з великим обсягом даних та вимогами до безпеки.

У таблиці 2.1 проведено порівняння розглянутих СКБД.

Таблиця 2.1 – Порівняльний аналіз СКБД

Характеристика	**MySQL**	**Microsoft SQL Server**	**PostgreSQL**
Мови програмування	Багато	Багато	Багато
Відкритий код	Так	Ні	Так
Функціональність	Дуже розширена	Середня	Розширена
Вартість	Комерційний / безкоштовно	Комерційний	Безкоштовно
Масштабованість	Добра	Дуже добра	Добра
Аналіз даних	Середній	Дуже розширений	Розширений
Продуктивність	Добра	Дуже добра	Добра

Для розробки додатку «Вебзастосунок для кулінарії з інтерактивним помічником» було вибрано PostgreSQL із наступних причин:

1) є відкритим ПЗ, що означає, що можна вільно використовувати, змінювати та розповсюджувати його без додаткових витрат. Це робить PostgreSQL привабливим вибором для проектів з обмеженим бюджетом або тих, хто хоче уникнути залежності від комерційних постачальників ПЗ;

2) відома своєю багатою функціональністю, яка включає підтримку складних типів даних, індекси, повнотекстовий пошук, геодані (PostGIS), та багато іншого. Це робить її потужним інструментом для різноманітних сценаріїв використання, від невеликих веб-додатків до великих аналітичних систем;

3) не потребує комерційної ліцензії, що забезпечує суттєву економію коштів у порівнянні з комерційними рішеннями, такими як Microsoft SQL Server. Ця економія може бути значною, особливо для стартапів або компаній, які прагнуть зменшити свої операційні витрати;

4) надає розширені можливості для аналізу даних, включаючи складні

запити, транзакції, агрегати, функції вікон і аналітичні функції. Це дозволяє ефективно виконувати складні аналітичні задачі без необхідності в додаткових інструментах;

5) демонструє добру продуктивність і масштабованість, що дозволяє їй ефективно працювати з великими обсягами даних і високими навантаженнями. Її архітектура дозволяє легко масштабувати систему, забезпечуючи стабільну роботу навіть під значним навантаженням.

Також для нашого застосунку можна використати Redis в контексті хеш бази даних, в якій будуть зберігатись повідомлення з чату.

Redis (Remote Dictionary Server) – це високопродуктивна, відкрита база даних типу key-value, яка зберігає дані в пам'яті (in-memory). Вона підтримує широкий набір структур даних, таких як рядки, списки, множини, хеші, геопросторові індекси та багато інших.

Основні особливості Redis:

– висока швидкість читання та запису;

– підтримка різноманітних структур даних (від простого кешування до складних обчислень);

– постійність даних;

– реплікація даних, що дозволяє створювати резервні копії;

– скриптування;

– підтримує механізм публікації та підписки (pub/sub), що дозволяє створювати системи обміну повідомленнями.

Redis часто використовується для кешування даних, управління сеансами, аналітики в реальному часі, систем черг повідомлень та інших сценаріїв, де потрібна швидка обробка великих обсягів даних. Завдяки своїй гнучкості, швидкості та надійності, Redis став популярним вибором для розробників у багатьох галузях.

2.4 Інструменти серверної частини

Серверна частина (бекенд) вебзастосунку відповідає за обробку даних, логіку бізнес-процесів, зберігання інформації та взаємодію з базами даних та зовнішніми сервісами. Вона приймає запити від клієнтів, обробляє їх і надсилає відповіді, управляє даними в базах даних, забезпечує аутентифікацію та авторизацію користувачів, інтегрується з іншими сервісами та забезпечує логування та моніторинг системи. Бекенд гарантує безпеку, масштабованість та ефективну роботу вебзастосунку, дозволяючи реалізувати складні бізнес-логіки і функціональність.

2.4.1 Node.js

Node.js - це середовище виконання JavaScript, яке дозволяє запускати JavaScript-код на сервері. Node.js побудовано на движку V8, який використовує Google Chrome, і забезпечує ефективне виконання коду завдяки асинхронній моделі вводу/виводу, заснованій на подіях [1]. Це робить Node.js ідеальним для створення швидких та масштабованих мережевих застосунків.

Основні характеристики Node.js: подійно-орієнтована архітектура (неблокуючу модель введення/виведення (I/O), що дозволяє обробляти багато запитів одночасно без блокування потоку виконання); єдиний потік; пакетний менеджер npm (Node Package Manager) - є вбудованим менеджером пакетів для Node.js (надає доступ до тисяч бібліотек і модулів, що спрощує розробку застосунків).

Express.js – основний інструмент для Node.js

Express.js (або просто Express) – це мінімалістичний і гнучкий веб-фреймворк для Node.js, який надає безліч потужних функцій для веб- і мобільних застосунків. Він дозволяє швидко і легко створювати серверні застосунки та API.

Для веб-застосунку для кулінарії з інтерактивним помічником, Express.js буде основним інструментом для створення серверної частини [7].

Він дозволить створити HTTP-сервер для обробки запитів від клієнтів

(користувачів) і відповідей на ці запити. Надає потужні засоби для маршрутизації запитів. Це дозволяє легко налаштовувати різні маршрути для обробки запитів на різні URL-адреси, такі як /recipes для отримання списку рецептів або /chat для доступу до чату. Використовуючи middleware, можна обробляти дані запитів (наприклад, дані форм або JSON-об'єкти), перевіряти автентичність користувачів, обробляти помилки та виконувати інші функції.

Express.js можна легко інтегрувати з ORM або іншими бібліотеками для роботи з базами даних, такими як PostgreSQL (що і було обрано). Це дозволить зберігати, отримувати та обробляти дані рецептів, користувачів та чатів.

Express.js є відмінним інструментом для створення RESTful API, що дозволить клієнтській частині застосунку (написаній на React) взаємодіяти з сервером для отримання та відправки даних.

Переваги використання Node.js:

– Node.js використовує движок V8, який забезпечує високу продуктивність виконання JavaScript-коду;

– неблокуюча модель I/O дозволяє обробляти безліч запитів одночасно, що робить Node.js ефективним для побудови масштабованих мережевих застосунків;

– використання JavaScript як на клієнтській, так і на серверній частині дозволяє спростити розробку та зменшити потребу в переключенні між різними мовами програмування;

– велика кількість доступних пакетів і модулів у npm дозволяє швидко додавати нові функціональності до застосунку;

– розробникам, знайомим з JavaScript, буде легко перейти до роботи з Node.js, оскільки не потрібно вивчати нову мову програмування.

Node.js є потужним інструментом для серверної розробки, який поєднує високу продуктивність, гнучкість та простоту у використанні. Використання Node.js дозволяє створювати швидкі, масштабовані та ефективні веб-застосунки, що відповідають сучасним вимогам.

2.4.2 NestJS

NestJS – це прогресивний фреймворк для побудови ефективних, масштабованих серверних додатків на Node.js. Він використовує TypeScript як основну мову програмування і створений на принципах об'єктно-орієнтованого програмування, функціонального програмування та функціонально-реактивного програмування [5].

Як NestJS може бути використаний у вебзастосунку?

Для веб-застосунку для кулінарії з інтерактивним помічником, NestJS може забезпечити потужний і гнучкий серверний фреймворк. Ось як він може бути інтегрований у проект.

По-перше, NestJS надає зручний спосіб створення HTTP-сервера та визначення маршрутів для обробки запитів. Можна створити контролери для управління різними частинами додатку, такими як маршрути для рецептів, чату і користувачів.

По-друге, використовуючи сервіси, можна впровадити бізнес-логіку застосунку, таку як обробка даних користувачів, управління рецептами, або обробка повідомлень в чаті.

Також модульна архітектура NestJS дозволяє розділити додаток на логічні частини [16], кожна з яких може бути незалежно розроблена та протестована. Наприклад, можна створити окремі модулі для автентифікації, управління користувачами, рецептами та чатом.

І на останок, NestJS дозволяє легко інтегрувати застосунок з іншими сервісами та бібліотеками, такими як OpenAI для інтерактивного помічника. Це дозволяє створювати потужні інтерактивні функції, такі як чат-бот, який може надавати рекомендації з рецептів.

2.4.3 Застосування OpenAI

У кулінарному веб-застосунку з інтерактивним помічником (чат-ботом) користувачі взаємодіють з чат-ботом, а він, у свою чергу, взаємодіє з API OpenAI.

Це забезпечує багатофункціональну та інтерактивну роботу з користувачами. Чат-бот виконує роль "прокладки" або менеджера фільтрації між користувачем і сервісом OpenAI, створюючи контекст і структуру відповідей. Основною функціональністю чат-бота є формування запитів до OpenAI та обробка отриманих відповідей.

Основні аспекти інтеграції наведено нижче.

1. Обробка запитів користувачів:

– користувач вводить запит у чат, який надсилається на сервер;

– чат-бот на сервері формує запит до API OpenAI, включаючи повідомлення, історію повідомлень, опціональні зображення та інструкції.

2. Взаємодія з OpenAI API:

– після обробки запиту сервер надсилає його до API OpenAI;

– API OpenAI генерує відповідь на основі переданого контексту і повертає її на сервер.

3. Фільтрація і форматування відповідей:

– сервер отримує відповідь від OpenAI, аналізує її та при необхідності фільтрує або змінює формат;

– оброблена відповідь надсилається користувачеві через чат, , де відбувається подальше взаємодія з користувачем.

Це важливо, тому що інтеграція через "прокладку" дозволяє контролювати якість і зміст відповідей, забезпечуючи відповідність політикам безпеки та якості. Менеджмент контексту дозволяє забезпечити більш релевантні та точні відповіді, утримуючи інформацію про попередні повідомлення користувача. Така архітектура дозволяє легко масштабувати систему, обробляючи більше запитів за допомогою розподілу навантаження між серверами. Також цей підхід дозволяє забезпечити ефективну та інтерактивну комунікацію з користувачем через чат-бота, використовуючи потужні можливості штучного інтелекту від OpenAI [11, 12, 14].

2.4.4 Prisma

Prisma – це сучасний інструмент для роботи з базами даних, який спрощує процес взаємодії з базою даних у додатках, написаних на JavaScript/TypeScript. Він поєднує ORM (Object-Relational Mapping) і генерацію типів для роботи з базами даних, що дозволяє зручно та безпечно працювати з даними [13].

Основні компоненти Prisma:

1) Prisma Client – автоматично згенерований клієнт для доступу до бази даних;

2) Prisma Migrate – інструмент для управління міграціями бази даних;

3) Prisma Schema – описує структуру бази даних у форматі декларативної конфігурації.

Структура бази даних описується у файлі schema.prisma. У цьому файлі визначаються моделі, що відповідають таблицям у базі даних, та їхні взаємозв'язки. Наприклад, модель User має зв'язок один-до-багатьох з моделлю Recipe (користувач може мати багато рецептів), а модель Recipe має зв'язок багато-до-одного з моделлю User.

Міграції – це спосіб керування змінами у структурі бази даних з часом. Вони дозволяють безпечно вносити зміни до схеми бази даних і підтримувати її синхронізованою з кодом.

Створення міграцій: коли вносяться зміни до schema.prisma, потрібно створити нову міграцію. Це робиться за допомогою команди Prisma CLI.

Ця команда:

– порівнює поточний стан схеми з попередніми міграціями;
– створює файл міграції з SQL-запитами, необхідними для внесення змін;
– застосовує міграцію до бази даних.

Після внесення змін до схеми і міграцій потрібно згенерувати новий клієнт Prisma, який враховуватиме ці зміни і який використовуватиметься у коді для взаємодії з базою даних [15].

Причини використання Prisma наведено нижче.

1. Prisma автоматично генерує типи TypeScript на основі схеми, що забезпечує безпеку типів і знижує кількість помилок під час розробки.

2. Prisma надає інтуїтивний і простий у використанні API для роботи з базою даних, що дозволяє швидко і ефективно виконувати CRUD-операції (create, read, update, delete).

3. Використовуючи міграції, Prisma забезпечує, що структура бази даних завжди синхронізована з кодом.

4. Prisma підтримує різні бази даних, включаючи PostgreSQL, який буде використано.

2.5 Інструменти клієнтської частини

Клієнтська частина, або фронтенд, відповідає за те, як інформація відображається та взаємодіє з користувачем на веб-сайтах, мобільних додатках та інших інтерфейсах. Вона включає в себе структуру, дизайн та функціональність, доступні користувачам. Фронтенд-розробники використовують HTML, CSS та JavaScript для створення зовнішнього вигляду та взаємодії елементів інтерфейсу. Основна мета фронтенду - забезпечити користувачам зручний та привабливий інтерфейс, що дозволить їм легко користуватися програмним забезпеченням чи веб-сайтом [20]. У фронтенді також зазвичай реалізується валідація введених даних, анімації та інші інтерактивні ефекти, що покращують взаємодію з користувачем.

2.5.1 React

React – це популярна бібліотека для побудови користувацьких інтерфейсів, розроблена компанією Facebook. Вона дозволяє створювати динамічні і швидкі веб-додатки завдяки своїй компонентній архітектурі. Компоненти в React дозволяють розробникам створювати користувацький інтерфейс шляхом побудови невеликих, ізольованих шматків коду, званих компонентами. Кожен компонент може мати власний стан і логіку, що робить код модульним та легким

для повторного використання. Логіка компонента включає декілька ключових аспектів, що визначають його функціональність [3, 10].

1. Завантаження даних на сторінку. Компоненти можуть бути відповідальні за отримання даних з сервера або іншого джерела. Це може включати відправку HTTP-запитів для отримання інформації, такої як користувацькі дані, списки товарів або інші необхідні ресурси. Наприклад, використання хуків, таких як useEffect, дозволяє виконувати побічні ефекти, наприклад, завантаження даних при монтажі компонента.

2. Робота з даними. Після отримання даних компонент може виконувати різні операції з ними, наприклад, фільтрацію, сортування або обробку даних для відображення. Це включає маніпуляцію станом компонента, використовуючи хуки, такі як useState, для зберігання та оновлення стану даних.

3. Зміна стану сторінки. Компоненти React можуть змінювати свій стан у відповідь на взаємодії користувача або інші події. Наприклад, натискання кнопки може змінити відображення певної частини інтерфейсу або оновити дані на сторінці. Хуки, такі як useState, дозволяють керувати станом компонента та оновлювати його у відповідь на події [10].

4. Модифікація кнопок. Логіка компонента може включати динамічну зміну властивостей кнопок та інших елементів інтерфейсу. Це може включати зміну тексту, стилів або стану кнопок у відповідь на дії користувача або зміни даних. Компоненти можуть використовувати стан для контролю умовного відображення та поведінки елементів.

5. Відображення вікон. Компоненти можуть включати логіку для відображення модальних вікон або інших спливаючих елементів. Це може бути реалізовано шляхом умовного рендерингу компонентів на основі стану або пропсів, що керують видимістю цих елементів.

6. Відправка форм. Компоненти часто відповідають за обробку введення користувача та відправку форм. Це включає збір даних з полів форми, валідацію введених даних та відправку запитів на сервер. Логіка обробки форм може бути

реалізована за допомогою подій, таких як onSubmit, та стану для зберігання введених даних.

Усі ці дії складають так звану "логіку компонента". Важливо розділяти цю логіку на окремі функції та хуки, щоб зберігати код чистим, організованим та легко підтримуваним. Використання хуків дозволяє ізолювати бізнес-логіку від коду, що відповідає за відображення інтерфейсу, що підвищує модульність та повторне використання компонентів. В компоненті є код, який описує HTML та CSS, необхідні для відображення інтерфейсу. Цей код записується у вигляді JSX (JavaScript XML), що дозволяє писати HTML-подібний код прямо в JavaScript. JSX використовується для визначення структури компоненту, включаючи розмітку та стилі. Це дозволяє легко створювати ієрархії елементів та забезпечує високу читабельність коду.

2.5.2 JWT токени

JWT (JSON Web Token) - це стандарт, що використовується для створення токенів доступу, які дозволяють користувачам отримувати доступ до ресурсів після успішної аутентифікації. Основна мета JWT - забезпечити безпеку інформації, яка передається між клієнтом і сервером [2].

JWT токен зазвичай складається з трьох частин: заголовка, корисної навантаження (payload) та підпису. Заголовок містить метадані про токен, наприклад, тип і метод шифрування. Корисна навантаження містить дані, які можуть бути використані для ідентифікації користувача або надання доступу до ресурсів. Підпис забезпечує перевірку цілісності токену і підтверджує, що він не був змінений.

JWT використовується для аутентифікації користувачів із використанням рефреш токенів для поновлення доступу після закінчення строку дії основного токену. Рефреш токен дає можливість користувачеві отримати новий JWT токен без необхідності повторної аутентифікації.

Інформація, збережена в JWT токені, може бути зашифрована для

забезпечення конфіденційності. Токени зазвичай зберігаються в куках (cookies), щоб забезпечити безпеку та уникнути можливості XSS атак.

Термін життя JWT токенів може бути налаштований на сервері і залежить від конкретних вимог додатка. Після закінчення строку дії токену, користувач повинен повторно аутентифікуватись для отримання нового токену.

Інтеграція JWT токенів на фронтенді передбачає виконання логіки для збереження, відправки та оновлення токенів при взаємодії з сервером. Використання JWT токенів спрощує процес аутентифікації та авторизації користувачів та забезпечує безпеку передачі даних між клієнтом і сервером [8].

2.5.3 Emotion

Emotion.js – це популярна бібліотека для CSS-in-JS у JavaScript та TypeScript. Вона дозволяє стилізувати компоненти, використовуючи CSS, безпосередньо у JavaScript або TypeScript коді, забезпечуючи гнучкість та динамічність стилізації. Emotion.js пропонує потужний та ефективний підхід до управління стилями в сучасних вебзастосунках.

У кулінарному вебзастосунку Emotion.js використовуватиметься для стильового оформлення компонентів. Це дозволяє нам зберігати стилі в одному місці разом із компонентами, що полегшує управління кодом. Застосуємо динамічні стилі, які змінюються в залежності від стану компонентів або пропсів. Використовуючи Emotion.js з TypeScript, ми забезпечимо типізований і безпечний код, що дозволяє швидко виявляти помилки і підвищує продуктивність розробки. Emotion.js також дозволяє управляти глобальними стилями додатка, що забезпечує єдину стильову систему для всіх компонентів і сторінок [6].

Висновки до розділу 2

У даному розділі, який має назву «Інформаційні технології для вирішення поставленої задачі. Підготовка до розробки» було детально розглянуто і обрано ключові технології, які будуть використовуватись для реалізації нашого

вебзастосунку для кулінарії з інтерактивним помічником. Досліджено моделі OpenAI, які забезпечать інтерактивного помічника на основі штучного інтелекту. Для розробки було обрано мову програмування JavaScript, яка буде використовуватися в обох частинах проекту –як на клієнтській, так і на серверній стороні. Середовище розробки Visual Studio Code було обрано за його зручність і багатий набір інструментів для розробників. Як систему керування базами даних було вибрано PostgreSQL через її потужність, масштабованість та розширені можливості для роботи з даними.

На клієнтській стороні використовуватимо React для створення динамічного і зручного інтерфейсу, JWT токени для аутентифікації та Emotion для стилізації компонентів. На серверній стороні застосовуватимуться Node.js та Express для створення серверної логіки, NestJS для структурування і організації коду, а також Prisma для роботи з базою даних. Технології OpenAI будуть інтегровані для забезпечення функціональності чат-бота. Усі ці технології разом забезпечать надійність, масштабованість та ефективність нашого вебзастосунку, відповідаючи в майбтньому вимогам та очікуванням користувачів.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ КУЛІНАРІЇ З ІНТЕРАКТИВНИМ ПОМІЧНИКОМ

3.1 Архітектура

3.1.1 Організація мультирепозиторія

Для кращої організації кодової бази вирішено об’єднати усі 3 застосунки в одному репозиторії. Мультирепозиторій може надати кілька ключових переваг порівняно з окремими репозиторіями для кожного застосунку. Ось деякі з основних переваг.

1. Спільне використання коду, мультирепозиторії легше спільно використовувати код між різними застосунками.

2. Керування версіями і релізами в єдиному репозиторії дозволяє легше синхронізувати випуски між проектами. Всі проекти можуть використовувати однакові версії залежних бібліотек, що зменшує ризики несумісності та помилки в роботі застосунків.

3. Один репозиторій може використовувати спільні налаштування для розгортання, тестування та інших процесів DevOps. Це зменшує кількість скриптів та конфігурацій, які потрібно створювати та підтримувати.

4. Управління кількома проектами в одному репозиторії дозволяє централізувати задачі, що спрощує трекінг проблем та розвиток проектів. Командам легше координувати зусилля і спільно працювати над завданнями.

5. Централізоване управління проектами може допомогти оптимізувати використання серверних та розробницьких ресурсів, оскільки команда використовує спільну інфраструктуру для розробки, тестування та розгортання.

6. Корінь мультирепозиторію містить файли конфігурацій лінтера і тайпскрипта та залежності усіх застосунків.

3.1.2 Проектування бази даних

Під час проектування бази даних для предметної області «Вебзастосунок для кулінарії з інтерактивним помічником» було застосовано метод «сутність-зв’язок» (ER метод), який передбачає вирішення завдань на нижчих рівнях перед тим, як розв’язувати завдання більш високого рівня. На основі цього підходу було розроблено логічну модель з використанням ER-діаграми, в якій було виділено наступні сутності та атрибути:

– користувач: унікальний ідентифікатор користувача, дата та час створення запису користувача, електронна адреса користувача (унікальна), пароль користувача, ім'я користувача, прізвище користувача, прапорець, який вказує, чи активований користувач, зв'язок з рецептами, які створив користувач;

– підтвердження користувача: ідентифікатор підтвердження користувача, дата та час закінчення терміну дії коду підтвердження, код підтвердження, новий пароль користувача, нова електронна адреса користувача, унікальний ідентифікатор користувача, з яким пов'язаний запис підтвердження, зв'язок з користувачем;

– рецепт: унікальний ідентифікатор рецепту, дата та час створення рецепту, дата та час останнього оновлення рецепту, назва рецепту, опис рецепту, унікальний ідентифікатор користувача, який створив рецепт, унікальний ідентифікатор чату, пов'язаного з рецептом, унікальний ідентифікатор категорії, до якої належить рецепт, зв'язок з користувачем, зв'язок з чатом, зв'язок з категорією;

– категорія: унікальний ідентифікатор категорії, дата та час створення категорії, дата та час останнього оновлення категорії, назва категорії, зв'язок з рецептами, що входять до категорії;

– чат: унікальний ідентифікатор чату, дата та час створення чату, дата та час останнього оновлення чату, назва чату, кількість повідомлень у чаті, унікальний ідентифікатор користувача, що створив чат, зв'язок з рецептами, що входять до чату.

У процесі проектування бази даних для «Вебзастосунку для кулінарії з інтерактивним помічником» було встановлено зв'язки між сутностями. Зв'язки встановлювалися між полями однакових типів і полями, які мають зв'язок між собою. За допомогою зовнішнього ключа однієї таблиці та внутрішнього ключа іншої встановлювалися зв'язки між таблицями. У логічній моделі бази даних для предметної області «Вебзастосунок для кулінарії з інтерактивним помічником» були встановлені зв'язки "один до одного", "один до багатьох", «один до нуля або багатьох», «один до одного або багатьох». Визначення типів зв'язків допомогло коректно побудувати зв'язки між таблицями, що в свою чергу забезпечить правильну роботу функціоналу бази даних.

Організовано такі зв'язки між таблицями:

– "User" (поле "id") – "UserConfirmation" (поле "userId"), вид зв'язку 1:1;
– "User" (поле "id") – "Recipe" (поле "userId"), вид зв'язку 1:0..N;
– "User" (поле "id") – "Chat" (поле "userId"), вид зв'язку 1:0..N;
– "Recipe" (поле "id") – "Category" (поле "categoryId"), вид зв'язку 1..N:1;
– "Chat" (поле "id") – "Recipe" (поле "chatId"), вид зв'язку 1:0..N.

ER-діаграма є важливим етапом проектування бази даних, оскільки дозволяє відображати зв'язки між сутностями та їх атрибутами. В результаті було створено ER-діаграму для бази даних, яка наглядно відображає взаємозв'язки між таблицями та їх полями на рис. 3.1. Ця модель є важливою при подальшому проектуванні та розробці функціоналу додатку.

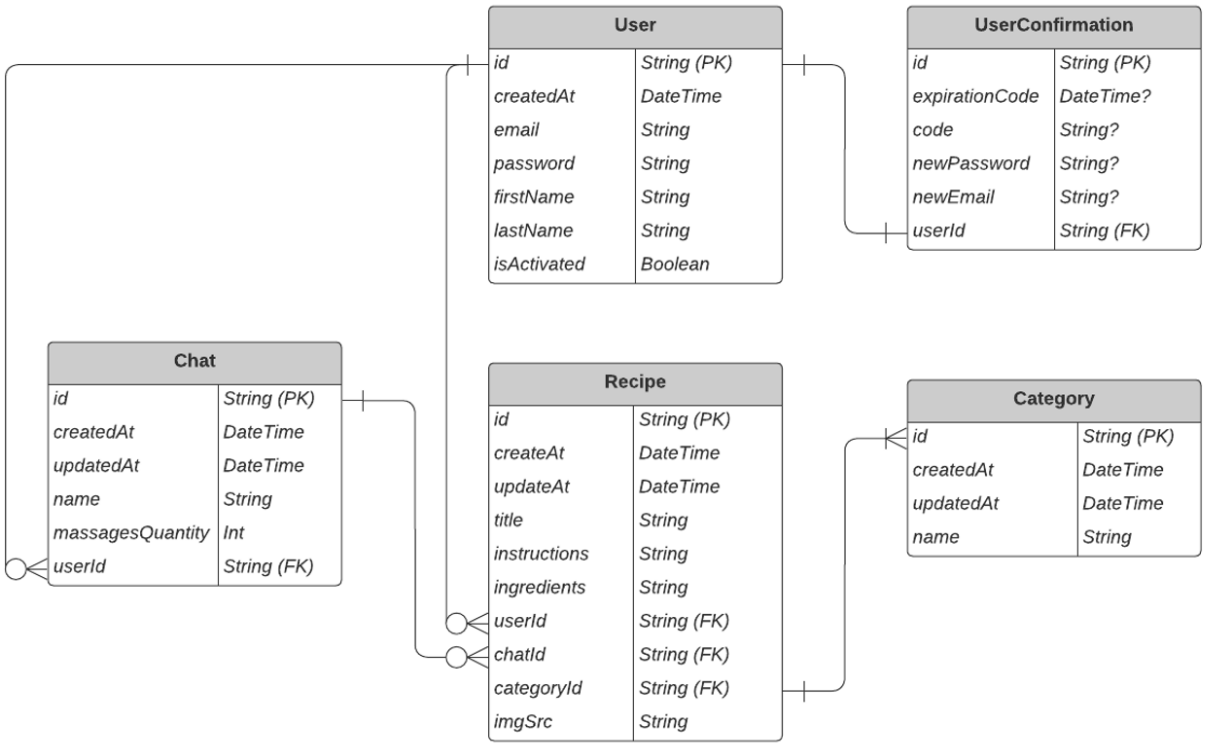

Рисунок 3.1 – ER-діаграма для бази даних

3.1.3 Модульна архітектура

Модульна архітектура в розробці програмного забезпечення дозволяє розділити додаток на окремі, незалежні модулі, кожен з яких виконує певну функцію або відповідає за певну частину функціональності. Цей підхід забезпечує високу організованість, легкість підтримки та масштабованість додатка [22-23]. Якщо використовується стиль модульної архітектури Angular, це означає, що дотримуються певних принципів і підходів, характерних для цього фреймворку.

Основні принципи модульної архітектури в стилі Angular ґрунтуються на розподілі додатка на окремі, незалежні модулі, кожен з яких виконує конкретну функцію або відповідає за певну частину функціональності (рис. 3.2). Такий поділ допомагає організувати код, полегшити його підтримку та забезпечити масштабованість додатка. Наприклад, це можуть бути модулі користувачів, управління продуктами або автентифікації. Кожен модуль декларує компоненти, директиви та пайпи, які він використовує, і імпортує інші модулі, що надають

необхідні залежності. Це робить код більш організованим та зрозумілим, оскільки всі залежності чітко визначені в межах кожного модуля.

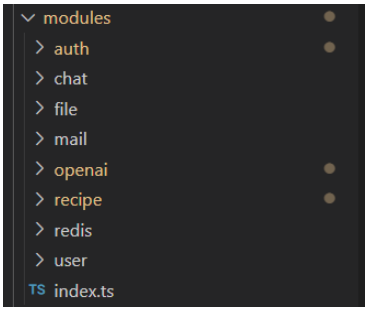

Рисунок 3.2 – Модульна архітектура

Розподіл відповідальності між модулями спрощує розробку, тестування та підтримку коду. Кожен модуль має свою область функціональності, що дозволяє розробникам зосередитися на конкретних завданнях, зменшуючи ризик помилок і полегшуючи налагодження. Оскільки функціональні частини додатка ізольовані в окремих модулях, код легко повторно використовувати. Модулі можна імпортувати і використовувати в інших частинах додатка або навіть в інших проектах.

Angular підтримує lazy loading модулів, що дозволяє завантажувати їх лише тоді, коли вони потрібні. Це покращує продуктивність додатка, зменшуючи час початкового завантаження. Застосування модульної архітектури в стилі Angular дозволяє створювати добре організовані, підтримувані та масштабовані додатки. Розділення додатка на модулі з чітко визначеними відповідальностями полегшує розробку та підтримку, підвищує продуктивність і сприяє повторному використанню коду, що дозволяє ефективно працювати над складними проектами.

Переваги модульної архітектури:

– організованість та структурованість;
– легкість підтримки та розширення;
– покращена продуктивність;
– масштабованість;

– повторне використання.

Застосовуючи модульну архітектуру, можна отримати чітко структурований, підтримуваний та масштабований додаток, який легко розвивати та підтримувати.

3.2 Автентифікація

3.2.1 Серверна частина

Модуль аутентифікації складається з кількох основних компонентів, розташованих у відповідних піддиректоріях: const, dto, services, і strategy, кожна з яких відіграє свою роль у процесі аутентифікації.

Модуль аутентифікації розділений на декілька підкомпонентів для забезпечення модульності та легкості управління кодом. Конфігураційні константи та типи помилок визначаються у папці const. Об'єкти передачі даних (DTO - Data Transfer Objects), які використовуються для перевірки вхідних даних користувача при реєстрації та вході, розміщені в папці dto. Логіка аутентифікації і видачі токенів, реалізована в сервісах у папці services, а валідація та екстракція токенів здійснюються за допомогою стратегій у папці strategy [24].

JWTService відповідає за генерацію токенів доступу та оновлення [8]. Під час генерації токенів, сервіс використовує допоміжні методи з JwtStrategy для створення та надсилання токенів у вигляді куків у відповіді до клієнта. Це дозволяє забезпечити безпечну та ефективну аутентифікацію на клієнтській стороні.

AuthService керує реєстрацією нових користувачів, активацією акаунтів та входом користувачів. Сервіс використовує UserService для доступу до даних користувачів та MailService для надсилання листів з кодом активації. Реєстрація користувачів включає перевірку наявності електронної пошти в базі даних і, у разі відсутності акаунту, створення нового користувача з хешованим паролем. Активація користувачів відбувається через перевірку наданого коду активації.

JwtStrategy реалізує стратегію аутентифікації на основі токенів JWT,

використовуючи бібліотеку Passport. Стратегія описує методи для екстракції токену з куків, валідації токену та асоціації користувача з отриманими даними. Зокрема, використовуються два типи токенів: доступу та оновлення, що дозволяє забезпечити безперервний доступ користувачів до системи.

У контексті безпеки важливо відмітити використання хешування паролів з bcrypt, що забезпечує захист від атак методом підбору. Також значну увагу приділено безпечному зберіганню та передачі JWT токенів за допомогою HTTPS та куків із належними налаштуваннями безпеки.

3.2.2 Клієнтська частина

На стороні фронтенду веб-додатку реалізовано систему аутентифікації, що використовує бібліотеку React і організована у вигляді двох основних модулів: sign-in та sign-up. Кожен модуль відповідає за окрему сторінку інтерфейсу користувача і інкапсулює всю необхідну логіку для виконання специфічних дій, таких як реєстрація або вхід користувачів.

Кожен з модулів включає кілька підпапок і файлів:

– компоненти (components), які виконують роль візуального представлення функціоналу на сторінках входу та реєстрації;

– хуки (hooks) кастомні визначають і керують логікою взаємодії з API та управлінням станом форм. Наприклад, хук useSignUp керує процесом реєстрації, включаючи валідацію даних і взаємодію з сервісом аутентифікації.

AuthService є основним сервісом для роботи з API. Цей сервіс використовує HttpService для відправлення запитів до сервера, які включають перевірку стану сесії користувача, реєстрацію нових користувачів, вхід в систему, активацію акаунту, повторну відправку коду активації, та вихід з системи.

У модулі sign-up реалізовано компонент SignUp, який використовує форму для збору даних користувача. Хук useSignUp керує логікою подання форми, включаючи перевірку валідності паролю, відправку даних на сервер через AuthService та перенаправлення на сторінку активації після успішної реєстрації.

Логіка валідації форми реалізована через validateFormValues, яка забезпечує додаткову перевірку введених даних перед відправкою на сервер.

Ця структура дозволяє чітко розділити логіку обробки даних та візуальне представлення, забезпечуючи легкість підтримки та розширення додатку. Кастомні хуки допомагають ізолювати специфічні процеси аутентифікації, роблячи код більш чистим і зрозумілим.

3.3 Розробка чат-бота

3.3.1 Модуль OpenAI

Для розробки функціональності чат-бота в проекті було вирішено інтегрувати API OpenAI, що дозволяє використовувати передові можливості машинного навчання для обробки природної мови та генерації зображень. Це інтеграція була здійснена через створення спеціалізованого модуля в рамках фреймворку NestJS.

Модуль ініціює з'єднання з API OpenAI через налаштування [4], що визначаються у функції createOpenaiConfig. Ця функція використовує сервіс ConfigService для отримання необхідних конфігураційних даних, таких як API ключ, базовий URL, ID організації та проекту. Налаштування включають також заголовки для авторизації запитів.

OpenaiService є основним сервісом для взаємодії з OpenAI. Він ініціалізує клієнт OpenAI з необхідними конфігураційними параметрами. Сервіс надає метод createPrompt, який формує вхідні дані для моделі чату OpenAI, включаючи обробку текстових повідомлень та зображень, переданих користувачем.

Метод createPrompt готує дані для передачі в модель чату OpenAI, де:

– якщо користувач надіслав зображення, воно обробляється як зображення страви, і модель повинна надати рецепт;

– у випадку текстового запиту, модель генерує рецепт за зазначеними інгредієнтами або запитом;

– прийом історії розмови дозволяє моделі враховувати попередній контекст для забезпечення консистентності відповідей.

OpenaiModule інтегрує всі необхідні елементи, включаючи HTTP модуль для асинхронної реєстрації залежностей та конфігурацій. Це дозволяє легко інтегрувати сервіс в загальну структуру додатку і використовувати його для обслуговування запитів користувачів.

Розроблений чат-бот може використовуватися в різноманітних сценаріях, включаючи кулінарні додатки, де користувачі можуть отримувати рецепти страв шляхом надсилання зображень або описів страв. Така функціональність не тільки збільшує залученість користувачів, але й покращує їхній досвід використання додатку.

3.3.2 Модуль Chat

У модулі Chat веб-додатку інтегровано кілька ключових технологій для оптимізації взаємодії з користувачами. Модуль складається з двох основних компонентів: ChatController і ChatService, які забезпечують управління чатами через веб-інтерфейс і виконання бізнес-логіки відповідно.

ChatController відповідає за прийом та обробку HTTP запитів від користувачів. Зокрема, контролер обробляє запити на отримання всіх чатів користувача, деталей конкретного чату, створення нових чатів, додавання повідомлень із можливістю включення зображень, а також видалення чатів. Захист маршрутів забезпечується за допомогою JwtAuthGuard, а завантаження файлів контролюється через FileInterceptor, що дозволяє встановлювати обмеження на типи файлів та їх розмір.

ChatService відіграє роль бекенду, виконуючи всю логіку, пов’язану з взаємодією з базою даних через Prisma, керуванням даними у Redis, а також інтеграцією з OpenAI. Сервіс обробляє створення, отримання, та видалення чатів, а також управління новими повідомленнями, включаючи взаємодію з OpenAI для аналізу зображень або тексту, зберігання повідомлень у Redis для швидкого

доступу та формування відповідей чат-бота.

Інтеграція з OpenAI дозволяє модулю обробляти повідомлення, відправляючи текст або зображення до OpenAI і отримуючи релевантні відповіді. Взаємодія з Redis використовується для оптимізації швидкості доступу до історії чатів, зменшення навантаження на базу даних, і забезпечення швидких відгуків користувачам.

Модуль Chat є прикладом того, як можна ефективно використовувати сучасні технології для створення динамічних і високопродуктивних веб-додатків. Він демонструє інтеграцію різних платформ і технологій, таких як бази даних, кешування в пам'яті, та штучний інтелект, для створення розширених користувацьких досвідів, що покращують взаємодію з користувачем і забезпечують миттєве відповідання на запити.

3.3.3 Модуль Redis та його інтеграція у модуль chat

Redis є високопродуктивною базою даних в пам'яті, яка забезпечує швидке зберігання та отримання даних, що ідеально підходить для використання у чат-додатках, де необхідна швидка реакція на запити користувачів.

RedisModule є глобальним модулем в NestJS, що забезпечує централізоване управління з'єднаннями Redis. Цей модуль визначає провайдера REDIS_CLIENT, який створює екземпляр клієнта Redis з параметрами конфігурації, взятими з змінних середовища. Використання глобального модуля дозволяє доступ до клієнта Redis з будь-якої частини додатку без необхідності його повторного імпортування або конфігурування.

Інтеграція Redis дозволяє швидко зберігати історію чатів та отримувати доступ до них при необхідності. Завдяки його високій швидкості читання та запису, Redis ідеально підходить для застосувань реального часу, таких як чат-боти, де важлива миттєва реакція на запити користувачів. Зберігання історії чатів у Redis також дозволяє легко відновлювати попередній контекст розмови, що є критично важливим для забезпечення послідовності та релевантності відповідей

чат-бота. У модулі Redis для чат-додатку, ми бачимо ефективне використання операцій на списку Redis для забезпечення швидкого доступу до історії повідомлень. Операція lrange використовується для отримання всіх повідомлень зі списку, що асоційований з конкретним чатом. Наприклад, у функції getChat, ключ chat:${id} використовується для доступу до історії чату, де id –це унікальний ідентифікатор чату. Запит Redis this.redisClient.lrange(key, 0, chat.messagesQuantity - 1) дозволяє отримати всі повідомлення з кешу, починаючи з найновіших. Повідомлення, отримані у відповідь, інвертуються методом reverse() для відображення у хронологічному порядку, лістинг коду наведено нижче.

```
const messages = await this.redisClient.lrange(key, 0, chat.messagesQuantity - 1);
return {
    chat,
    messages: messages.reverse(),
}
```

Цей приклад демонструє, як Redis оптимізує відгук додатку, забезпечуючи миттєвий доступ до важливих даних, не завантажуючи базу даних.

Ще один аспект використання Redis у чат-додатку – це зберігання нових повідомлень у чаті. При створенні нового повідомлення, як текстового, так і з медіа, повідомлення спочатку обробляються через OpenAI для отримання реакції або аналізу зображення, а потім зберігаються у Redis за допомогою команди lpush. Це дозволяє зберегти історію взаємодії у вигляді стеку, де нові повідомлення додаються на початок списку, як наведено в лістингу коду нижче.

```
await this.redisClient.lpush(key, questionPayload);
await this.redisClient.lpush(key, answerPayload);
```

Цей метод ефективно використовується для зберігання історії чатів у форматі, який може бути швидко відновлений і представлений користувачам, забезпечуючи збереження контексту та неперервність розмови. Ці операції показують, як використання Redis може покращити продуктивність і масштабованість веб-додатків, особливо в умовах, де швидкість відповіді та доступу до даних є критично важливими.

3.3.4 Фронтенд модуль Chat

Цей модуль створений для інтерактивного спілкування користувачів з системою через чат, де користувачі можуть ставити запитання (з можливістю додавання зображень) про рецепти, які оброблятимуться на бекенді, звідки і надходять відповіді. В чаті також доступна історія переписки і можливість збереження рецепту з чату.

Модуль містить кілька ключових компонентів. Компонент Input.component.tsx призначений для введення тексту повідомлень та завантаження зображень. Message.component.tsx відображає окреме повідомлення у чаті. MessagesSection.component.tsx відображає всю секцію повідомлень, де користувач може бачити всю історію переписки. SaveRecipeModal.component.tsx є модальним вікном, що дозволяє користувачу зберегти рецепт із чату.

У папці hooks знаходяться такі хуки як useGetChat, який завантажує дані чату, включаючи повідомлення та історію. useInput управляє станом введення тексту та завантажених файлів. useMessageParser аналізує повідомлення на предмет інформації про рецепт і визначає, чи можна зберегти рецепт. useSaveRecipe обробляє логіку збереження рецепту.

Коли користувач вводить повідомлення або завантажує зображення через компонент Input, відбувається виклик хука useInput. Після обробки введення та можливого завантаження файлу, формується запит до бекенду для обробки повідомлення. Логіка завантаження та відправки даних на сервер включає створення FormData та додавання в неї файлу зображення і тексту повідомлення. Відповіді від сервера використовуються для оновлення стану чату в сторі.

3.4 Збереження рецептів

3.4.1 Модуль Recipe. Генерація зображень

Цей модуль забезпечує повний спектр функціональностей для управління рецептами, включаючи створення, оновлення, видалення, а також отримання списку рецептів або окремого рецепта за категорією або ідентифікатором.

Ключовою функцією модуля є створення нових рецептів, де зазначено інтеграцію з сервісом OpenAI для генерації відповідного зображення до рецепта. Наприклад, при створенні нового рецепта система спочатку звертається до OpenAI з проханням створити зображення за назвою рецепта. Відповідь від OpenAI, яка включає зображення у форматі base64, обробляється і зберігається за допомогою сервісу файлів, як наведено в лістингу коду нижче.

```
const res = await this.openAI.createImage(data.title);

const buffer = Buffer.from(res.data[0]!.b64_json, 'base64');

const uploaded = await this.fileService.uploadFile({

    buffer,

    originalname: `${res.created}.png`,

    encoding: 'binary',

    mimetype: 'image/png',

    size: buffer.length,

    destination: 'generated',

    path: '',

    fieldname: '',

}, userId);
```

Після цього, зображення та інша інформація про рецепт зберігаються в базі даних. Цей процес не тільки автоматизує створення візуального контенту для рецептів, але й покращує візуальну привабливість запропонованих рецептів на платформі.

Для управління існуючими рецептами розроблено методи, які дозволяють користувачам оновлювати та видаляти рецепти. Оновлення рецепту може включати зміну назви, інструкцій, інгредієнтів або категорії.

Окрім того, модуль містить функції для отримання списків рецептів, що дозволяє користувачам переглядати рецепти, фільтрувати їх за критеріями, такими як категорія, назва або інгредієнти, а також переглядати деталі конкретних

рецептів. Ці можливості роблять взаємодію з рецептами гнучкою та зручною, підтримуючи високий рівень користувацького досвіду.

3.4.2 Фронтенд модулі Recipe і Recipes

Менеджмент рецептів у фронтенді реалізовано через два основних модулі: "recipe" і "recipes". Модуль "recipes" відповідає за сторінку зі всіма рецептами, де користувачі мають можливість фільтрувати рецепти за категоріями. Компонент "Home" є вхідним пунктом до цього модуля, де використовується хук useGetRecipes для завантаження рецептів, а стан рецептів керується через контекст useRecipes.

Компонент "Home" має кнопку для додавання нових рецептів, яка активує модальне вікно AddRecipe за допомогою діалогового хука useDialog. Рецепти відображаються у секції, де кожен рецепт є клікабельним і веде до детальної сторінки цього рецепта. Кліки по рецептах обробляються функцією clickHandler, яка навігує користувача до сторінки з деталями рецепта.

Модуль "recipe" представляє сторінку одного рецепта. Він містить компоненти для відображення, редагування та видалення детальної інформації про рецепт. Користувачі можуть перейти до редагування рецепту через форму "EditForm", яка забезпечує інтерфейс для оновлення даних рецепта. Хук useEditRecipe використовується для керування станом форми редагування, включаючи валідацію даних і обробку подій форми. У цьому контексті, "EditForm" відображає різні поля, які можуть бути полями вводу або випадаючими списками (Dropdown), в залежності від типу даних, які потрібно ввести. Кнопки "Reset" та "Update" дозволяють користувачам відновити початкові значення або зберегти оновлення.

Висновки до розділу 3

У розділі 3 було детально розглянуто архітектуру та функціональність фронтенду та бекенду системи керування рецептами, включаючи інтеграцію з сервісами для обробки зображень та текстових даних. Ключові аспекти

включають розробку інтерактивних модулів чату та рецептів, які забезпечують динамічну взаємодію з користувачами та ефективне управління контентом. Модуль чату демонструє високий рівень інтерактивності та зручності для користувачів, дозволяючи їм в реальному часі взаємодіяти з системою, ставлячи запитання та отримуючи інформацію про рецепти. Функціональність модуля включає можливість збереження рецептів з чату, що збільшує залученість користувачів і полегшує зберігання улюблених рецептів. Інтеграція з веб-сервісами, такими як OpenAI для обробки запитань і зображень, додатково підсилює функціональність модуля.

Модуль рецептів, який включає сторінки для перегляду всіх рецептів і деталей окремих рецептів, надає користувачам гнучкі інструменти для управління рецептами. Функції фільтрації за категоріями та зручний інтерфейс для редагування та видалення рецептів забезпечують ефективне управління контентом. Використання форм для введення даних та інтеграція з базою даних через ORM підкреслює технічну ефективність і безпеку додатку.

В цілому, розроблені модулі демонструють глибоку інтеграцію клієнської та серверної частин з сучасними технологіями для обробки даних та зручності користувачів. Використання сучасних JavaScript фреймворків та бібліотек, таких як React і Zustand для управління станом додатку, підсилює здатність системи масштабуватися та адаптуватися до зростаючих вимог користувачів. Забезпечення безпеки, швидкості обробки запитів і зручності користувацького інтерфейсу є ключовими факторами, які сприяють загальній успішності системи управління рецептами.

4 ОСНОВНІ ФУНКЦІЇ ТА ІНТЕРФЕЙС ВЕБЗАСТОСУНКУ

4.1 Реєстрація та автентифікації

Під час відкриття сайту користувачі спочатку потрапляють на сторінку автентифікації (рис. 4.1). Ця сторінка дозволяє зареєстрованим користувачам ввести свої облікові дані (електронну пошту та пароль) для доступу до функцій вебзастосунку.

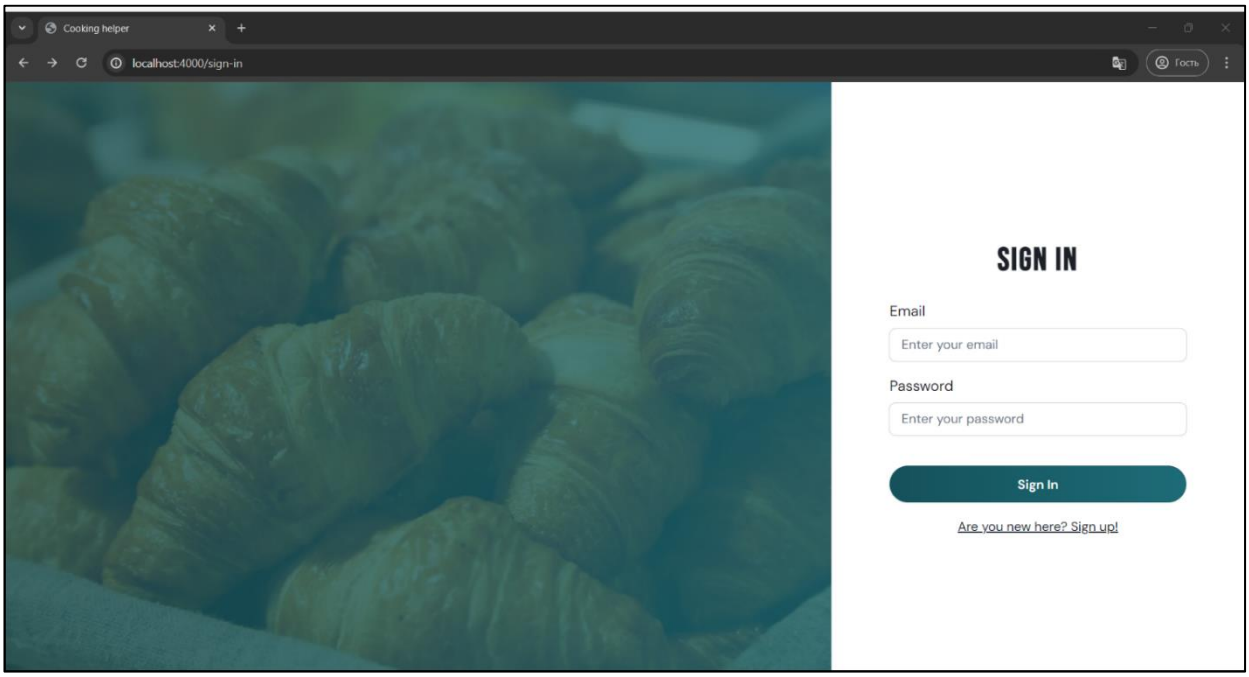

Рисунок 4.1 – Сторінка автентифікації

Якщо користувач не має акаунту, на цій же сторінці є опція для переходу до сторінки реєстрації (рис. 4.2). Процес реєстрації включає введення необхідної інформації, такої як ім'я, прізвище, електронна пошта, пароль та його підтвердження.

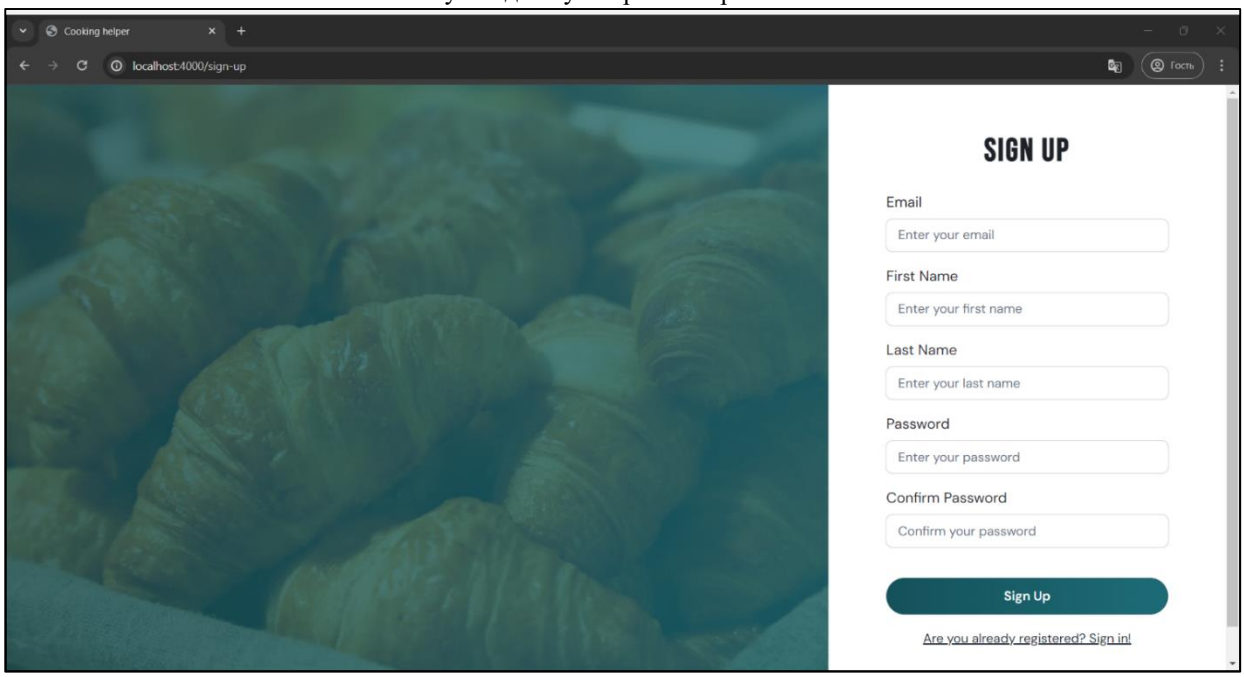

Рисунок 4.2 – Сторінка реєстрації

При реєстрації у нашому кулінарному вебзастосунку, користувачам необхідно створити безпечний пароль, який відповідає певним вимогам.

Пароль повинен відповідати наступним критеріям:

– довжина паролю від 8 до 20 символів;

– хоча б одну малу літеру (a-z);

– хоча б одну велику літеру (A-Z);

– хоча б одну цифру (0-9);

– може містити літери (a-z, A-Z), цифри (0-9) та спеціальні символи (@$!%*#?&{}()[]"'^+,=-).

Ці вимоги забезпечують достатній рівень безпеки пароля, роблячи його важким для злому.

Для підтвердження правильності введеного пароля, користувачі повинні повторно ввести свій пароль у поле підтвердження пароля (рис. 4.3). Це допомагає запобігти помилкам при введенні та забезпечує, що користувач запам’ятає свій пароль.

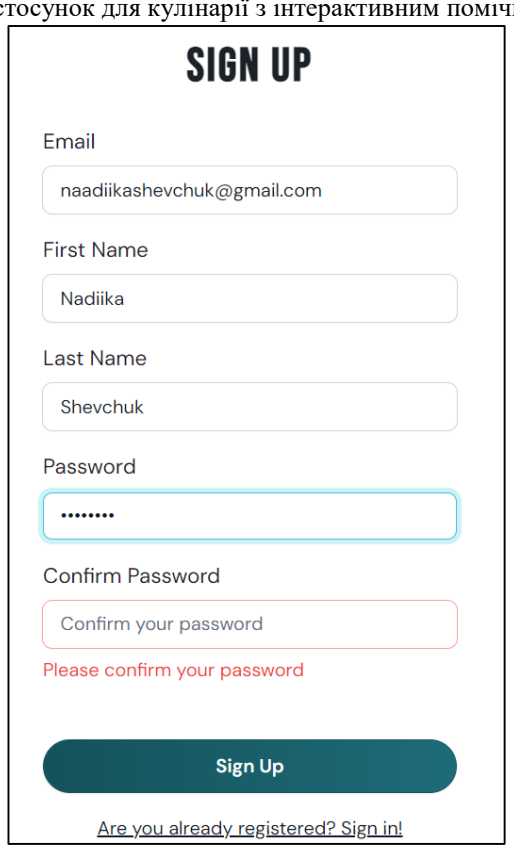

Рисунок 4.3 – Фрагмент зі сторінки реєстрації

Процес підтвердження чи схожості введених паролів відбувається одразу під час введення. Система автоматично перевіряє, чи збігаються обидва паролі, і повідомляє користувача, якщо вони не співпадають (рис. 4.4).

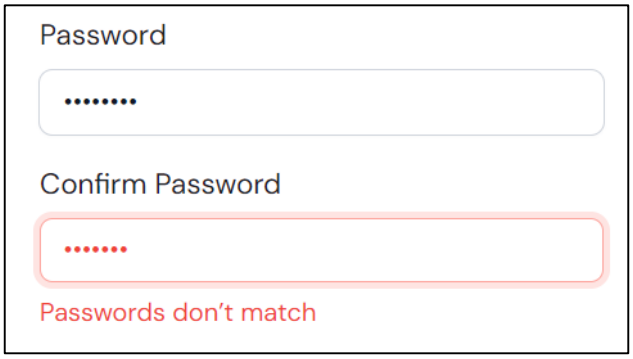

Рисунок 4.4 – Фрагмент зі сторінки реєстрації з підтвердженням паролю

Після натискання кнопки "Sign up" на вказану електронну пошту надсилається лист з верифікаційним кодом підтвердження а на самому сайті

активується сторінка активації акаунту для введення коду (рис. 4.5). Це необхідно для підтвердження, що користувач вказав дійсну електронну адресу.

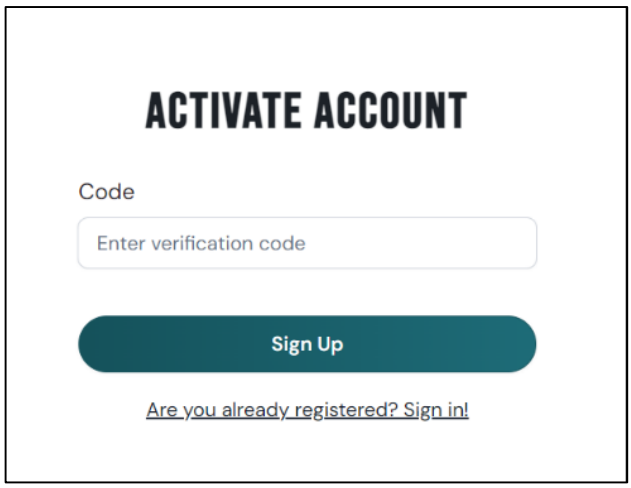

Рисунок 4.5 – Сторінка активації акаунту при реєстрації

Користувач відкриває лист (рис. 4.6), читає верифікаційний код і вводить його на сайті, щоб завершити процес реєстрації. Це підтверджує, що електронна пошта дійсна та належить користувачу.

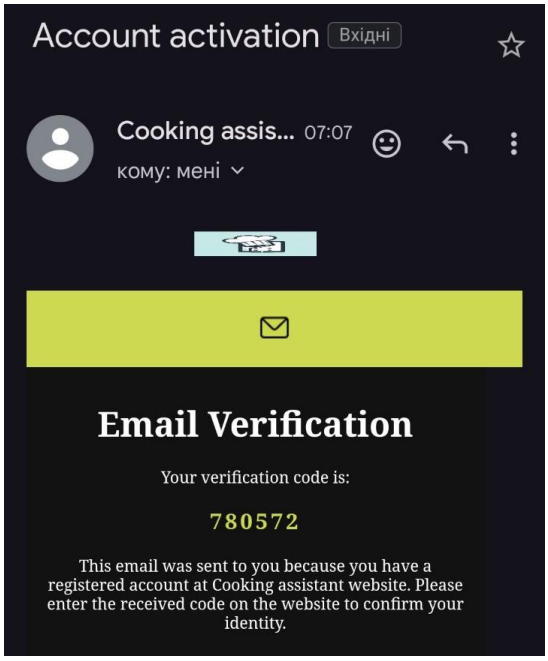

Рисунок 4.6 – Лист з верифікаційним кодом

Після підтвердження електронної пошти користувач автоматично отримує доступ до свого нового облікового запису. Тепер він може увійти в систему, використовуючи свою електронну пошту та пароль.

4.2 Основні функції та інтерфейс

Після успішної автентифікації користувач опиняється на головній сторінці вебзастосунку (рис. 4.7). На цій сторінці представлено декілька рецептів та кнопка для додавання нового рецепту. Окрім цього, на сторінці розміщено слайдбар із кнопкою для переходу до інтерактивного помічника, саме до чат-бота, та рецепти, розподілені за категоріями, такими як:

– bakes goods (випічка);
– snacks (закуски);
– first courses (перші страви);
– second courses (другі страви);
– salads (салати);
– desserts (десерти);
– drinks (напої).

Слайдбар також містить кнопку для виходу з акаунта, що дозволяє користувачу швидко здійснити вихід з вебзастосунку. На головній сторінці відображаються кілька останніх рецептів. Користувач може переглядати ці рецепти, натискаючи на них для отримання детальнішої інформації. Категорії рецептів дозволяють швидко фільтрувати та переглядати рецепти за обраною категорією.

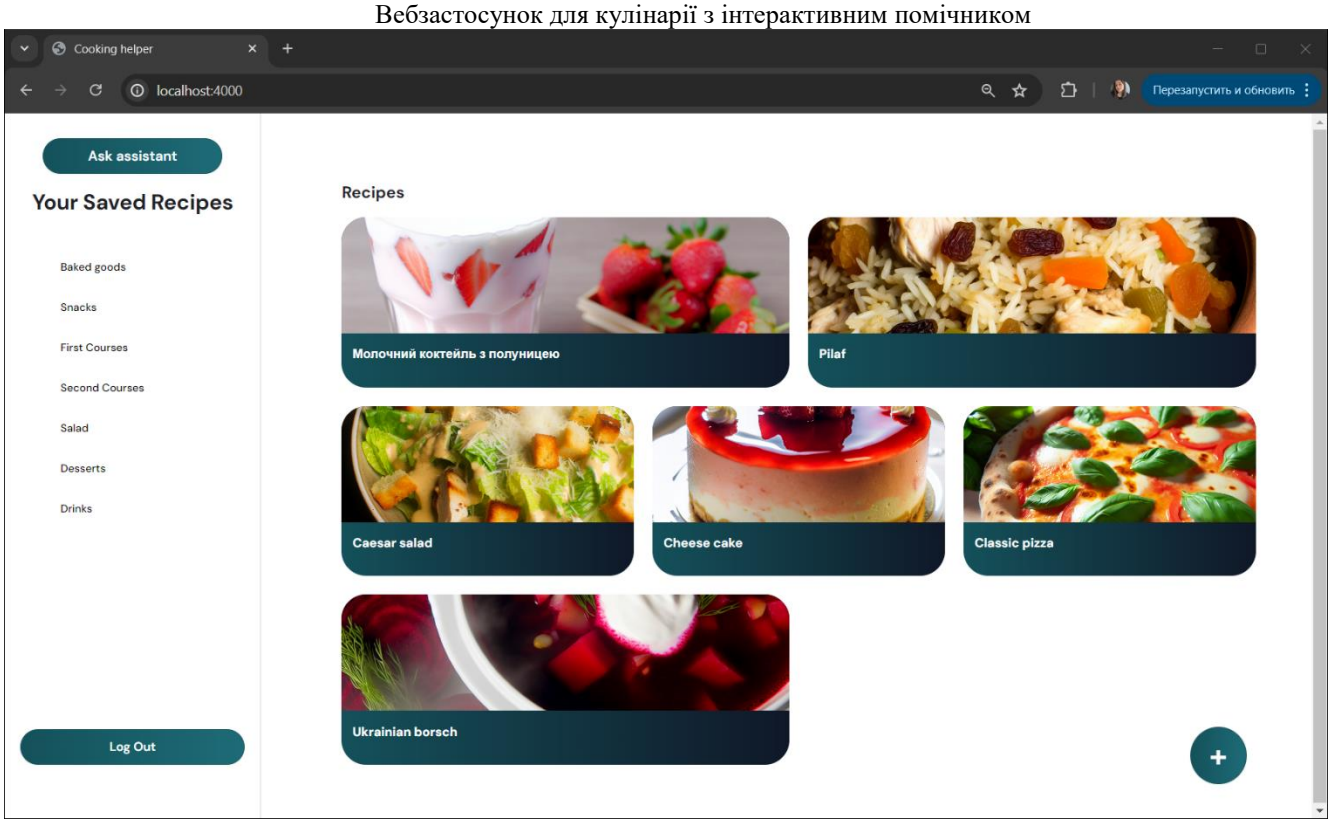

Рисунок 4.7 – Головна сторінка вебзастосунку

Цей інтерфейс забезпечує зручний доступ до основних функцій застосунку, дозволяючи користувачам легко знаходити рецепти, додавати нові, а також отримувати допомогу від інтерактивного помічника.

4.3 Використання інтерактивного помічника

Після переходу до інтерактивного помічника користувач опиняється на сторінці "Your assistants". На цій сторінці (рис. 4.8) відображається список доступних чатів, які він може використовувати для отримання консультацій по будь-яким темам рецептів. Крім того, на сторінці є кнопка для додавання нового чату для отримання консультацій.

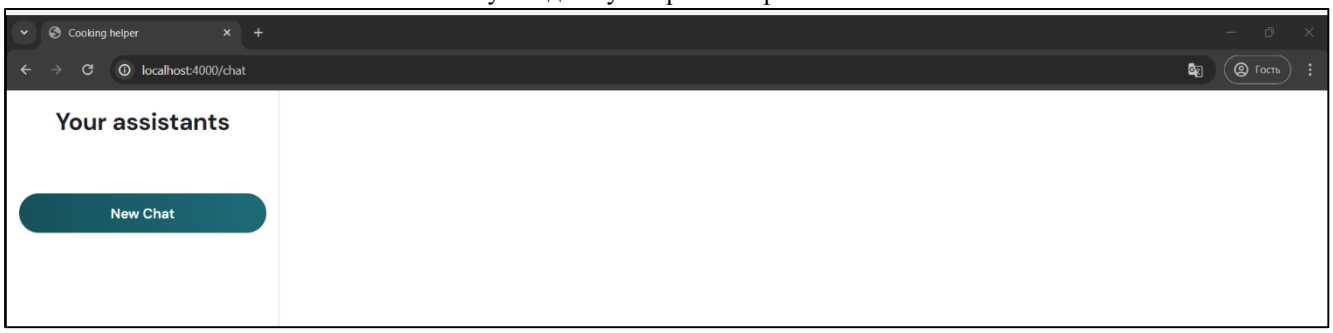

Рисунок 4.8 – Сторінка інтерактивного помічника

Під час натискання кнопки «New Chat» відкривається спеціальна форма (рис. 4.9) для введення назви нового чату. Користувач вводить назву чату в поле на формі.

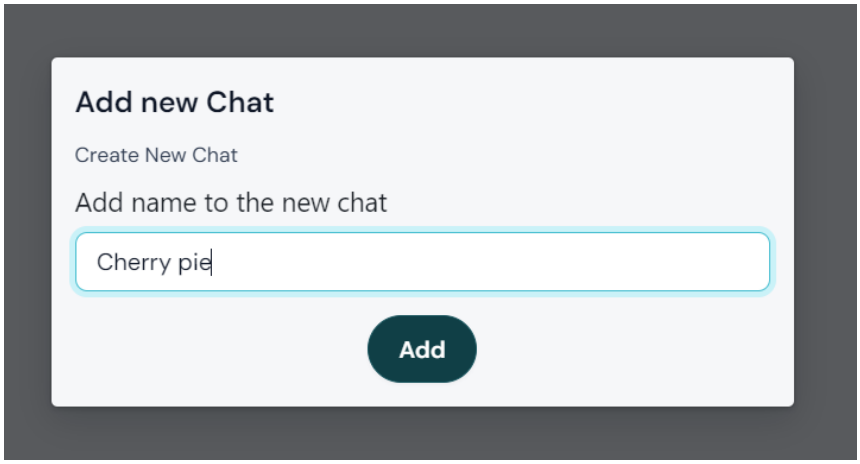

Рисунок 4.9 – Форма для створення чату

Після введення назви чату з'являється поле для введення тексту та можливість прикріплення файлів, зокрема фото. Користувач може запитувати про рецепти різноманітних страв, інгредієнти, приготування та інші аспекти, що стосуються кулінарії. Інтерактивний помічник готовий відповісти на запити у цій тематиці, надаючи користувачеві інформацію, поради або повні рецепти з ім'ям, інгредієнтами та інструкціями приготування (рис. 4.10).

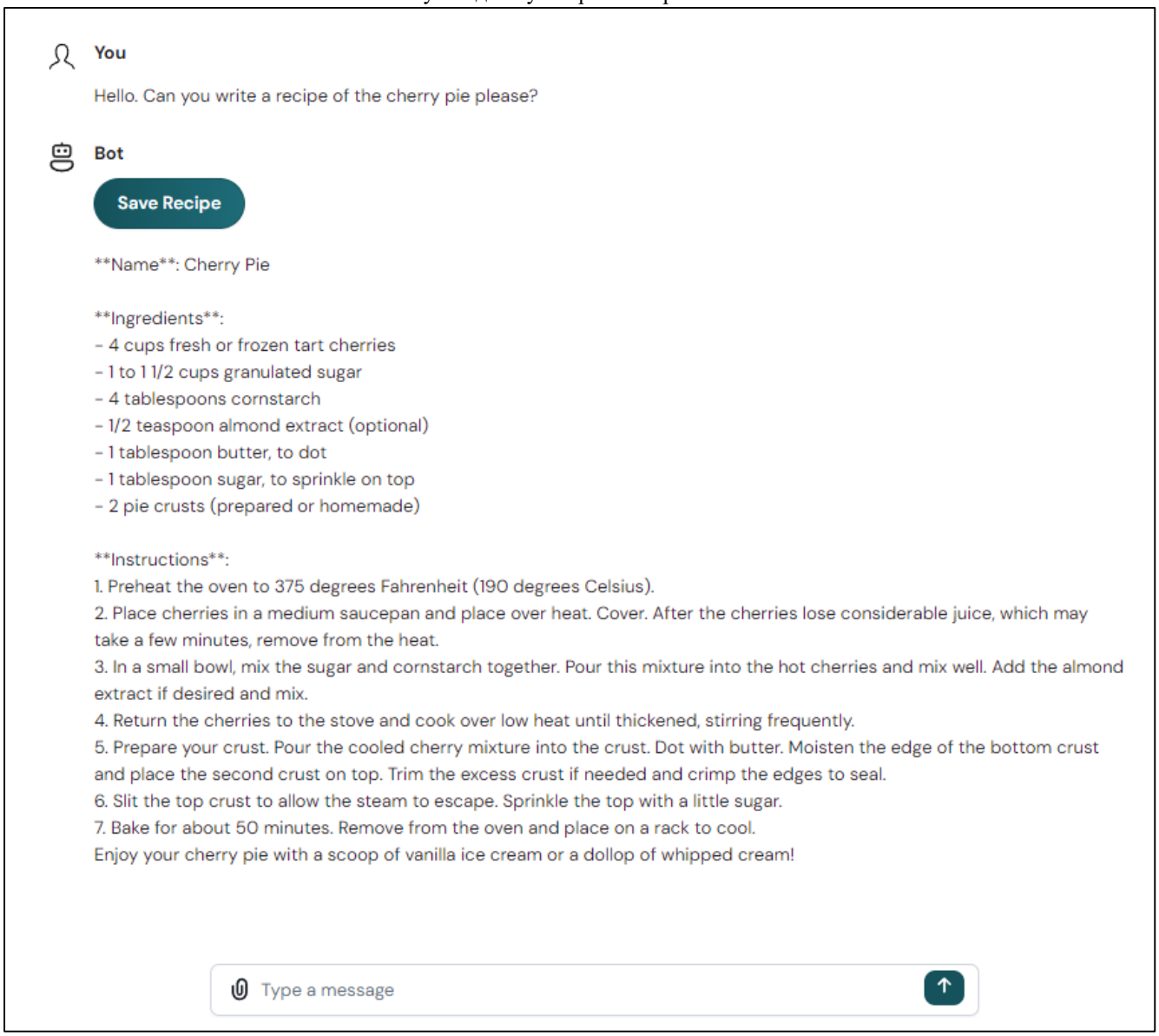

Рисунок 4.10 – Спілкування з інтерактивним помічником

Так як чат запам’ятовує інформацію про те, що користувач питав у нього раніше, то він може надавати консультації та видозмінювати рецепти (рис. 4.11), що є дуже зручним.

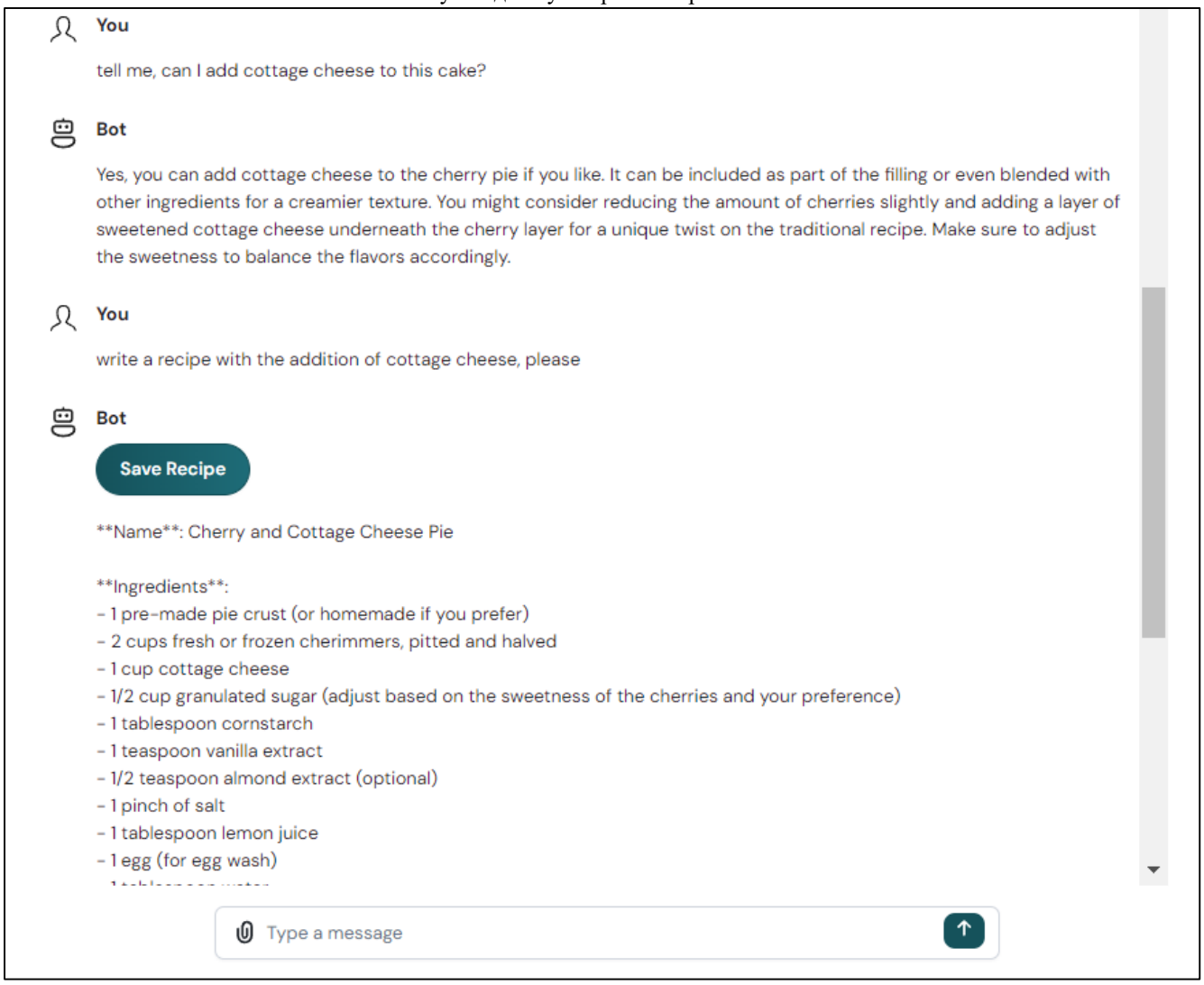

Рисунок 4.11 –Спілкування з інтерактивним помічником

Користувач може вибрати фото (рис. 4.12) страви для подальшого розпізнавання та отримання консультацій щодо цього рецепту (рис. 4.13).

Рисунок 4.12 –Можливість прикріплення рис.

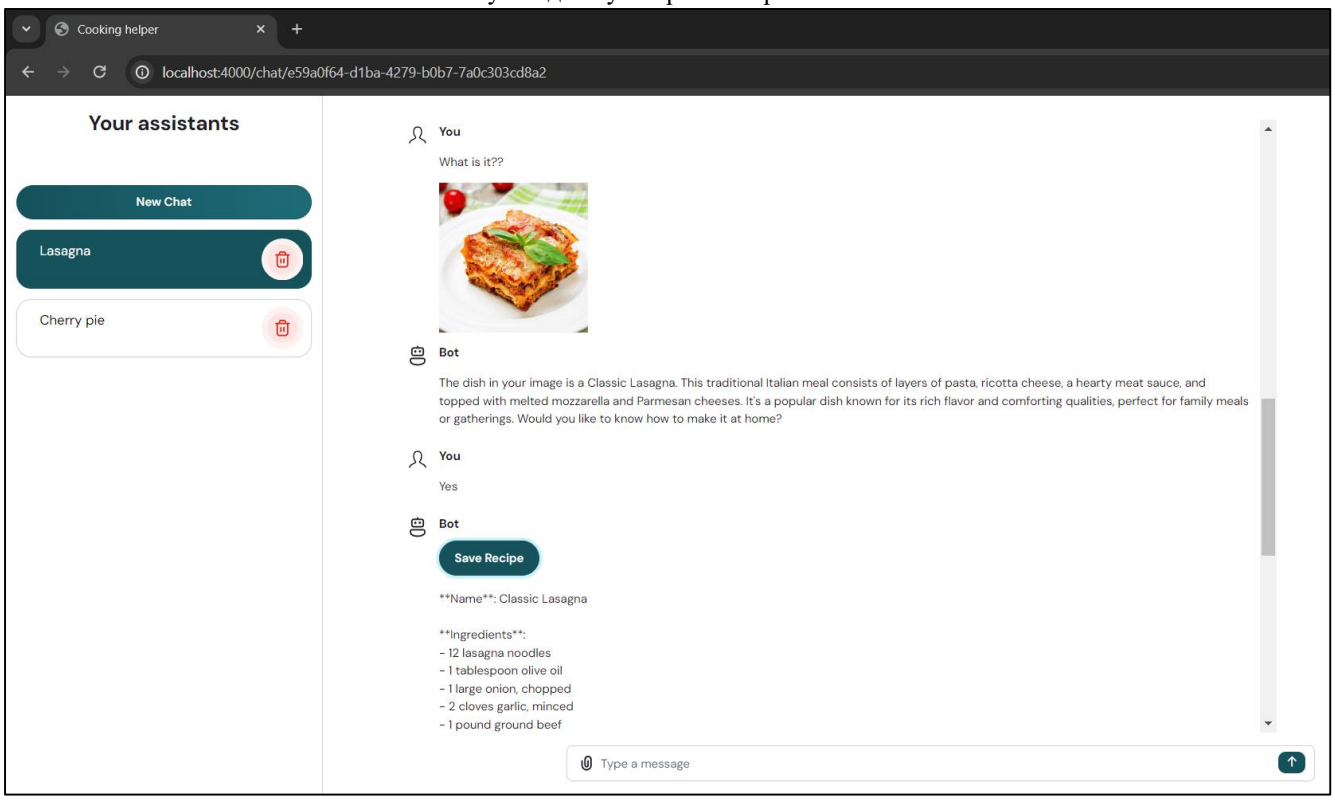

Рисунок 4.13 – Фрагмент розмови з помічником із рисунком

З інтерактивного помічника присутня можливість зберігати згенерований рецепт за допомогою кнопки «Save Recipe» до власного каталогу за обраною з наявних категорій страв в застосунку (рис. 4.14).

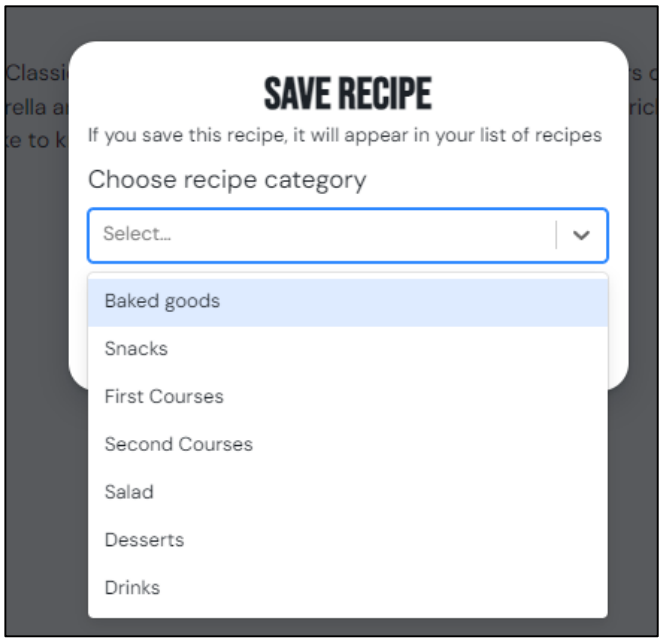

Рисунок 4.14 – Зберігання рецепту з чату з вибором категорії

При натисканні кнопки «Save» рецепти зберігаються в наступному вигляді (рис. 4.15), їх можна редагувати і видаляти.

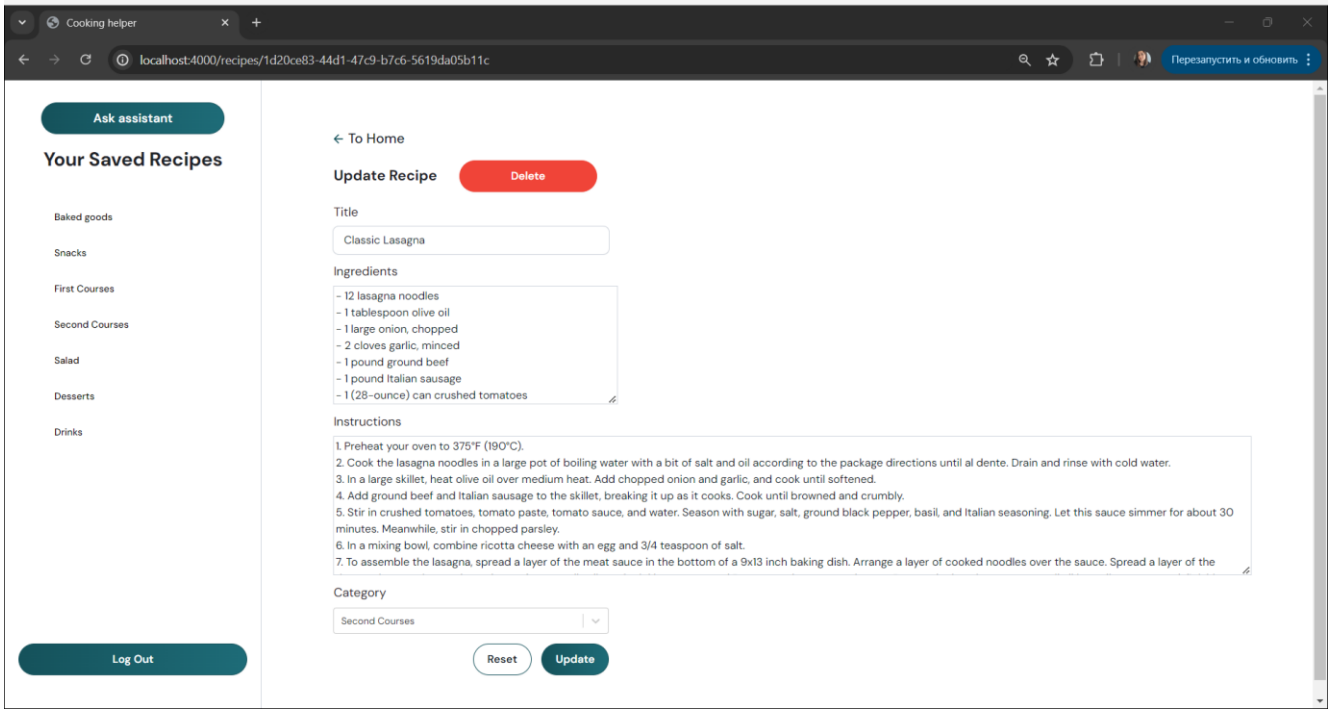

Рисунок 4.15 –Вигляд збереженого рецепту страви

Збережені рецепти з інтерактивного помічника зберігаються із автоматично згенерованими фотографіями страв, як наведено на рис. 4.16.

Рисунок 4.16 –Вигляд збереженого рецепту страви із згенерованою фотографією

4.4 Додавання та управління рецептами

При натисканні кнопки додавання рецепту на головній сторінці сайту випливає форма для вводу даних рецепта, а саме: назви, інгредієнтів, інструкції та обрання категорії (рис. 4.17).

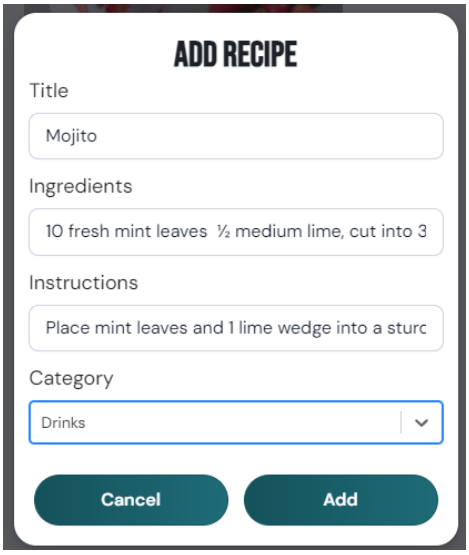

Рисунок 4.17 – Форма додавання рецепту

При додаванні рецепта автоматично генерується зображення за назвою страви (рис. 4.18) і прикріплюється до страви.

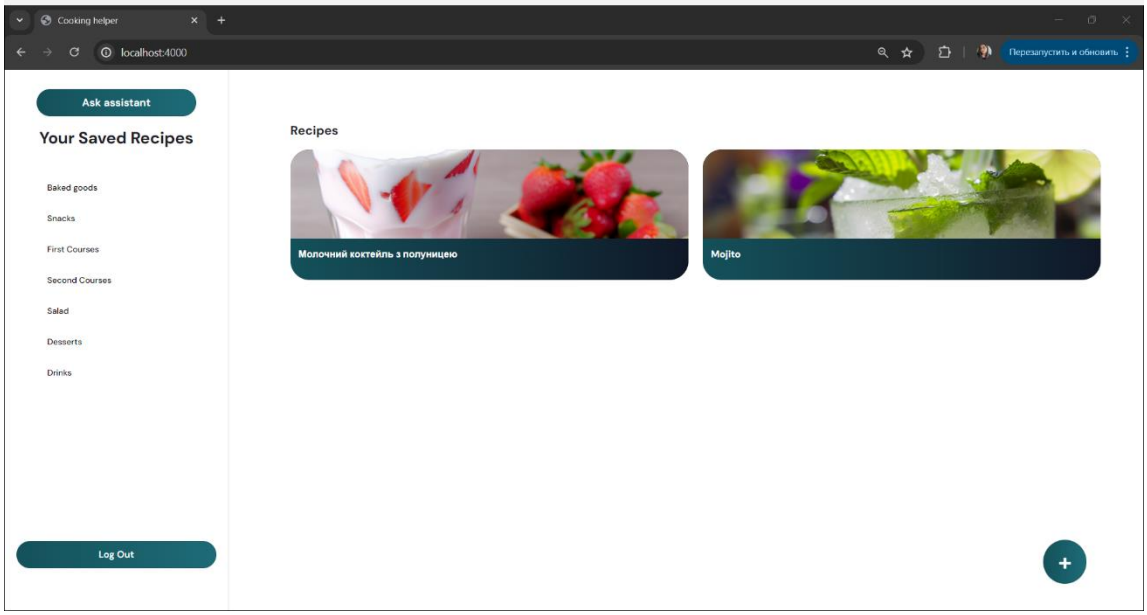

Рисунок 4.18 – Відображення доданого рецепту

При зменшенню розміру екрану можна побачити все генероване зображення (рис. 4.19).

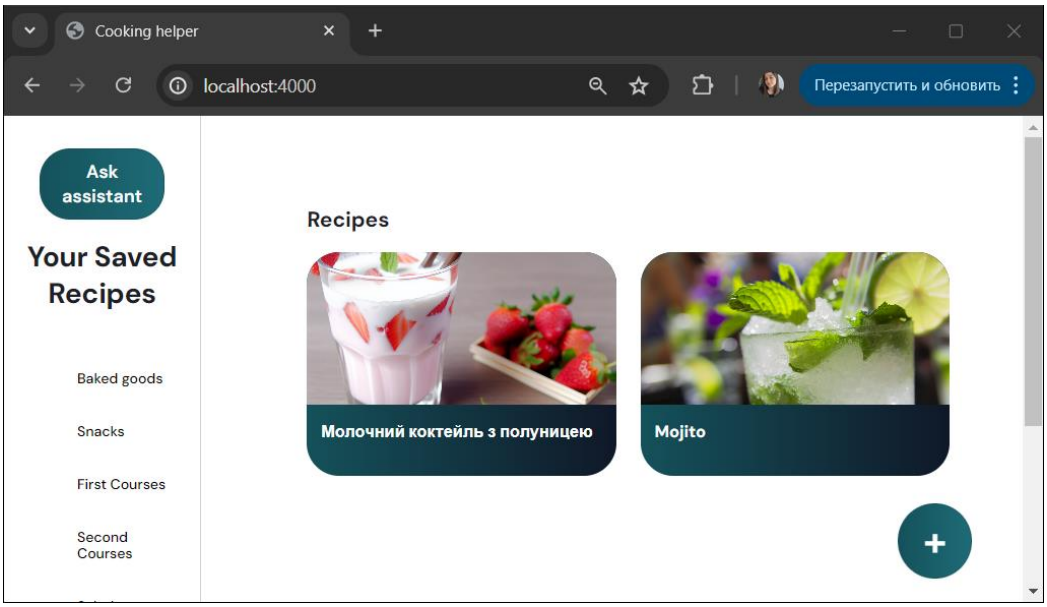

Рисунок 4.19 – Відображення рецептів в іншому масштабі сторінки

Натискаючи на страву відкривається сторінка з переглядом рецепту з можливістю змінення чи видалення його, так само, як і при збереженні їх з інтерактивного помічника.

Висновки до розділу 4

У розділі 4 було подано детальний опис функцій вебзастосунку, що включав в себе опис та навчання з реєстрації та авторизації на платформі, огляд основних функцій та інтерфейсу вебзастосунку, процес додавання та управління рецептами, а також використання інтерактивного помічника. Кожен пункт інструкції був ретельно розглянутий та проілюстрований за допомогою скріншотів, що дозволить користувачам легко зрозуміти та освоїти функціонал вебзастосунку. Представлений матеріал сприяє покращенню користувацького досвіду та забезпечує ефективне використання ресурсів вебзастосунку для досягнення поставлених завдань.

ВИСНОВКИ

Завданням кваліфікаційної роботи бакалавра є створення інтерактивного чат-бота у вебзастосунку, який вміє надавати користувачеві персоналізовані рекомендації щодо приготування конкретних страв та забезпечує можливість зберігати за окремими категоріями отримані рецепти.

У сфері онлайн-сервісів для кулінарії, цей проект вирішує актуальну проблему недостатньої персоналізації та інтерактивності у вже існуючих вебзастосунках. Для досягнення цієї мети використанні сучасні технології для веброзробки та штучний інтелект. Було розглянуто існуючі аналоги подібних вебзастосунків для кулінарії та здійснено порівняльний аналіз їхніх недоліків та переваг. Також був проведений аналіз сучасного стану проблеми та огляд основних методів вирішення. У даній роботі методом вирішення проблеми є реалізація чат-бота у вебзастосунку, який вміє надати персоналізовану підтримку та консультацію користувачам щодо приготування страв. Однією з головних переваг цього проекту є можливість користувачів дізнаватись рецепт страви не лише за їхніми назвами, а й за фотографіями. Ця функція дозволяє користувачам просто сфотографувати будь-яку страву на свій телефон і надіслати її в чат, де інтерактивний помічник зможе автоматично розпізнати її та надати відповідні рекомендації. У сучасному світі, коли соціальні мережі переповнені фотографіями їжі без назв, ця функція стає надзвичайно корисною та актуальною для користувачів, які шукають нові рецепти для готування.

Чат-бот використовує штучний інтелект для аналізу обмежень та уподобань конкретного користувача, відповідати на питання та надавати рекомендації щодо приготування страв. Окрім того, користувачі мають можливість отримані рецепти зберігати у вебсайту з генерацією зображень для подальшого використання. Результати розробки вебзастосунку з інтерактивним помічником можуть бути корисними для кола користувачів, які проявляють цікавість до кулінарії та шукають зручний та ефективний інструмент для приготування у домашніх умовах страв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. About NodeJs, URL: https://nodejs.org/en/about (дата звернення: 07.05.2024).

2. Auth0. Blog on JWT. URL: https://auth0.com/blog/jwt-handbook/ (дата звернення: 17.05.2024).

3. Banks A., Porcello E. Learning React. 2nd ed. O'Reilly Media, 2020. 45 с.

4. Coursera. AI for Everyone. URL: https://www.coursera.org/learn/ai-for-everyone (дата звернення: 05.05.2024).

5. Documentation NestJs, URL: https://docs.nestjs.com/ (дата звернення: 07.05.2024).

6. Egghead.io. Styling React Components with Emotion. URL: https://egghead.io/courses/styling-react-components-with-emotion (дата звернення: 01.06.2024).

7. ExpressJs Fast, unopinionated, minimalist web framework for Node.js, URL: https://expressjs.com/ (дата звернення: 09.05.2024).

8. Gurin S. JWT Handbook. Auth0, 2019. 75 с.

9. Large language Models, LLMs, URL: https://www.it.ua/knowledge-base/technology-innovation/large-language-models-llms (дата звернення: 05.05.2024).

10. Learn React, URL: https://react.dev/learn (дата звернення: 09.05.2024).

11. OpenAI – розпізнавання образів та відео у реальному часі – переваги та можливості, URL: https://mediacom.com.ua/openai-rozpiznavannya-obraziv-ta-video-u-realnomu-chasi-perevagi-ta-mozhlivosti/ (дата звернення: 19.05.2024).

12. OpenAI. API Documentation. URL: https://beta.openai.com/docs/ (дата звернення: 29.04.2024).

13. Prisma. Documentation. URL: https://www.prisma.io/docs/ (дата звернення: 05.05.2024).

14. The fastest and most powerful platform for building AI products, URL: https://openai.com/api/ (дата звернення: 01.05.2024).

15. Udemy. Databases with Prisma v2. URL: https://www.udemy.com/course/databases-with-prisma-v2/ (дата звернення: 12.05.2024).

16. Udemy. NestJS - Building Efficient, Reliable and Scalable Server-Side Applications. URL: https://www.udemy.com/course/nestjs-building-efficient-scalable-applications/ (дата звернення: 02.05.2024).

17. Бондаренко І. С. Системи розпізнавання зображень на основі моделей CLIP / І. С. Бондаренко // Вісник ОНУ. – 2020. – № 7. – С. 63-70.

18. Вебсайт з кулінарії «Клопотенко», URL: https://klopotenko.com/ (дата звернення: 04.05.2024).

19. Вебсайт з кулінарії «Пательня», URL: https://patelnya.com.ua/ (дата звернення: 05.05.2024).

20. Громов В. І. Основи розробки інтерфейсу користувача / В. І. Громов. Київ: Видавництво КПІ, 2020. 55 с.

21. Ковальчук І. М. Функціональність вебзастосунків: підхід до проектування / І. М. Ковальчук. Львів: ЛНУ, 2018. 28 с.

22. Левченко О. Г. Модульна архітектура програмного забезпечення: навчальний посібник / О. Г. Левченко. Київ: НАУ, 2016. 36 с.

23. Лисенко О. В. Сучасні методи автентифікації у серверних системах / О. В. Лисенко. Одеса: ОНАПР, 2020. 15 с.

24. Розуміння моделей OpenAI та їхнього впливу на бізнес, URL: https://www.ranktracker.com/uk/blog/understanding-open-ai-models-and-their-impact-on-businesses/ (дата звернення: 13.05.24).

25. Як працює ChatGPT простими словами, URL: https://drukarnia.com.ua/articles/yak-pracyuye-chatgpt-prostimi-slovami-MtpBS (дата звернення: 03.05.24).

ДОДАТОК А

Лістинг програми (головні файли)

schema.prisma

```
generator client {
  provider        = "prisma-client-js"
  previewFeatures = ["metrics"]
  binaryTargets   = ["native", "linux-musl-openssl-3.0.x", "darwin", "darwin-arm64"]
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id          String   @id @default(uuid())
  createdAt   DateTime @default(now())
  email       String   @unique
  password    String
  firstName   String
  lastName    String
  isActivated Boolean  @default(false)
  receipes    Recipe[]

  userConfirmation UserConfirmation?
  chat             Chat[]
}

model UserConfirmation {
  id             String    @id @default(uuid())
  expirationCode DateTime?
  code           String?
  newPassword    String?
  newEmail       String?
  userId         String    @unique
  user           User      @relation(fields: [userId], references: [id])
}

model Recipe {
  id          String   @id @default(uuid())
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
  title       String
  instructions String
  ingredients String

  user   User?   @relation(fields: [userId], references: [id])
  userId String?

  chat   Chat?   @relation(fields: [chatId], references: [id])
  chatId String?

  category   Category? @relation(fields: [categoryId], references: [id])
  categoryId String?

    imgSrc          String @default("")
```

```
}

model Category {
  id        String   @id @default(uuid())
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  name String

  recipe Recipe[]
}

model Chat {
  id              String   @id @default(uuid())
  createdAt       DateTime @default(now())
  updatedAt       DateTime @updatedAt
  name            String
  messagesQuantity Int      @default(0)
  user            User     @relation(fields: [userId], references: [id])
  userId          String
  recipe          Recipe[]
}
```

chat.service.ts

```
import { HttpException, HttpStatus, Inject, Injectable } from '@nestjs/common'
import type { Chat } from '@prisma/client'
import { PrismaService } from 'nestjs-prisma'
import Redis from 'ioredis'

import { OpenaiService } from '../openai/services/openai.service'
import type { IGetChatRes, IMessage, INewMsg, INewMsgRes } from './types'
import { FileService } from '../file/file.service'
import { CHAT_ERRORS } from './const'
import type { IChatResponse } from '../openai/types/response.types'

@Injectable()
export class ChatService {
   constructor(private readonly prisma: PrismaService,
     private readonly openai: OpenaiService,
     private readonly file: FileService,
     @Inject('REDIS_CLIENT') private readonly redisClient: Redis
   ) {}

   public async createChat(userId: string, name: string): Promise<Chat> {
      return this.prisma.chat.create({
         data: {
            name,
            userId
         },
      })
   }

   public async getChat(userId: string, id: string): Promise<IGetChatRes> {
      const chat = await this.prisma.chat.findUnique({
         where: {
            id,
         },
      })

      if (!chat) {
         throw new HttpException(CHAT_ERRORS, HttpStatus.NOT_FOUND)
      }
```

```
        if (chat.userId !== userId) {
            throw new HttpException(CHAT_ERRORS.YOU_HAVE_NO_ACCESS, HttpStatus.FORBIDDEN)
        }

        const key = `chat:${id}`

        const messages = await this.redisClient.lrange(key, 0, chat.messagesQuantity - 1)

        return {
            chat,
            messages: messages.reverse(),
        }
    }

    public async getAllUsersChats(userId: string): Promise<Array<Chat>> {
        return this.prisma.chat.findMany({
            where: {
                userId,
            },
        })
    }

    public async newMsg(data: INewMsg, userId: string): Promise<INewMsgRes> {
        const chat = await this.prisma.chat.findUnique({
            where: {
                id: data.chatId,
            },
        })

        if (!chat) {
            throw new HttpException(CHAT_ERRORS.CHAT_NOT_FOUND, HttpStatus.NOT_FOUND)
        }

        if (chat.userId !== userId) {
            throw new HttpException(CHAT_ERRORS.YOU_HAVE_NO_ACCESS, HttpStatus.FORBIDDEN)
        }

        let res: IChatResponse
        let answerPayload: string
        let questionPayload: string
        const deep = chat.messagesQuantity - 6 > 0 ?
chat.messagesQuantity - 6 :
0
        const messagesRaw = await this.redisClient.lrange(`chat:${chat.id}`, 0 , deep)

        const history = messagesRaw.map((message) => {
            const parsedMsg: IMessage = JSON.parse(message)

            return parsedMsg.text
        })

        if (data.image) {
            const uploadedFile = await this.file.uploadFile(data.image, 'chat')

            const base64Image = data.image.buffer.toString('base64')
            const url = `data:${data.image.mimetype};base64,${base64Image}`

            res = await this.openai.createPrompt(data.msg,history, url)

            answerPayload = JSON.stringify({
```

```
            text: res.choices[0]?.message.content ?? ''
        })

        questionPayload = JSON.stringify({
            text: data.msg,
            imgUrl: uploadedFile.publicUrl,
        })
    } else {
        res = await this.openai.createPrompt(data.msg, history)
        questionPayload = JSON.stringify({
            text: data.msg,
        })

        answerPayload = JSON.stringify({
            text: res.choices[0]?.message.content ?? ''
        })
    }

    const key = `chat:${data.chatId}`
    await this.redisClient.lpush(key, questionPayload)
    await this.redisClient.lpush(key, answerPayload)

    const response = {
        question: JSON.parse(questionPayload),
        answer: JSON.parse(answerPayload),
    }

    await this.prisma.chat.update({
        where: {
            id: chat.id
        },
        data: {
            messagesQuantity: {
                increment: 2
            }
        }
    })

    return response
}

public async deleteChat(userId: string, id: string): Promise<void> {
    const chat = await this.prisma.chat.findUnique({
        where: {
            id,
        },
    })

    if (!chat) {
        throw new HttpException(CHAT_ERRORS.CHAT_NOT_FOUND, HttpStatus.NOT_FOUND)
    }

    if (chat.userId !== userId) {
        throw new HttpException(CHAT_ERRORS.YOU_HAVE_NO_ACCESS, HttpStatus.FORBIDDEN)
    }

    await this.redisClient.del(`chat:${chat.id}`)
    await this.prisma.chat.delete({
        where: {
            id,
        },
```

```
    })
  }
}
```

auth.service.ts

```
import {  HttpException, HttpStatus, Injectable } from '@nestjs/common'
import type { Request } from 'express'
import type { User } from '@prisma/client'
import { JwtService as PassportService } from '@nestjs/jwt'
import * as bcrypt from 'bcrypt'

import { JWTService } from './jwt.service'
import { UserService } from '../../user/services/user.service'
import { MailService } from 'src/modules/mail/mail.service'
import { ConfigService } from '@nestjs/config'
import type { SetNewPasswordDto, SignInDto, SignUpDto } from '../dto'
import { ERRORS } from '../const'
import { isPasswordValid } from 'src/utils'

@Injectable()
export class AuthService {
  constructor(
    private readonly userService: UserService,
    private readonly jwtService: JWTService,
    private readonly passportService: PassportService,
    private readonly mailService: MailService,
    private readonly config: ConfigService,
  ) {}

  public async signUp(data: SignUpDto,): Promise<void> {
    const existingUser = await this.userService.getUser({
      email: data.email,
    })

    if (existingUser && existingUser.isActivated) {
      throw new HttpException(
        {
          email: ERRORS.USER_ALREADY_EXIST,
        },
        HttpStatus.BAD_REQUEST,
      )
    } else if (existingUser && !existingUser.isActivated) {
      await this.userService.deleteOne(existingUser.id)
    }

    const hash = await bcrypt.hash(data.password, 10)

    const { code, user } = await this.userService.createUser({
      ...data,
      password: hash,
    })

    await this.mailService.sendAccountActivationCode(user.email, code)
  }

  public async activateAccount(code: string): Promise<User> {
```

```
    const { user } = await this.userService.checkCode(code)
    const updUser = await this.userService.activate(user.id)
    return updUser
  }

  public async signIn({ email, password }: SignInDto): Promise<User> {
    const user = await this.userService.getUser({
      email,
    })

    if (!user) {
      throw new HttpException(
        {
          email: ERRORS.USER_NOT_EXIST,
        },
        HttpStatus.NOT_FOUND,
      )
    } else if (!user.isActivated) {
      throw new HttpException(
        {
          email: ERRORS.USER_NOT_ACTIVATED,
        },
        HttpStatus.FORBIDDEN,
      )
    }

    const isPasswordValid = await bcrypt.compare(password, user.password!)

    if (!isPasswordValid) {
      throw new HttpException(
        {
          password: ERRORS.PASSWORD_INVALID,
        },
        HttpStatus.NOT_FOUND,
      )
    }
    return user
  }

  public async checkJWT(req: Request): Promise<{ auth: boolean; user: User | null }> {
    if (Boolean(req.cookies['jwt'])) {
      const decodedJWT = this.passportService.decode(req.cookies['jwt']) as { id: string }

      const user = await this.userService.getUser({
        id: decodedJWT.id,
      })
      return {
        auth: Boolean(user),
        user,
      }
    }
    return {
      auth: false,
      user: null,
    }
  }
```

```
  public async forgotPassword(email: string): Promise<boolean> {
    const user = await this.userService.getUser({
      email,
    })

    if (!user) {
      throw new HttpException({
        message: ERRORS.EMAIL_NOT_FOUND,
        email: ERRORS.EMAIL_NOT_FOUND,
      }, HttpStatus.NOT_FOUND)
    } else if (!user.isActivated) {
      throw new HttpException({
        message: ERRORS.USER_NOT_ACTIVATED,
        email: ERRORS.USER_NOT_ACTIVATED,
      }, HttpStatus.UNAUTHORIZED)
    }

    const code = await this.userService.createCode(user.id)

    const resetLink = `${this.config.get<string>('FRONTEND_URL')!}/auth/password-reset/${code}`
      await this.mailService.sendPasswordResetLink(email, resetLink, 'Hello. This email is for your email verification.')
    return true
  }

  public async setNewPassword({code, newPassword}: SetNewPasswordDto): Promise<boolean> {
    const { user } = await this.userService.checkCode(code)

    const isPasswordsSame = await isPasswordValid(newPassword, user.password)

    if (isPasswordsSame) {
      throw new HttpException({
        message: ERRORS.PASSWORD_CANNOT_BE_THE_SAME,
        newPassword: ERRORS.PASSWORD_CANNOT_BE_THE_SAME,
      }, HttpStatus.BAD_REQUEST)
    }

    const hash = await bcrypt.hash(newPassword, 10)

    await this.userService.updateUser({
      where: {
        id: user.id,
      },
      data: {
        password: hash,
      },
    })
    return true
  }
}
```