МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЕКТ

на тему

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ОБМІНУ КУЛІНАРНИМИ РЕЦЕПТАМИ ТА ПОРАДАМИ

здобувача Клименка Олексія Володимировича _____

Науковий керівник:
к.е.н., доцент
_____Гордієнко І.В.

Проект допущений до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент
_____Тішков Б.О.

Київ 2024

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

ОСВІТНЬО_ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:	
Керівник проектної групи(гарант)	ЗАТВЕРДЖУЮ:
освітньо-професійної програми	Завідувач кафедри
_____Іванченко Г.Ф.	_____Тішков Б.О.
"____" _____2024 р.	"____" _____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти ***Клименка Олексія Володимировича***
очної (денної) форми навчання
на підготовку кваліфікаційного бакалаврського проекту
на тему: «Розробка веб-застосунку для обміну кулінарними рецептами та порадами»

Тему затверджено наказом ректора Університету від «_11_» _березня_ 2024 р. № 529-ст.

Кваліфікаційний бакалаврський проект виконується на матеріалах
___наукових публікацій, інтернет-ресурсів, технічної документації_____

План кваліфікаційного бакалаврського проекту

Розділ І *_Характеристика та аналіз предметної галузі_____*

Розділ ІІ *_Розробка вимог і моделювання інформаційної системи_____*

Розділ ІІІ__*Проектування та реалізація компонентів системи_____*

Об'єкт дослідження *_процеси ефективного управління та документообігу в галузі інформаційної інтернет-підтримки кулінарії*

Предмет дослідження *_сукупність теоретичних, методичних і прикладних аспектів з моделювання, проєктування та програмування модулів для підвищення ефективності обміну кулінарною інформацією з використанням веб-технологій*

Мета кваліфікаційного бакалаврського проекту *_систематизація і узагальнення сучасного досвіду з розробки нових рішень в галузі обміну кулінарною інформацією та розробка власного прототипу веб-застосунку*

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі I *__ Описати предметну галузь інформаційної інтернет-підтримки кулінарії. Навести характеристику об'єкта дослідження. Подати результати аналізу літературних джерел та існуючих варіантів інформаційних систем і технологій для інформаційної інтернет-підтримки кулінарії* _____
_____
_____
_____
_____

У розділі II *___Визначити бізнес-вимоги, функціональні та нефункціональні вимоги до проєктованої системи та подати їх у формі діаграм і проєктної специфікації, розроблених з використанням CASE-засобів. Подати постановку та алгоритм розв'язання задачі автоматизації обміну кулінарними рецептами та порадами з використанням веб-технологій. Розробити моделі системи у специфікації мови UML*
_____
_____
_____
_____

У розділі III *__ Спроектувати базу даних та елементи інформаційного забезпечення веб-застосунку для обміну кулінарними рецептами та порадами. Обгрунтувати вибір та розташування комплексу технічних засобів на об'єкті управління. Описати структуру та складові програмного забезпечення веб-застосунку. Висвітлити результати реалізації веб-застосунку для обміну кулінарними рецептами та порадами.* _____
_____
_____
_____
_____
_____

Завдання підготував
науковий керівник _____ **Гордієнко Ірина Василівна**
"_____"_____ 2024 р.

Завдання одержав
студент _____ **Клименко Олексій Володимирович**

"_____"_____ 2024 р

Відгук
про кваліфікаційний бакалаврський проєкт
здобувача навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

Клименка Олексія Володимировича
на тему: ***«Розробка веб-застосунку для обміну кулінарними рецептами та порадами»***

1. **Актуальність теми**: Останнім часом в інтернет-просторі зростає популярність різноманітних тематичних сайтів, що містять корисну інформацію з певного напряму діяльності та забезпечують користувачам можливість обміну цією інформацією між собою. Існують різноманітні підходи до проєктування подібних сайтів та інформаційні технології для їх реалізації. Тому тематика кваліфікаційного бакалаврського проєкту Клименка О.В., присвяченого дослідженню галузі обміну кулінарними рецептами та проєктуванню сучасного веб-застосунку для підтримки такого обміну є актуальною.
2. **Позитивні риси кваліфікаційного бакалаврського проєкту**: Кваліфікаційний проєкт містить результати дослідження здобувачем предметної області кулінарії, відображає сучасні підходи до розробки веб-застосунку для обміну кулінарною інформацією, демонструє володіння інформаційно-технологічним інструментарієм розробки ІС, зокрема, Python 3.11., СУБД SQLite, фреймворком Django 4.2.5.
3. **Наявність самостійних розробок автора**: Здобувач самостійно глибоко проаналізував і визначив вимоги до веб-застосунку для обміну кулінарними рецептами та порадами, спроєктував відповідні діаграми і специфікації, підготував постановку і алгоритм розв'язання задачі обміну кулінарними рецептами та порадами, здійснив моделювання веб-застосунку та розробку основних видів забезпечення системи.
4. **Цінність теоретичних висновків та практичних рекомендацій**: Розроблений здобувачем веб-застосунок відповідає вимогам до інформаційної системи і може бути рекомендований для практичного використання. .
5. **Наявність недоліків**: Здобувач виправив недоліки, які було виявлено під час роботи над проєктом.
6. **Загальна оцінка кваліфікаційного бакалаврського проєкту та його допущення до захисту перед ЕК**: Кваліфікаційний бакалаврський проєкт відповідає методичним вимогам, заслуговує високої оцінки і може бути допущений до захисту перед ЕК.

Науковий керівник _____ доцент, к.е.н. Гордієнко І.В.
"___"_____ 2024 р.

Рецензія
на кваліфікаційний бакалаврський проект
Клименка Олексія Володимировича

тема
«Розробка веб-застосунку для обміну кулінарними рецептами та порадами»

Актуальність теми з розробки кулінарної веб-платформи зумовлена зростаючим інтересом до кулінарії, здорового харчування та онлайн-спільнот. Доцільність розроблення такої платформи полягає у створенні інноваційного інструменту, який підвищує доступність кулінарної інформації, покращує взаємодію між користувачами та стимулює обмін досвідом. Платформа забезпечує зручний доступ до рецептів, їх систематизацію за різними категоріями, можливість коментування та оцінювання, що підвищує якість контенту.

Якість проведеного дослідження відзначається високим рівнем аналізу сучасних тенденцій у галузі веб-розробки та потреб користувачів кулінарних платформ. Виконано ретельне дослідження існуючих рішень, визначено їх переваги та недоліки, що дозволило розробити конкурентоспроможний продукт. Застосовані методи дослідження були обґрунтованими та відповідали поставленим завданням.

Серед позитивних рис проекту слід відзначити інтуїтивно зрозумілий інтерфейс платформи, високу швидкість роботи та надійність системи. Використання сучасних технологій, таких як мова програмування Python та фреймворк Django, забезпечує стабільність та безпеку роботи платформи. Крім того, платформа підтримує інтерактивні функції, що підвищують зручність використання для користувачів.

Серед зауважень варто зазначити необхідність подальшого вдосконалення алгоритмів пошуку та рекомендацій рецептів для забезпечення більшої точності та релевантності результатів. Також слід звернути увагу на можливість інтеграції з соціальними мережами для полегшення реєстрації та входу користувачів.

Практична значимість висновків і рекомендацій проекту полягає в їхньому безпосередньому застосуванні для покращення роботи веб-платформи. Запропоновані рішення та рекомендації можуть бути використані для подальшого розвитку платформи, залучення більшої кількості користувачів та підвищення якості наданих послуг. Це дозволить створити конкурентоспроможний продукт, що задовольнить потреби сучасного ринку кулінарних веб-платформ.

Директор ТОВ «Супердрастик» _____ Гаркуша Д. В.

АНОТАЦІЯ

кваліфікаційного бакалаврського проекту студента 4 курсу
Навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
Клименка Олексія Володимировича, виконаного на тему:
«Розробка веб-застосунку для обміну кулінарними рецептами та порадами»
Київ: кафедра інформаційних систем в економіці, 2024 р.

Кваліфікаційний бакалаврський проект присвячений систематизації і узагальненню сучасного досвіду з розробки нових рішень в галузі обміну кулінарною інформацією та розробці власного прототипу веб-застосунку з використанням сучасних інформаційних технологій.

Кваліфікаційний бакалаврський проект складається з трьох розділів які пов’язані між собою.

В першому розділі описана предметна галузь інформаційної інтернет-підтримки кулінарії. Наведена характеристика об’єкта дослідження. Подані результати аналізу літературних джерел та існуючих варіантів інформаційних систем і технологій для інформаційної інтернет-підтримки кулінарії.

Другий розділ визначає бізнес-вимоги, функціональні та нефункціональні вимоги до проєктованої системи та подає їх у формі діаграм і проєктної специфікації, розроблених з використанням CASE-засобів. Подається постановка та алгоритм розв’язання задачі автоматизації обміну кулінарними рецептами та порадами з використанням веб-технологій. Описані та показані моделі системи у специфікації мови UML.

В третьому розділі спроектовано базу даних та елементи інформаційного забезпечення веб-застосунку для обміну кулінарними рецептами та порадами. Обгрунтувано вибір та розташування комплексу технічних засобів на об’єкті управління. Описано структуру та складові програмного забезпечення веб-застосунку. Висвітлено результати реалізації веб-застосунку для обміну кулінарними рецептами та порадами.

РЕФЕРАТ

Кваліфікаційний бакалаврський проект містить 109 сторінок, 12 таблиць, 19 рисунків, список літератури з 14 найменувань, 7 додатків.

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ОБМІНУ КУЛІНАРНИМИ РЕЦЕПТАМИ ТА ПОРАДАМИ

Перелік ключових слів: кулінарний рецепт, веб-платформа, обмін інформацією, база даних, Django, автоматизація, інформаційна система.

Об’єктом дослідження є процеси ефективного управління та документообігу в галузі інформаційної інтернет-підтримки кулінарії

Предметом дослідження є сукупність теоретичних, методичних і прикладних аспектів з моделювання, проєктування та програмування модулів для підвищення ефективності обміну кулінарною інформацією з використанням веб-технологій.

Метою кваліфікаційного бакалаврського проекту є систематизація і узагальнення сучасного досвіду з розробки нових рішень в галузі обміну кулінарною інформацією та розробка власного прототипу веб-застосунку

Завданнями кваліфікаційного бакалаврського проекту є розробка веб-платформи для обміну кулінарними рецептами та порадами, яка забезпечує можливість перегляду, додавання, коментування і оцінювання рецептів, а також адміністрування контенту і користувачів.

При проектуванні використовувалися такі програмні та апаратні засоби:

- програмне середовище Python 3.11.1;
- фреймворк Django 4.2.5;
- система керування базами даних SQLite
- операційна система Windows 10

Результати досягнуті в процесі роботи включають розробку функціональної веб-платформи для обміну кулінарними рецептами та

порадами. Система дозволяє користувачам переглядати, додавати, коментувати та оцінювати рецепти, а адміністраторам управляти контентом і користувачами. Новизна проекту полягає у створенні інтегрованої системи, яка об'єднує всі ці функції в одному зручному інтерфейсі та використовує українську мову.

Одержані результати можуть бути застосовані для створення програмного забезпечення, яке автоматизує процеси на кулінарних платформах.

Рік виконання кваліфікаційного бакалаврського проекту – 2024.

Рік захисту кваліфікаційного бакалаврського проекту – 2024.

ЗМІСТ

ВСТУП

У сучасному світі кулінарія є не лише засобом задоволення базових потреб, але й важливою складовою культурного життя суспільства. З розвитком інтернет-технологій спостерігається зростання популярності онлайн-спільнот, де користувачі можуть ділитися кулінарними рецептами та порадами. Веб-платформи для обміну рецептами сприяють формуванню спільнот однодумців, обміну знаннями та досвідом, що робить їх важливими соціальними інструментами. Зважаючи на це, розробка веб-платформи для обміну кулінарними рецептами та порадами є актуальною та значущою темою для дослідження.

Актуальність теми полягає в необхідності створення зручного та ефективного інструменту для обміну кулінарними знаннями. Існуючі платформи часто мають обмежені можливості або незручний інтерфейс, що ускладнює користування ними. Нова веб-платформа, розроблена з використанням сучасних технологій, таких як мова програмування Python та фреймворк Django, забезпечить високу надійність, безпеку та ефективність роботи, що сприятиме покращенню взаємодії між користувачами та підвищенню якості контенту.

Об'єктом дослідження є процеси ефективного управління та документообігу в галузі інформаційної інтернет-підтримки кулінарії. Предметом дослідження виступає сукупність теоретичних, методичних і прикладних аспектів з моделювання, проєктування та програмування модулів для підвищення ефективності обміну кулінарною інформацією з використанням веб-технологі.

Метою цього дослідження є систематизація і узагальнення сучасного досвіду з розробки нових рішень в галузі обміну кулінарною інформацією та розробка власного прототипу веб-застосунку.

Для досягнення цієї мети необхідно виконати наступні завдання:

• Аналіз існуючих платформ: Провести дослідження наявних платформ для обміну кулінарними рецептами, визначити їхні переваги та недоліки, а також виявити можливості для покращення.

• Проектування бази даних: Розробити інфологічну та даталогічну моделі бази даних, які забезпечуватимуть ефективне зберігання, організацію та обробку кулінарних рецептів та інформації про користувачів.

• Розробка веб-платформи: Створити веб-застосунок, використовуючи мову програмування Python та фреймворк Django, який буде відповідати сучасним стандартам веб-розробки та забезпечуватиме зручний інтерфейс для користувачів.

• Забезпечення безпеки та надійності: Впровадити сучасні методи захисту даних для забезпечення безпеки інформації користувачів та стабільної роботи платформи.

• Тестування та налагодження: Провести ретельне тестування веб-платформи, виявити та усунути можливі помилки, щоб забезпечити її надійне та безперебійне функціонування.

Ці завдання дозволять створити інноваційну веб-платформу, яка не тільки задовольняє потреби користувачів у зручному доступі до кулінарних рецептів, але й сприяє активному обміну кулінарними знаннями та досвідом між учасниками спільноти.

Для вирішення поставлених завдань були використані наступні методи дослідження: аналіз літературних джерел та існуючих платформ для обміну кулінарними рецептами, методи математичного та статистичного аналізу для обробки даних, структурно-логічний підхід для побудови моделі бази даних, а також вербально-описовий метод для документування процесу розробки та результатів.

Теоретична значущість роботи полягає у розширенні знань про процеси розробки веб-платформ для обміну інформацією, а також у дослідженні можливостей використання фреймворку Django для створення складних

інформаційних систем. Практична значущість полягає у створенні реальної веб-платформи, яка може бути використана для обміну кулінарними рецептами та порадами, що сприятиме формуванню онлайн-спільноти однодумців та покращенню кулінарних навичок користувачів.

У цій роботі розглядаються всі необхідні етапи розробки веб-платформи – від аналізу існуючих рішень до тестування та впровадження готової системи, що робить цей проект важливим внеском у галузь інформаційних технологій та кулінарії.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об’єкта дослідження

Економічний зміст задач, пов'язаних з веб платформою для обміну кулінарними рецептами і порадами, включає кілька ключових аспектів, які мають вирішальне значення для прийняття рішень. Основним завданням є забезпечення ефективного обміну кулінарною інформацією між користувачами, що сприяє підвищенню кулінарної обізнаності та вдосконаленню навичок приготування їжі. Платформа повинна бути орієнтована на зручність використання, швидкий доступ до потрібної інформації та підтримку активної взаємодії між користувачами різних рівнів.

Одним із важливих економічних аспектів є монетизація платформи через рекламу. Це означає розробку моделі, яка дозволить ефективно розміщати рекламу на сайті або додатку таким чином, щоб вона була вигідною як для рекламодавців, так і для користувачів. Паралельно важливо забезпечити безпеку та конфіденційність даних користувачів, щоб зберегти їхню довіру та лояльність.

Наголос також робиться на підтримці якості контенту та спільноти користувачів, що допомагає платформі зберігати свою репутацію та приваблювати нових користувачів. Усі ці аспекти вимагають ефективного управління ресурсами та неперервного вдосконалення для досягнення успіху на цьому конкурентному ринку.

У предметну галузь веб-платформи входить широкий спектр об'єктів, пов'язаних з кулінарією та інформаційними технологіями. Одним з основних об'єктів є кулінарні веб-сайти та блоги, які спеціалізуються на публікації рецептів, кулінарних порад та оглядів кулінарних технік і продуктів. Вони служать як джерела інформації для любителів кулінарії та професіоналів,

допомагаючи їм розширювати свої знання та вдосконалювати навички приготування їжі.

Соціальні медіа платформи, такі як Facebook, Instagram та Pinterest, також є важливими об'єктами у цій галузі. Вони дозволяють користувачам ділитися рецептами, фотографіями приготованих страв, а також обговорювати кулінарні питання з великою аудиторією. Ці платформи сприяють швидкому поширенню кулінарних ідей та трендів, забезпечуючи інтерактивність та залученість користувачів.

Іншим значущим об'єктом є кулінарні школи та навчальні заклади, які пропонують курси та майстер-класи з кулінарії. Вони можуть співпрацювати з веб платформою, надаючи професійний контент та навчальні матеріали для користувачів. Кулінарні фестивалі та заходи також входять до предметної галузі, оскільки вони служать майданчиком для презентації нових рецептів і технік приготування їжі, а також для обміну досвідом між кулінарами.

Нарешті, технологічні об'єкти, такі як програмні засоби для розробки веб додатків, бази даних, хмарні сервіси та системи управління контентом, є невід'ємною частиною предметної галузі. Вони забезпечують технічну основу для функціонування платформи, підтримуючи її стабільність, безпеку та продуктивність. Ці об'єкти допомагають створити зручну та ефективну систему для обміну кулінарною інформацією та взаємодії користувачів.

У предметній галузі платформи рішення приймаються з метою забезпечення ефективного та зручного користування платформою, залучення та утримання користувачів, а також монетизації сервісу через рекламу.

Проблемні ситуації можуть включати в себе зменшення активності користувачів, низьку якість контенту, проблеми з безпекою даних, технічні неполадки тощо. Ціллю їх розв'язання є створення привабливого та безпечного середовища для обміну рецептами та порадами, забезпечення стабільної роботи платформи та залучення нових користувачів.

Особи, що приймають рішення, можуть включати в себе розробників платформи, менеджерів продукту, адміністраторів, модераторів та інших працівників. Розробники відповідають за технічні аспекти платформи, менеджери продукту визначають стратегію розвитку сервісу, а адміністратори та модератори контролюють контент та взаємодію з користувачами.

Переваги різних осіб включають в себе технічну експертизу розробників, стратегічне мислення та аналітичні здібності менеджерів продукту, а також спеціалізовані знання та емпатію адміністраторів та модераторів.

Принципи узгодження індивідуальних переваг в разі групового прийняття рішень можуть включати в себе діалог, співпрацю та компроміс. Важливо, щоб кожна сторона мала можливість висловити свої погляди та переваги, а потім спільно шукати найкращий варіант розв'язання проблеми, що задовольняв би всіх учасників.

Фактори які впливають на середовище прийняття рішень можуть бути різноманітні. Перш за все, потреби користувачів мають величезний вплив. Їх вимоги, очікування та звички використання платформи визначають способи розвитку та функціонал платформи. Наприклад, якщо користувачі вимагають покращення у функціоналі або більше можливостей для взаємодії, це може вплинути на прийняття рішень щодо розробки нових функцій чи змін у дизайні.

Технологічні обмеження також відіграють суттєву роль у процесі прийняття рішень. Технічні можливості платформи обмежують впровадження певних функцій та розвиток нових можливостей. Наприклад, якщо платформа обмежена з точки зору обробки даних або пропускної здатності, це може вплинути на можливість впровадження складних функцій або підвищення швидкості роботи платформи.

Крім того, конкуренція на ринку також має величезний вплив на прийняття рішень. Нові ідеї та функції можуть бути впроваджені для

збільшення конкурентоспроможності платформи та привертання більшої кількості користувачів. Такі фактори, як зміни відміток у споживачів або нові тренди в галузі кулінарії, також можуть впливати на напрямок розвитку платформи і сприйматися як можливості для нових рішень.

Взаємозв'язки рішень в контексті предметної галузі можуть мати величезне значення. Рішення, що приймаються на ранніх етапах розвитку платформи, можуть впливати на подальші стратегії та рішення. Наприклад, визначення цільової аудиторії та основних функцій платформи може вплинути на архітектуру та дизайн.

Крім того, рішення, прийняті в процесі взаємодії з користувачами та відгуків спільноти, також можуть мати великий вплив на подальший розвиток. Наприклад, якщо користувачі висловлюють певні проблеми або запити, це може призвести до змін у функціоналі або додаванню нових можливостей.

Крім внутрішніх взаємозв'язків, зовнішні фактори також можуть впливати на рішення. Наприклад, зміни в індустрії кулінарії або конкурентні стратегії можуть вимагати адаптації платформи та зміни стратегій розвитку.

Таким чином, сукупність рішень, які приймаються на всіх рівнях розвитку платформи, формує загальний шлях та стратегію розвитку. Попередні рішення можуть мати величезний вплив на майбутні стратегії та способи розвитку платформи, створюючи інноваційні можливості та відповідаючи на потреби користувачів які постійно змінюються.

Для прийняття обґрунтованих рішень необхідна як внутрішня, так і зовнішня інформація. Внутрішня інформація включає дані про активність користувачів на платформі, рівень використання різних функцій, реакції користувачів на нові функції або зміни, а також результати тестування нових функцій.

Зовнішня інформація може включати дані про тренди у галузі кулінарії, поведінку конкурентів, реакцію аудиторії на їхні стратегії, а також зміни в законодавстві або регулятивних вимогах.

Типи даних можуть бути як емпіричними, так і об'єктивними. Емпіричні дані можуть включати числові показники про активність користувачів, рейтинги рецептів, кількість коментарів тощо. Об'єктивні дані можуть включати дані про тренди на ринку, результати досліджень у галузі кулінарії, аналіз поведінки конкурентів тощо.

Джерела надходження даних можуть бути різноманітні, включаючи внутрішні бази даних платформи, зовнішні джерела, такі як аналітичні звіти, дослідження ринку, а також результати взаємодії з користувачами через опитування, фокус-групи або комунікацію у соціальних мережах.

Ситуації прийняття рішень можна класифікувати за різними критеріями, включаючи тип (ситуації закритих задач, ситуації відкритих задач або кризові ситуації) та вид (детерміновані, за умов ризику, за умов невизначеності, за умов нечітких цілей або посилені відкриті рішення).

Ситуації закритих задач можуть включати, наприклад, рутинні рішення, які базуються на певних стандартах або процедурах, таких як впровадження нової функціональності на основі вже існуючих технологій або аналіз даних для виявлення популярних рецептів.

Ситуації відкритих задач можуть виникати, коли потрібно прийняти рішення в умовах невизначеності або за умов ризику, коли інформація є неповною або невпевненою. Наприклад, визначення стратегії розвитку платформи або вибір нового функціоналу, який може вплинути на користувачів та конкурентоспроможність платформи.

Кризові ситуації можуть виникати при непередбачуваних обставинах, таких як технічні проблеми, кібератаки або негативна реакція громадськості на певні події. У таких ситуаціях необхідно приймати швидкі та рішучі дії для мінімізації збитків та відновлення довіри користувачів.

Види прийняття рішень можуть варіюватися від детермінованих, коли рішення приймаються на основі чітких правил та критеріїв, до ситуацій за умов невизначеності, коли інформація є обмеженою або недоступною, та

посилені відкриті рішення, коли ставки високі і не можна передбачити всі можливі наслідки рішення. Вибір підходу до прийняття рішень залежить від контексту та конкретної ситуації, а також від доступної інформації та ресурсів.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Аналіз літературних джерел та практичний досвід використання інформаційних систем і технологій в предметній галузі є важливою складовою сучасного наукового дослідження та професійної практики. Літературні джерела становлять основу для теоретичного аналізу та розуміння сутності проблеми, виявлення тенденцій розвитку та ідентифікації потенційних рішень. Практичний досвід використання ІС і технологій дозволяє перевірити теоретичні концепції на практиці, реалізувати їх у різних сферах діяльності та визначити ефективність застосування.

Під час аналізу літературних джерел важливо враховувати як класичні роботи, що заклали основи даної предметної галузі, так і нові наукові публікації та інноваційні розробки. Це дозволяє отримати повніше уявлення про сучасний стан та напрямки розвитку галузі. Крім цього, аналіз літературних джерел сприяє виявленню прогалин у наявних дослідженнях і встановленню нових напрямків для майбутніх наукових досліджень.

Практичний досвід використання інформаційних систем і технологій полягає в розробці та впровадженні конкретних інформаційних систем, програмних рішень та технологічних інструментів у різних сферах діяльності. Це може включати в себе створення програмних додатків для автоматизації бізнес-процесів, впровадження систем управління базами даних для зберігання та аналізу інформації, а також застосування різноманітних аналітичних інструментів для отримання нових знань та підтримки прийняття рішень.

В цьому аналізі буде розглянуто та порівняно існуючі інформаційні системи, які вже використовуються у сфері обміну кулінарними рецептами та порадами. Головною метою цього порівняння є визначення переваг та недоліків цих систем у порівнянні з нашим проектом. Буде зосереджено увагу на ключових аспектах, таких як повнота функціональних можливостей, економіко-математичні моделі та методи підтримки прийняття рішень, організація баз даних, технологічно-організаційні аспекти і якість інтерфейсу. Це дослідження допоможе зрозуміти, які аспекти вже реалізовані в існуючих системах та які можливості можна вдосконалити або внести в розроблюваний проект для забезпечення кращої ефективності й зручності для користувачів у сфері обміну кулінарними рецептами та порадами.

Розглянемо платформу "Allrecipes" (рис. 1.1) [1]. Цей сайт відомий своєю широкою та різноманітною кулінарною платформою, яка пропонує багато можливостей для користувачів у сфері готування їжі. Сайт включає в себе тисячі рецептів, представлених користувачами, що створює обширне кулінарне співтовариство. Серед основних переваг "Allrecipes" можна відзначити зручний інтерфейс, що дозволяє користувачам з легкістю знаходити, зберігати та експериментувати з рецептами. Функції, такі як створення списків покупок та персоналізованих рецептових книг, роблять процес готування ще зручнішим. Комунікативний характер сайту дозволяє користувачам обговорювати рецепти, ділитися враженнями та допомагати один одному в удосконаленні кулінарних навичок. Крім того, постійне оновлення новими ідеями та трендами в готуванні робить "Allrecipes" джерелом натхнення для кулінарів різного рівня. Ураховуючи ці позитивні аспекти, "Allrecipes" може бути цікавим для тих, хто шукає варіанти рецептів, спілкування та інноваційні ідеї в галузі кулінарії. Основною проблемою є відсутність української мови, що робить не дуже зручним використання даного сервісу для української спільноти.

Іншим прикладом буде "Smachno.ua" [2], що є важливим кулінарним ресурсом в Україні (рис. 1.2), який пропонує широкий вибір рецептів і кулінарних порад для користувачів з різними гастрономічними вподобаннями. Сайт визначається своєю різноманітністю страв, охоплюючи як традиційні українські рецепти, так і страви зі світової кухні. Однією з ключових переваг "Smachno.ua" є його зручний інтерфейс, який полегшує користувачам пошук та вибір рецептів. Сайт регулярно оновлюється новими рецептами та кулінарними статтями, що робить його цікавим та актуальним для широкого кола користувачів. Недоліком даної платформи є відсутність можливості додавання власних рецептів користувачами на сайт.

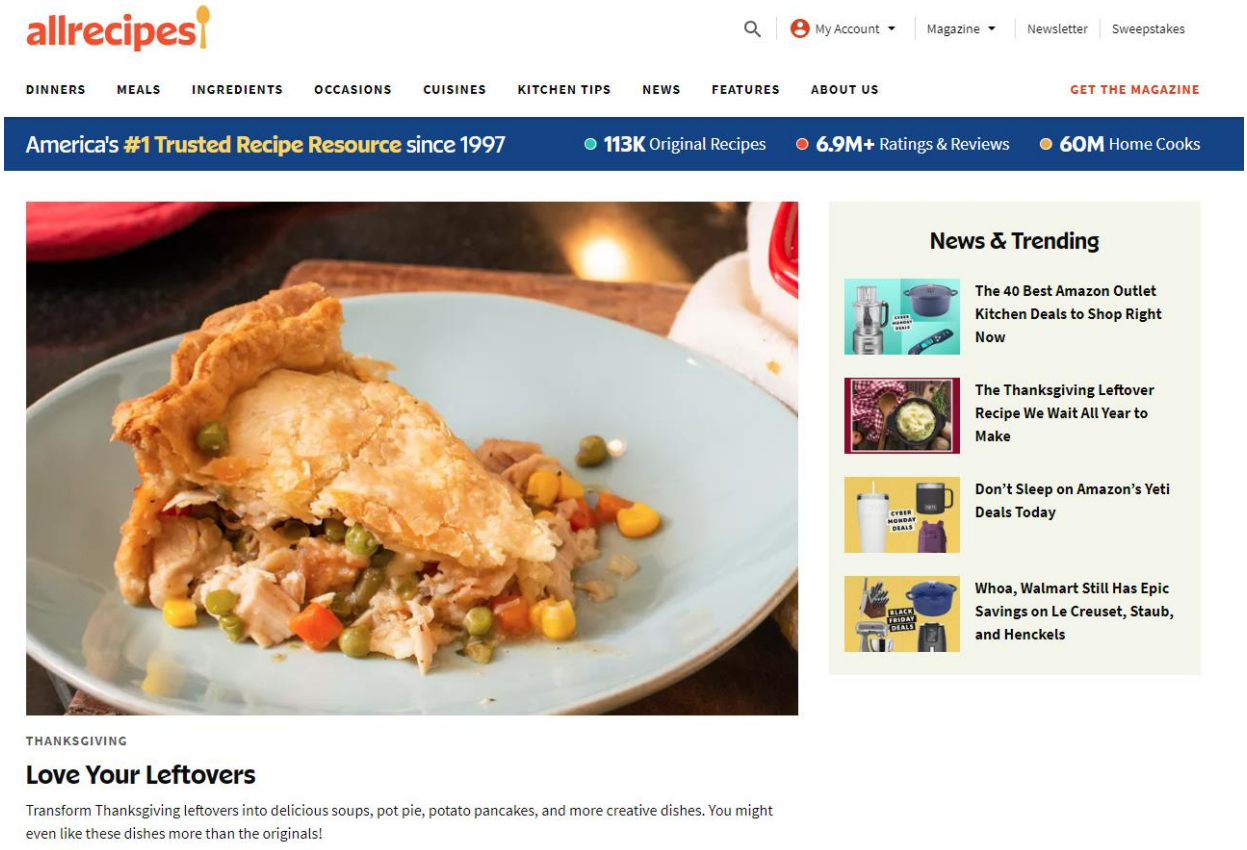


Рисунок 1.1 – Головна сторінка платформи Allrecipes або [1]

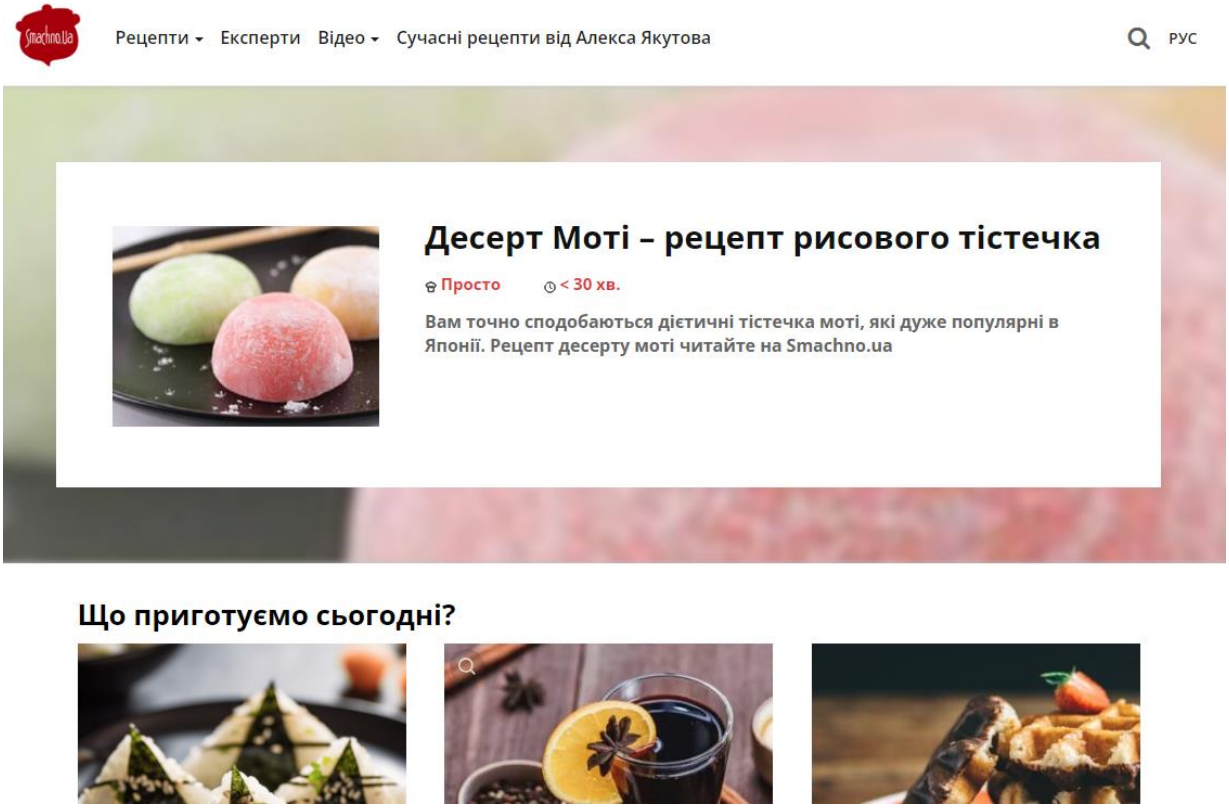


Рисунок 1.2 – Головна сторінка платформи Smachno.ua або [2]

Таблиця 1.1– Порівняльна характеристика існуючих інформаційних систем

Ключові аспекти	**Розроблюваний проект**	**Allrecipes**	**Smachno.ua**
повнота функціональних можливостей ІС	+	+	-
економіко-математичні моделі та методи підтримки прийняття рішень	+	+	+
ситуації, пов’язані з прийняттям рішень, функції та завдання прийняття рішень, на які орієнтована ІС	+	+	+
організація бази/сховища даних, програмного забезпечення	+	+	+
технологічно-організаційні аспекти	+	+	+

Продовження таблиці 1.1

якість інтерфейсу «ІС – користувач»	+	+	+
склад і форма результатів	+	+	+
новітні методи, технології та підходи до розроблення ІС	+	+	+
Україномовний інтерфейс	+	-	+

На основі ретельного аналізу кулінарних інформаційних систем, таких як Allrecipes та Smachno.ua, використованих у сфері обміну кулінарними рецептами та порадами, можна визначити кілька ключових висновків.

Зокрема, розроблюваний проект видається конкурентоспроможним у забезпеченні повноти функціональних можливостей. Позитивні аспекти спостерігаються в усіх важливих сферах, таких як економіко-математичні моделі, організація баз даних, технологічно-організаційні аспекти і якість інтерфейсу.

Allrecipes вражає своїм обсягом та актуальністю інформації, надаючи зручну систему фільтрації для користувачів під час пошуку рецептів. З іншого боку, Smachno.ua визначається своєю місією просування гастрономічних вражень та наданням повної інформації про кулінарію.

Цей порівняльний аналіз висвітлює, які аспекти вже успішно реалізовані в існуючих кулінарних системах, і які можливості можна подальше вдосконалити або внести до нашого проекту для забезпечення ще кращої ефективності та зручності для користувачів

РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної системи

У сучасному світі успішний бізнес вимагає не лише інноваційних ідей, але й точного аналізу вимог ринку. Вирішальним аспектом стає вміння відповідати зростаючим потребам споживачів, які постійно змінюються. Бізнес-вимоги охоплюють широкий спектр аспектів, від технологічних інновацій до стратегічного управління. Вони визначають якість продукту чи послуги, його конкурентоспроможність та сприятливе становище на ринку.

Нижче наведено ключові бізнес-вимоги до системи обміну кулінарними рецептами та порадами. Ці вимоги визначають стратегічні цілі та функціональні можливості системи, які необхідні для задоволення потреб користувачів та досягнення успішного розвитку проекту.

Бізнес-вимоги:

- Публікація та обмін рецептами: система повинна надавати можливість користувачам пропонувати свої кулінарні рецепти для обміну з іншими користувачами;
- Пошук та фільтрація рецептів: система повинна дозволяти користувачам здійснювати пошук рецептів за ключовими словами та мати можливість відсортовувати рецепти за категорією;
- Управління власними рецептами та улюбленими: можливість користувачів зберігати рецепти у своєму особистому обліковому записі та створювати список улюблених рецептів для зручного доступу;
- Забезпечення зручності та доступності: розробка зручного інтерфейсу користувача, який буде доступний з різних пристроїв та браузерів, забезпечуючи комфортне користування;

- Підтримка росту та масштабованість: розробка системи, яка може розширюватися та масштабуватися для забезпечення зростання кількості користувачів та обсягу даних;
- Комунікація та взаємодія користувачів: наявність коментарів до рецептів, де користувачі можуть обговорювати та ділитися власними думками та порадами.

Функціональні вимоги визначають, як система повинна функціонувати, щоб задовольнити потреби користувачів та виконувати свої завдання. Ці вимоги описують конкретні функції, операції та дії, які має забезпечити система. Вони можуть включати обробку даних, взаємодію з користувачем, а також внутрішні процеси системи.

Ключовим аспектом функціональних вимог є їх вимірюваність та перевірка. Вони повинні бути конкретними та однозначними, щоб розробники могли відповідно імплементувати їх у систему. Крім того, функціональні вимоги повинні бути відповідні бізнес-потребам та очікуванням користувачів.

Забезпечення якості функціональних вимог є важливим етапом у розробці будь-якої системи чи програмного продукту. Чітко сформульовані та вимірювані вимоги дозволяють забезпечити ефективну та надійну роботу системи, що відповідає потребам користувачів та бізнес-цілям.

Нижче подано детальний опис функціональних вимог, які визначають, як система повинна працювати та які функції вона повинна виконувати для забезпечення ефективності та зручності використання.

Функціональні вимоги:

- Перегляд рецептів: платформа повинна надавати користувачеві можливість переглядати всі наявні рецепти з детальною інформацією. Кожен рецепт має список інгредієнтів, інструкції щодо приготування, вказівки про час приготування та фотографії страви. Користувач може шукати рецепти за різними критеріями, такими як тип кухні, склад інгредієнтів або вид страви.

- Пошук рецептів: система повинна надавати можливість користувачам швидко знаходити потрібні страви за різними критеріями. Крім того, користувачі можуть відсортувати рецепти за категорією.
- Перегляд коментарів: користувач повинен мати змогу переглядати коментарі та відгуки інших користувачів під кожним рецептом. Це дозволяє отримати додаткову інформацію про страву, враження від її приготування та поради інших користувачів.
- Пропозиції рецепту: користувач повинен мати можливість пропонувати нові рецепти для включення до системи, щоб інші користувачі мали змогу ознайомитися з унікальними стравами та ідеями.
- Додавання коментарів: користувач повинен мати змогу залишати свої коментарі та відгуки під рецептами, висловлювати свої враження та думки щодо страви, ділитися порадами або просто виражати свою думку.
- Додавання рецептів в улюблені: користувач повинен мати можливість додавати певні рецепти до списку улюблених, щоб мати зручний доступ до них у майбутньому.
- Модерація контенту: адміністратор повинен мати можливість переглядати, редагувати та видаляти контент на платформі, зокрема рецепти, коментарі, та пропозиції рецептів, забезпечуючи відповідність правилам та стандартам.
- Управління користувачами: адміністратор повинен мати можливість керувати обліковими записами користувачів, здійснюючи такі дії як створення, блокування, або видалення облікових записів, а також призначення рівнів доступу.
- Публікація контенту: адміністратор повинен мати можливість додавати новий контент на платформу, включаючи рецепти, новини, або оголошення, забезпечуючи актуальність та рознообразіє інформації для користувачів.

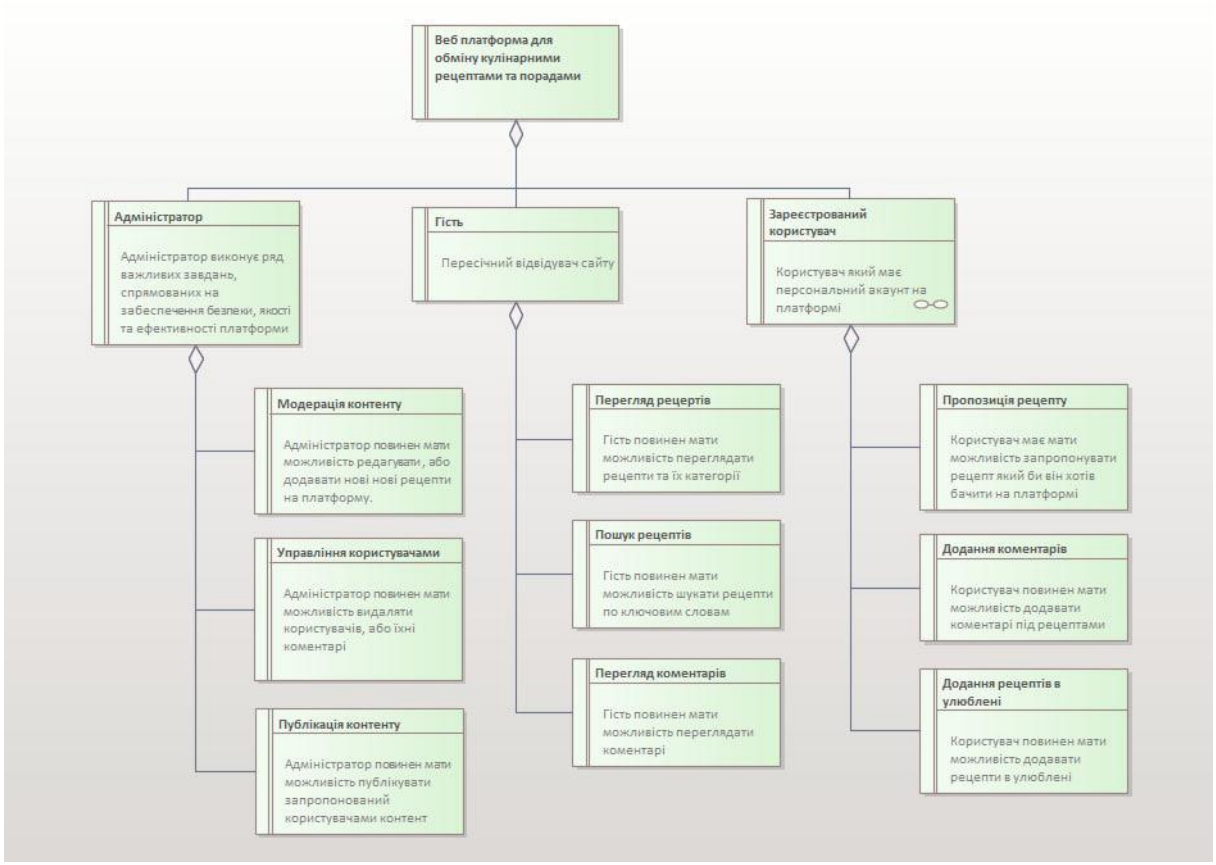


Рисунок 2.1 – Діаграма функціональних вимог

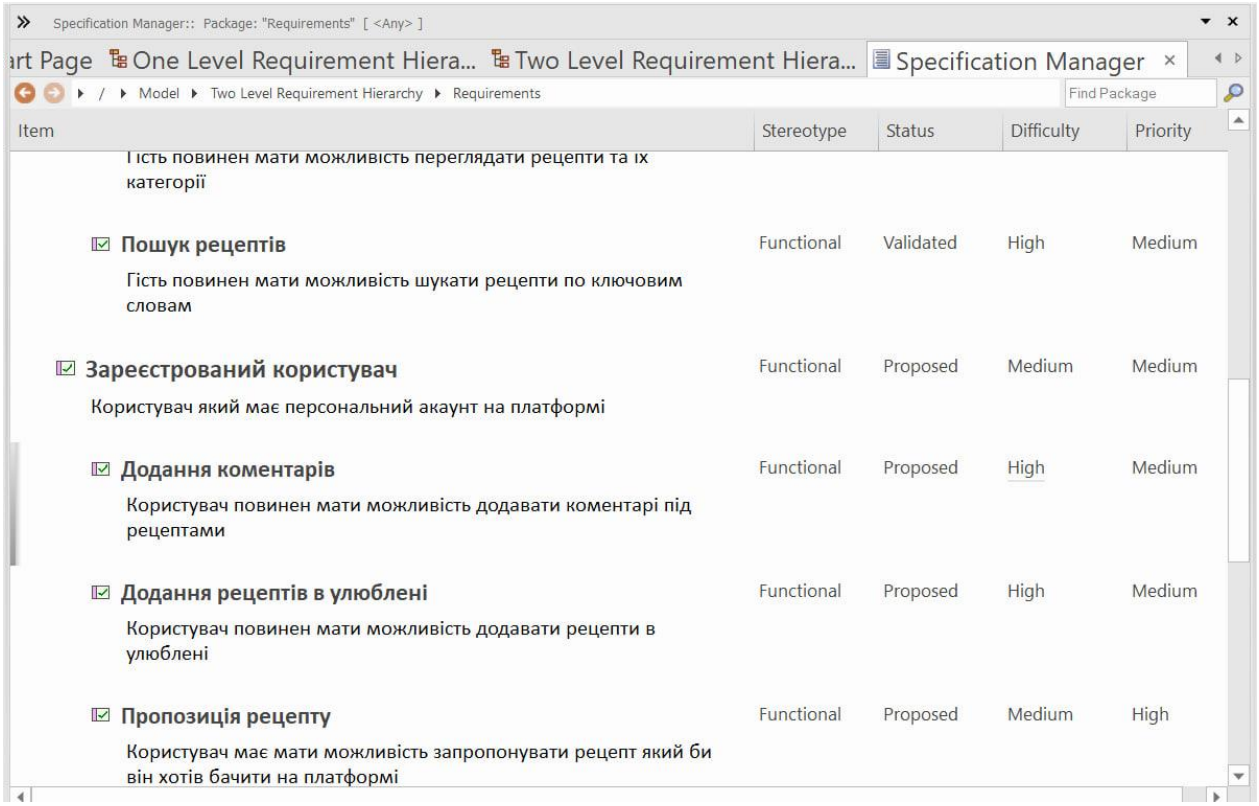


Рисунок 2.2 – Таблиця специфікації нефункціональних вимог

Нефункціональні вимоги описують якості та характеристики системи, які не є пов'язаними з конкретними функціями, але визначають її загальну ефективність, надійність та зручність використання. Ці вимоги враховують аспекти, такі як продуктивність, безпека, масштабованість, сумісність із зовнішніми системами та інші.

Нефункціональні вимоги часто є критичними для успіху продукту чи системи, оскільки вони визначають загальне враження користувача від його використання. Забезпечення високої якості нефункціональних характеристик дозволяє зберегти конкурентні переваги, підвищити задоволеність користувачів та забезпечити стабільну та надійну роботу системи у різних умовах експлуатації.

Ефективне врахування нефункціональних вимог під час розробки та впровадження дозволяє створити продукт чи систему, яка відповідає вимогам ринку та забезпечує успіх на довгострокову перспективу.

Нижче наведено нефункціональні вимоги до системи обміну кулінарними рецептами та порадами, які визначають якість, безпеку та ефективність системи, щоб забезпечити задоволення потреб користувачів та стабільну роботу платформи.

Нефункціональні вимоги:

- Безпека даних: система повинна забезпечувати високий рівень захисту особистої інформації користувачів та конфіденційності рецептів, відповідно до вимог законодавства про захист персональних даних.
- Продуктивність та масштабованість: система повинна бути здатна обробляти великий обсяг даних та витримувати високе навантаження в періоди пікової активності без втрати продуктивності.
- Інтерфейс користувача: інтерфейс користувача повинен бути інтуїтивно зрозумілим та дружнім до користувача, забезпечуючи зручність навігації та швидкий доступ до потрібної інформації.

- Сумісність з різними пристроями: платформа повинна бути оптимізована для різних типів пристроїв (комп'ютери, планшети, мобільні телефони) та різних браузерів, забезпечуючи однакову функціональність та відображення на всіх платформах.
- Швидкість завантаження: система повинна максимально швидко завантажувати сторінки та контент, забезпечуючи позитивний досвід користувача та зменшуючи час очікування.
- Доступність та відмовостійкість: система повинна бути доступною для користувачів у будь-який час без перерв у роботі, а також повинна бути стійкою до відмов та швидко відновлювати роботу після непередбачуваних ситуацій.

2.2. Постановка та алгоритм розв’язання задачі

2.2.1. Постановка задачі

Призначенням даної задачі є створення веб-платформи для обміну кулінарними рецептами і порадами. Техніко-економічна сутність полягає в тому, щоб створити цифровий продукт, який буде забезпечувати користувачам зручний доступ до різноманітних рецептів, можливість обговорення та обміну думками щодо них, а також створення власного контенту. Така платформа може відповісти на потреби користувачів у відкритому обміні кулінарними знаннями та досвідом, що сприятиме підвищенню інтересу до готування та розвитку кулінарної культури.

Використання електронних обчислювальних машин (ЕОМ) для розв'язання цієї задачі обгрунтоване через декілька причин. По-перше, ЕОМ дозволять забезпечити швидку та ефективну роботу платформи, що важливо для забезпечення задоволення потреб користувачів у максимально короткі терміни. По-друге, використання ЕОМ дозволить автоматизувати багато процесів, пов'язаних з управлінням та обробкою інформації на платформі, що спростить роботу адміністраторів та покращить користувацький досвід. По-

третє, використання ЕОМ дозволить забезпечити високу рівень безпеки та захисту персональних даних користувачів, що має велике значення в сфері обміну конфіденційною інформацією, як кулінарні рецепти.

Таким чином, використання ЕОМ для розв'язання цієї задачі є доцільним і ефективним з точки зору забезпечення якісного та зручного функціонування веб-платформи для обміну кулінарними рецептами і порадами.

Технічно-економічна сутність полягає в автоматизації процесів зберігання, пошуку та обміну рецептами, що робить користування платформою простим та захоплюючим. Об'єктами управління є різні компоненти, які надають можливість користувачам отримувати інформацію про рецепти, а саме:

– Категорія рецепту: Легка навігація за типами страв або іншими категоріями для швидкого знаходження потрібного.

– Назва рецепту: Інформація про конкретну страву для швидкого розпізнавання та вибору.

– Інгредієнти: Тут подаються детальні переліки продуктів, необхідних для приготування кожного рецепту. Це допомагає забезпечити зручність та ясність у процесі підготовки, а також гарантує, що ви точно знаєте, що вам потрібно для створення смачного шедевра.

– Інструкції та поради: Цей розділ містить крок за кроком інструкції щодо процесу готування, а також надає корисні поради від інших талановитих кулінарів. Ви не лише вивчаєте нові рецепти, а й збагачуєте свій досвід та техніку завдяки спільноті експертів.

Цей функціонал також надає можливість виставляти оцінки та залишати відгуки, допомагаючи іншим користувачам отримати додатковий контекст та рекомендації. Враження та спільні думки стають необхідною частиною кулінарного діалогу та обміну досвідом. Такий підхід сприятиме створенню динамічної та взаємодіючої спільноти кулінарів, яка об'єднує їхні ідеї та творчість в гастрономічному вимірі.

Умови, за яких припиняється автоматизоване розв’язання задачі:

– Відсутність Інтернет-з'єднання: У випадку відсутності доступу до мережі Інтернет, автоматизоване розв’язання може припинитися, оскільки користувач не матиме змоги отримувати оновлення, а також ділитися своїми рецептами чи спілкуватися з іншими учасниками.

– Технічні Порушення: У випадку технічних проблем, таких як відмова сервера чи системних збоїв, автоматизоване розв’язання може призупинитися до усунення неполадок та відновлення нормального функціонування системи.

– Невідповідність Правилам та Умовам: Якщо користувач порушує правила чи умови використання платформи, його можуть виключити з автоматизованого обслуговування до вирішення адміністративних питань.

Адміністратори платформи мають на собі відповідальні обов'язки, пов'язані з забезпеченням безперебійної та ефективної роботи системи. Вони відповідають за моніторинг, обслуговування, вирішення технічних проблем, а також за захист та безпеку даних користувачів. Адміністратори взаємодіють з користувачами, вирішуючи їхні запитання та надаючи підтримку. Крім того, вони відповідають за публікацію запропонованих від користувачів рецептів, або ж їх видалення.

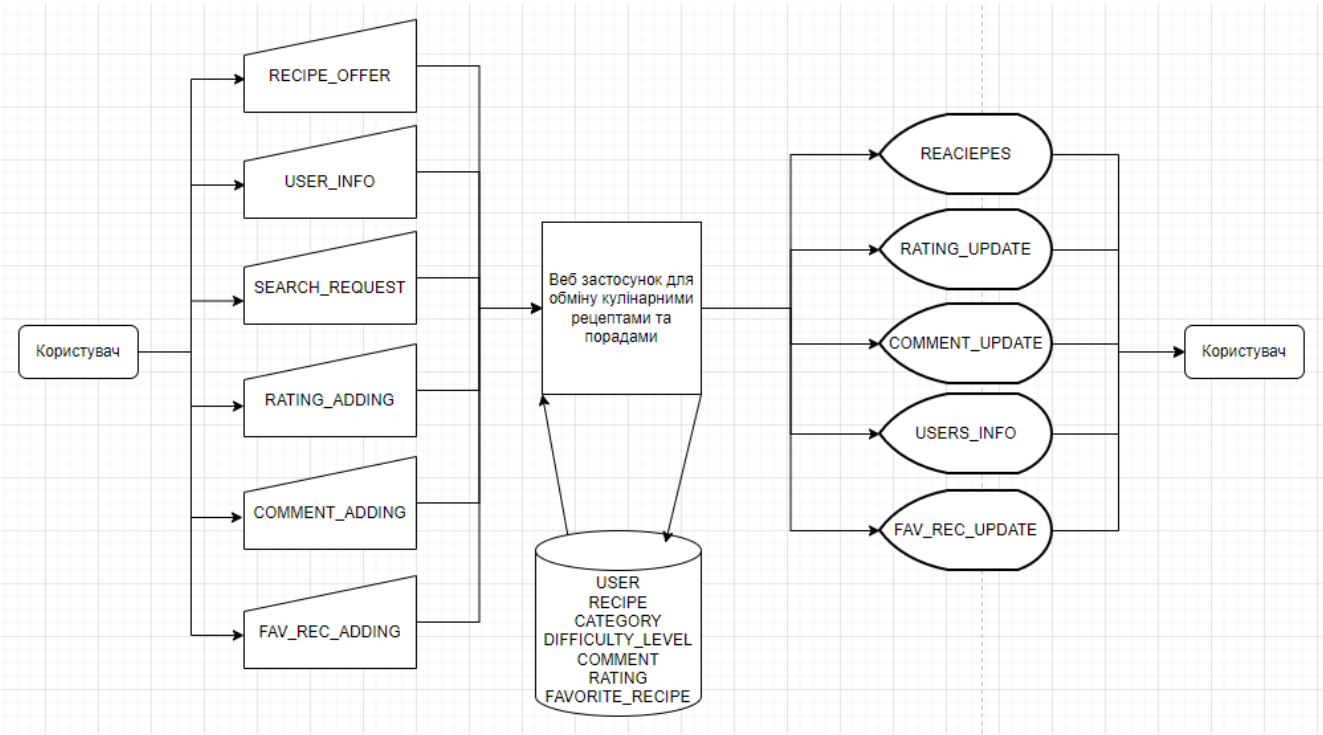


Рисунок 2.3 – Інформаційна модель

Вхідна інформація для цього проекту включає дані, які використовуються при пропозиції рецепту, створенні акаунту, додаванні оцінки та інших аспектах взаємодії з платформою. Для докладності щодо вхідної інформації розглянемо дані, представлені в Таблиці 2.1.

Таблиця 2.1 – Перелік і опис вхідних повідомлень

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1.	Пропозиція рецепту від користувача	RECIPE_OFFER USER RECIPE	Електронна форма, масив	В будь-який момент	Користувач Рецепт БД
2.	Облікові записи користувачів	USER_INFO USER	Електронна форм, масив	В будь-який момент	Користувач БД
3.	Запит на пошук рецепту	SEARCH_REQUEST USER RECIPE	Пошуковий рядок	В будь-який момент	Користувач Рецепт БД
4.	Додання оцінки	RATING_ADDING USER RATING	Електронна форма, масив	В будь-який момент	Користувач Рейтинг БД

Продовження таблиці 2.1

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
5.	Додання коментарю	COMMENT_ADDING USER COMMENT	Електронна форма, масив	В будь-який момент	Користувач Коментар БД
6.	Додання рецепту в улюблені	FAV_REC_ADDING USER RECIPE	Електронна форма, масив	В будь-який момент	Користувач Рецепт БД

Вихідна інформація в рамках цього проекту використовується для передачі деталей про користувачів, рецепти та поради, запропоновані користувачами, а також для передачі рейтингу рецептів та відгуків, які залишили інші користувачі. Для детальнішого розгляду вихідних даних розглянемо інформацію, представлену в Таблиці 2.2.

Таблиця 2.2 – Перелік і опис вихідних повідомлень

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1.	Рецепти та поради користувачів	REACIPES COMMENT RATING	Сторінка веб-сайту, масив	Після оновлення веб-сайту	Кулінарний веб-сайт
2.	Оновлення рейтингу	RATING_UPDATE RATING	Сторінка веб-сайту, масив	Після оновлення веб-сайту	Кулінарний веб-сайт
3.	Оновлення відгуків	COMMENT_UPDATE COMMENT	Сторінка веб-сайту, масив	Після оновлення веб-сайту	Користувач, який залишив відгук
4.	Дані про користувачів	USERS_INFO USER	Дані облікових записів, які включають інформацію про користувачів та їхню активність на платформі	При реєстрації, при зміні налаштувань або активності на платформі	Облікові записи користувачів та їхні дії на платформі
5.	Оновлення списку улюблених рецептів	FAV_REC_UPDATE FAVORITE_RECIPE	Сторінка веб-сайту, масив	Після оновлення веб-сайту	Кулінарний веб-сайт

2.2.2. Алгоритм розв’язання задачі

Для точного та детального опису алгоритмів вирішення задачі необхідно визначити кілька ключових елементів: використовувану інформацію, очікувані результати, математичний опис, а також сам алгоритм, що реалізується на електронно-обчислювальній машині (ЕОМ). Використовувана інформація, яка є невід'ємною частиною алгоритму та необхідна для створення, редагування і пошуку об'єктів у системі, представлена у таблиці 2.3.

Таблиця 2.3 – Перелік масивів використовуваної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Рецепти	RECIPE	10000
Користувачі	USER	10000
Коментарі	COMMENT	100000
Рейтинг	RATING	100000
Улюблений рецепт	FAVORITE_RECIPE	100000
Рівень складності	DIFFICULTY_LEVEL	30
Категорія	CATEGORY	50

Результуюча інформація, яка формується в результаті виконання алгоритму та є важливою для подальшого аналізу і прийняття рішень, представлена у таблиці 2.4.

Таблиця 2.4 Перелік масивів результатної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Інформація про рецепт	RecipeInfo	10000
Список рецептів	RecipeList	10000
Список коментарів	CommentList	100000
Рейтинг рецептів	RatingList	100000
Список категорій	CategoryList	50
Список рівнів складності	DifficultyList	30

У проекті планується використання математичних моделей та розробка алгоритмів, які оптимально забезпечать реалізацію основних функцій системи. Це включає декілька важливих аспектів, які спрямовані на забезпечення ефективності, швидкодії та зручності використання платформи.

Формула розрахунку загальної кількості рецептів:

$$RecCount = \sum_{id} rec_{id} \quad (2.1)$$

де RecCount – загальна кількість рецептів,

id – ідентифікатор користувача,

rec_{id} – кількість рецептів від певного користувача.

Формула розрахунку рейтингу рецептів:

$$Rat_d = \frac{\sum_u (u\_rat_{d)}}{Tot\_U} \quad (2.2)$$

де Rat_d – рейтинг конкретної страви,

u – користувач,

u_rat_d оцінка користувача для даної страви,

Tot_U – загальна кількість користувачів.

Формула для розрахунку активності користувача:

$$Act_u = \frac{Tot\_Rec\_Post_u + Tot\_Com\_M_u}{Mem\_Dur_u} \quad (2.3)$$

де Act_u – активність конкретного користувача,

$Tot\_Rec\_Post_u$ – загальна кількість рецептів, які додав користувач,

$Tot\_Com\_M_u$ загальна кількість коментарів, які залишив користувач,

Mem_Dur_u – тривалість членства користувача на сайті.

Процес авторизації:

1. Гість вводить дані у форму авторизації;

1.2. Якщо введені дані є у базі даних то користувач проходить авторизацію;

1.2. Якщо введених даних немає в базі даних то:

1.2.1. Користувач переходить на сторінку реєстрації;

1.2.2. Користувач вносить дані у форму;

1.2.3. Дані вносяться до БД, зберігаються та переадресовують користувача до сторінки входу, де він проходить етапи, написані у пункті 1 процесу авторизації (див. пункт. 1).

Процес перегляду рецепту:

1. Користувач знаходить необхідний для нього рецепт;
2. Робиться запит до БД на видачу інформації про рецепт;
3. Передана інформація, форматується і передається користувачу;

4. Користувач переглядає рецепт, може додати його до улюблених, поставити оцінку, додати коментар або не зробити нічого.

Процес перегляду власних рецептів:

1. Користувач натискає на своє ім’я біля кнопки “Вийти”.

2. Відкривається особистий кабінет;

3. Користувач натискає на “Мої рецепти”;

4. Передана інформація, форматується і передається користувачу;

4.1. Якщо користувач створював рецепти на сайті, то відображаються всі опубліковані рецепти користувача;

4.2. Якщо користувач не створював рецепти на сайті відображається текст “На жаль рецепти не знайдені”.

Процес перегляду улюблених рецептів:

1. Користувач натискає на своє ім’я біля кнопки “Вийти”;

2. Відкривається особистий кабінет;

3. Користувач натискає на “Улюблені рецепти”;

4. Передана інформація, форматується і передається користувачу;

4.1. Якщо користувач додавав рецепти в улюблені, то відображаються всі улюблені рецепти користувача;

4.2. Якщо користувач не додавав рецепти в улюблені відображається текст “На жаль рецепти не знайдені”.

Процес пропозиції нового рецепта:

1. Користувач відкриває вкладку “Запропонувати рецепт”;

2. Відкривається форма для пропозиції рецепту;

3.1 Користувач вносить дані до форми;

3.2 Дані з форми передаються на БД та зберігаються;

3.2.1 Адміністратор перевіряє та публікує рецепт;

3.2.2 Адміністратор перевіряє та видаляє рецепт;

4. Користувач відміняє створення.

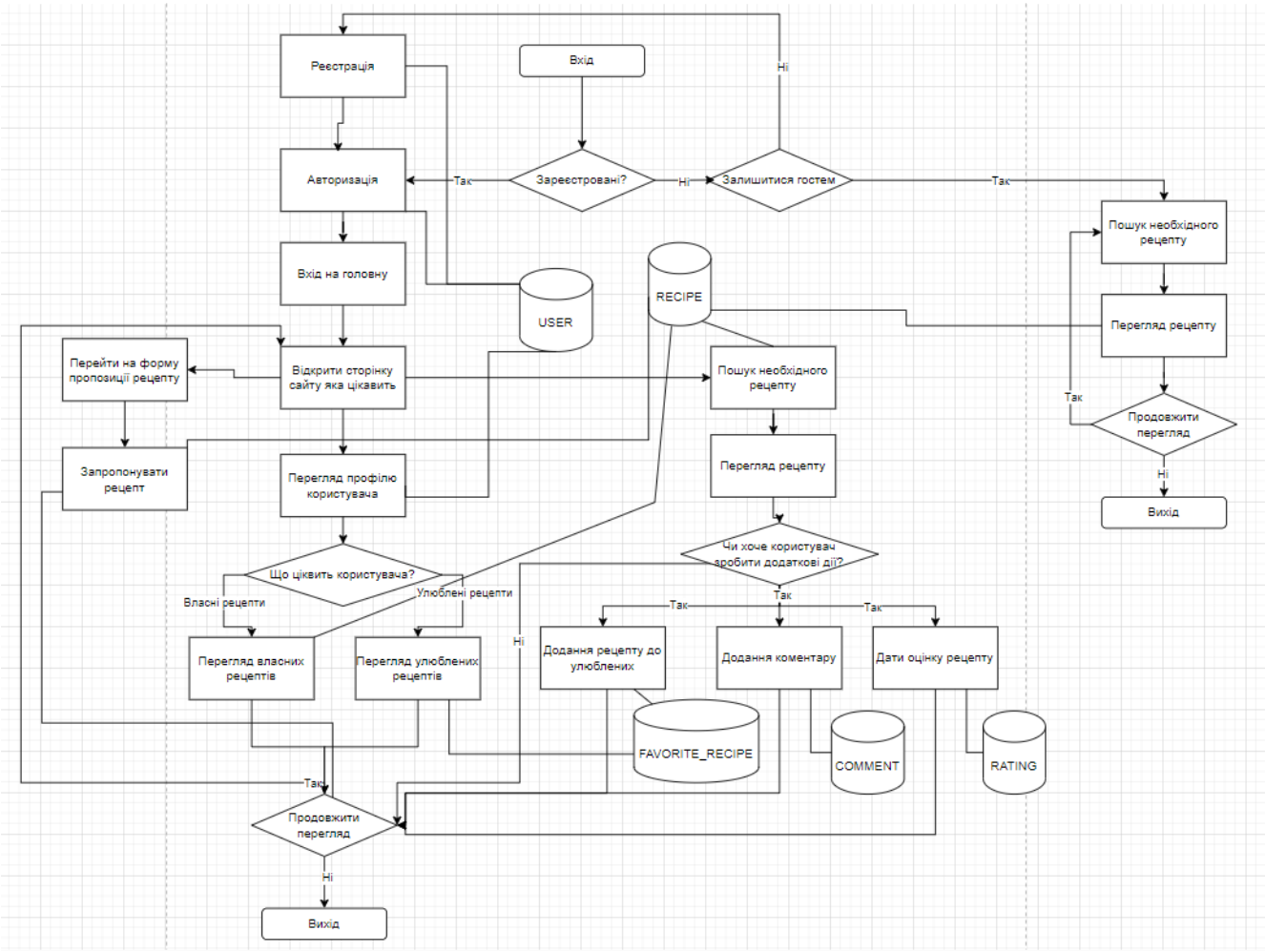


Рисунок 2.4 – Алгоритм функціонування системи

Цей алгоритм візуалізує всі етапи та послідовність операцій у системі обробки інформації. Він розглядає різноманітні кроки, доступні користувачеві, що сприяє глибшому розумінню концепції інформаційної системи та полегшує подальші кроки у розробці веб-застосунку.

2.3. Моделювання інформаційних систем

2.3.1 Моделювання поведінки системи

Діаграма прецедентів - це важливий інструмент у методології Unified Modeling Language (UML), який використовується для моделювання функціональності системи. Вона дозволяє визначити всі можливі дії, які можуть бути виконані користувачами (акторами) у системі.

Головною метою діаграми прецедентів є ідентифікація функціональних вимог до системи, тобто тих операцій, які система повинна виконувати для задоволення потреб користувачів. Кожен прецедент представляє собою конкретну функціональну можливість, яка може бути виконана системою.

Діаграма прецедентів є важливим інструментом для розробки програмного забезпечення, оскільки вона дозволяє команді розробників та стейкхолдерам зрозуміти, як система буде використовуватися та які функції вона повинна виконувати. Це допомагає забезпечити, що розроблена система відповідає потребам користувачів і відповідає бізнес-вимогам.

Діаграма прецедентів веб-застосунку для обміну кулінарними рецептами та порадами описує різні дії, які можуть виконувати гості, авторизовані користувачі та адміністратори.

Гості системи мають можливості виконувати такі дії:

- Пошук рецепту за назвою: цей прецедент дозволяє гостям здійснювати пошук конкретного рецепту за його назвою. Гості можуть швидко знаходити потрібні страви, використовуючи назву як ключове слово для пошуку;
- Сортування рецепту за категорією: цей прецедент дозволяє гостям сортувати рецепти за їхньою категорією. Гості можуть отримати доступ до рецептів, які відповідають їхнім конкретним вимогам або уподобанням;
- Перегляд рецептів: цей прецедент дозволяє гостям переглядати всі наявні рецепти на платформі. Гості мають змогу отримати детальну інформацію про кожен рецепт, таку як список інгредієнтів, інструкції щодо приготування та фотографії страви;
- Реєстрація: Цей прецедент дозволяє новим користувачам зареєструватися на веб-сайті, надаючи необхідну інформацію та обираючи унікальне нік користувача та пароль.

Авторизовані користувачі мають розширені функціональні можливості та можуть додатково:

• Додати рецепт в улюблені: цей прецедент дозволяє користувачам додавати певні рецепти до свого списку улюблених. Користувачі можуть зручно зберігати та отримувати доступ до цих рецептів у майбутньому, щоб повторно готувати свої улюблені страви;

• Оцінити рецепт: цей прецедент дозволяє користувачам оцінювати рецепти, виражаючи свої враження від них за певною шкалою, наприклад, від одного до п'яти зірок. Оцінки дозволяють іншим користувачам швидко зрозуміти, які рецепти отримали позитивні відгуки від спільноти;

• Додати коментар: цей прецедент дозволяє користувачам залишати свої коментарі та відгуки під рецептами. Користувачі можуть висловлювати свої думки, ділитися своїм досвідом або надавати корисні поради щодо приготування конкретної страви.

• Запропонувати рецепт: цей прецедент дозволяє користувачам додавати нові рецепти до системи. Користувачі можуть надати детальну інформацію про страву, включаючи назву, список інгредієнтів, інструкції з приготування та будь-які інші відомості, які вважають потрібними. Після цього рецепт може бути переглянутий модераторами для оцінки відповідності правилам та якості контенту перед його публікацією.

Адміністратори мають розширені функціональні можливості та можуть додатково:

• Опублікувати рецепт: цей прецедент дозволяє адміністраторам публікувати нові рецепти у системі. Після надання всієї необхідної інформації про страву, адміністратор може поставити галочку "Опубліковано", після чого рецепт стане доступним для перегляду всіма користувачами платформи.

• Видалити рецепт: цей прецедент дозволяє адміністраторам видаляти рецепти з системи. Причини для видалення можуть включати неправильну інформацію, порушення правил або власне рішення автора. Після видалення рецепту він більше не буде доступний для перегляду користувачам.

• Редагувати рецепт: цей прецедент дозволяє адміністраторам редагувати існуючі рецепти. Після внесення змін до рецепту, таких як корекція інформації або оновлення списку інгредієнтів, вони можуть зберегти зміни, щоб вони відображалися для всіх користувачів.

• Видалити коментарі: цей прецедент дозволяє адміністраторам видаляти коментарі, залишені під рецептами. Причини для видалення можуть включати порушення правил спільноти, образливий чи нецензурний вміст або будь-яку іншу неприпустиму поведінку.

• Видалити користувачів: цей прецедент дозволяє адміністраторам видаляти користувачів з системи. Причини для видалення можуть включати порушення правил, спам, небажану активність або будь-які інші обставини, які можуть загрожувати безпеці або функціонуванню платформи.

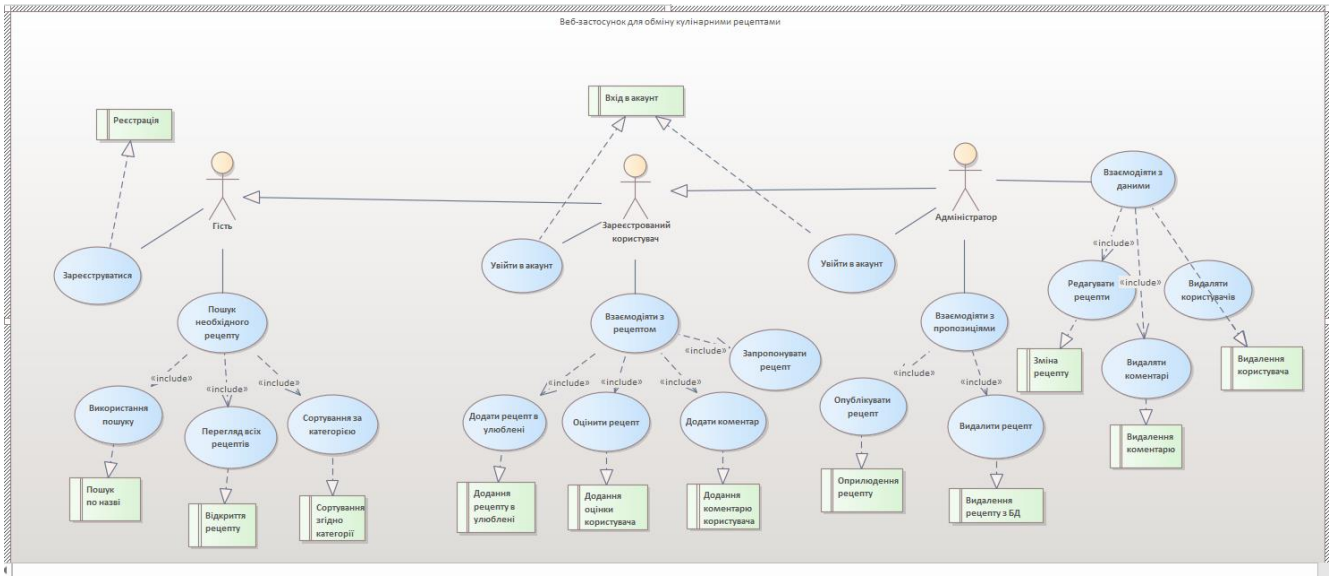


Рисунок 2.5 – Діаграма прецедентів

1. <u>Користувач реєструється на сайті:</u>

Нормальний сценарій:

1) Користувач натискає на кнопку “Реєстрація” яка знаходиться у меню сайту;

2) Користувач вводить персональні дані у форму реєстрації (нік, електронну пошту та пароль);
3) Після введення коректних даних користувач натискає на кнопку “Зареєструватися”;
4) Потім система додає користувача у базу даних та повертає аунтентифікованого користувача на головну сторінку.

Альтернативний сценарій:

• Користувач вводить некоректні дані у форму реєстрації, такі як невірний формат електронної пошти або надлишково короткий пароль;

• Після спроби відправити форму з некоректними даними, система виявляє помилки у введених даних і відображає користувачеві повідомлення про необхідність виправлення помилок перед продовженням реєстрації.

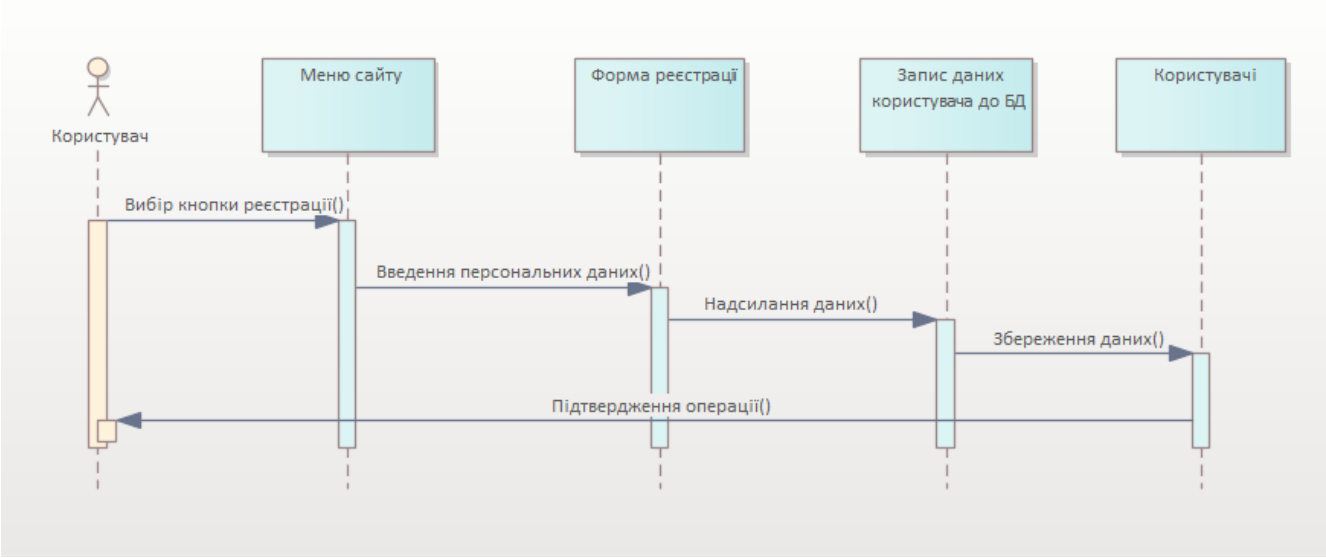


Рисунок 2.6 – Діаграма послідовності прецеденту “Реєстрація”

2. Користувач пропонує рецепт:

Нормальний сценарій:

1) Користувач обирає веб сторінку “Запропонувати рецепт” з меню сайту;

2) Користувач вводить всі необхідні дані у форму рецепту (інгредієнти, кроки до приготування, фотографія страви, кількість порцій, час приготування і тд.);
3) Після введення всіх даних користувач натискає на кнопку “Запропонувати рецепт”;
4) Потім система додає рецепт у базу даних, але не публікує на сайті, бо чекає перевірку від адміністратора.

Альтернативний сценарій:

Користувач намагається надіслати пропозицію рецепту, але у процесі відправлення виникає помилка через технічні проблеми на сервері. В такому випадку, система відображає повідомлення про тимчасову недоступність сервісу та рекомендує спробувати пізніше.

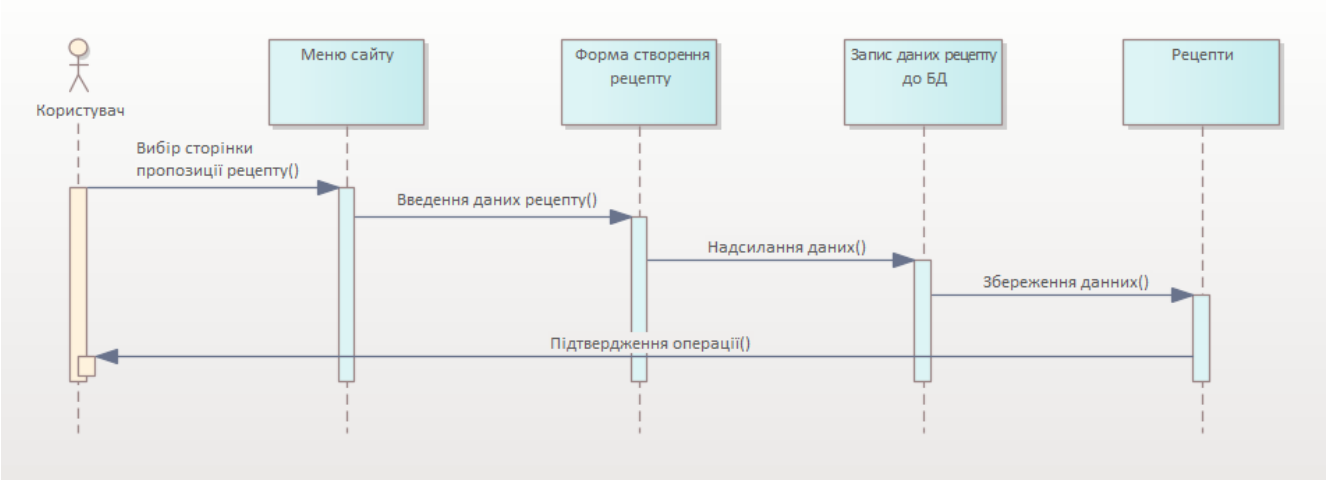


Рисунок 2.7 – Діаграма послідовності “Запропонувати рецепт”

3. Адміністратор публікує рецепт:

Нормальний сценарій:

1) Адміністратор відкриває панель адміністратора;
2) Потім він вибирає вкладку “Рецепти”;
3) Після цього адміністратор відсортовує всі неопубліковані рецепти;
4) Потім він відкриває будь-який не опублікований рецепт;

5) Після того як рецепт був відкритий, адміністратор може перевірити його на відповідність правилам і стандартам платформи. Якщо перевірки пройшла, адміністратор ставить галочку “Опублікувати” та натискають кнопку “Зберегти”;
6) Після цього система оновлює рецепт, та публікує його на сайті, після чого рецепт стає доступним для перегляду всіма користувачами платформи.

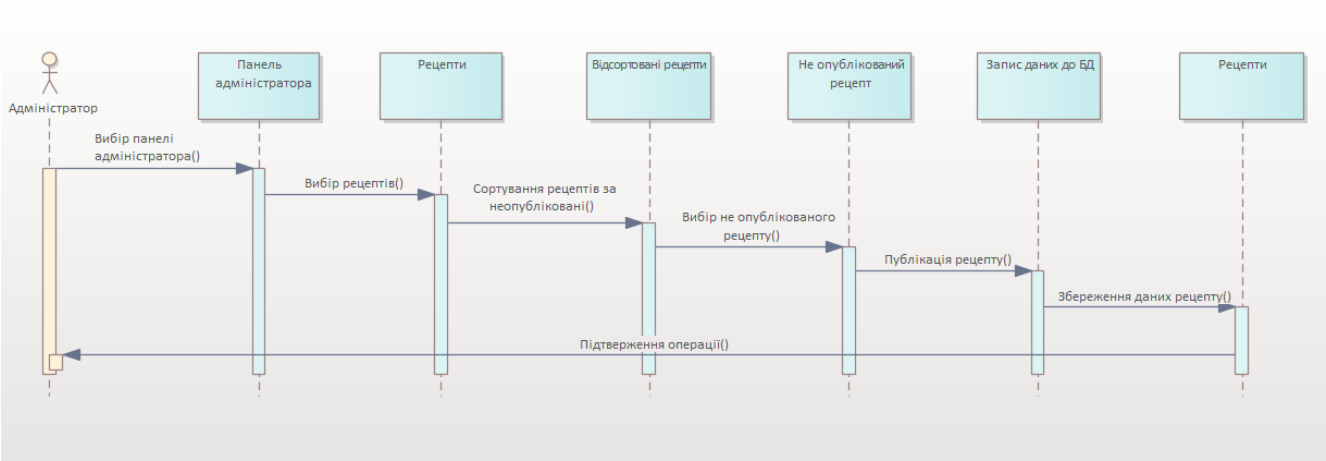


Рисунок 2.8 – Діаграма послідовності “Опублікувати рецепт”

4. Користувач додає коментар:

Нормальний сценарій:

1) Користувач обирає цікавий йому рецепт;
2) Після відкриття рецепту, користувач вводить текст в поле для коментарів;
3) Після того як користувач натиснув на кнопку “Додати коментар”, текст коментарю відправляється на сервер;
4) Потім система оброблює коментар і публікує його під рецептом.

Альтернативний сценарій:

Замість натискання на кнопку "Додати коментар", користувач втрачає з'єднання з Інтернетом під час введення тексту коментаря.

• Система не може відправити коментар на сервер через відсутність зв'язку з Інтернетом.

• Користувач отримує повідомлення про помилку із порадою перевірити підключення до Інтернету та спробувати знову пізніше.

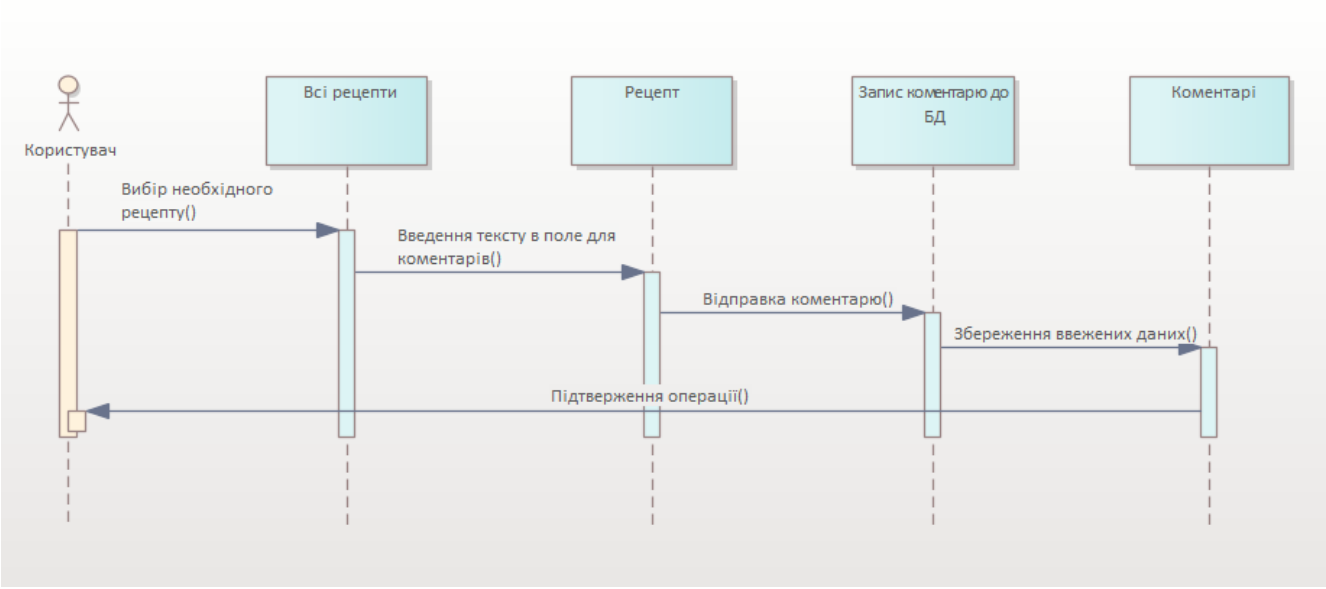


Рисунок 2.9 – Діаграма послідовності “ Додати коментар”

Таким чином, для певних прецедентів визначено "нормальний" сценарій використання, який описує послідовність дій у стандартних умовах. Крім того, кожен описаний прецедент має альтернативний сценарій, який відображає можливі відхилення від "нормального" варіанту в залежності від певних обставин чи помилок. Такий підхід дозволяє системі ефективно впоратися з різними ситуаціями та забезпечити користувачам надійну та стабільну роботу.

2.3.2. Модель структури системи

Модель структури системи є ключовим аспектом в аналізі та проектуванні програмного забезпечення. Ця модель визначає, як компоненти системи взаємодіють між собою та як вони організовані для досягнення бажаної функціональності. Вона включає в себе діаграми визначення блоків, діаграми внутрішніх блоків та діаграми класів.

Діаграма визначення блоків

Головний блок "Веб-застосунок для обміну кулінарними рецептами" складається з різноманітних компонентів, які спільно працюють для забезпечення його функціональності та ефективності. Серед цих компонентів можна виділити базу даних, адміністрація, веб-інтерфейс, та серверну частину системи.

База даних у веб-застосунку для обміну кулінарними рецептами є ключовим компонентом, який забезпечує зберігання та організацію всієї інформації, пов'язаної з рецептами, користувачами та іншими аспектами системи. Ця база даних дозволяє ефективно керувати великим обсягом інформації та забезпечує швидкий доступ до потрібних даних.

Основна функція бази даних - забезпечення структурованого та безпечного зберігання даних, гарантуючи їх цілісність та доступність для необхідних операцій. Вона також відповідає за забезпечення безпеки даних та контроль доступу до них, щоб забезпечити конфіденційність інформації користувачів.

Адміністрація веб-застосунку для обміну кулінарними рецептами відповідає за ефективне функціонування та надійність системи, а також за керуванням контентом, включаючи публікацію, модерацію та управління рецептами. Ця команда фахівців забезпечує належне розгортання технічних ресурсів, таких як сервери та бази даних, для забезпечення стабільної роботи веб-застосунку.

Основні завдання адміністрації включають вирішення технічних проблем, які виникають у користувачів, та надання їм відповідної технічної підтримки. Вони також відповідають за забезпечення безпеки даних та конфіденційності інформації, що зберігається в системі.

Веб-інтерфейс веб-застосунку для обміну кулінарними рецептами є ключовим елементом взаємодії користувачів з системою. Це інтерактивна платформа, яка надає зручний та доступний спосіб взаємодії з рецептами, порадами та іншими функціями застосунку. Основні характеристики веб-

інтерфейсу включають навігаційне меню, яке надає користувачам зручний спосіб переходу між різними розділами та функціями застосунку, та пошукову систему, що дозволяє користувачам швидко знаходити потрібні рецепти за ключовими словами.

Серверна частина веб-застосунку для обміну кулінарними рецептами відіграє ключову роль у забезпеченні функціональності та надійності системи. Вона є центральним елементом, що обробляє запити від користувачів та забезпечує обмін даними між клієнтами та базою даних. Основні завдання серверної частини включають обробку запитів, зберігання та керування даними, а також забезпечення безпеки та моніторингу стану системи.

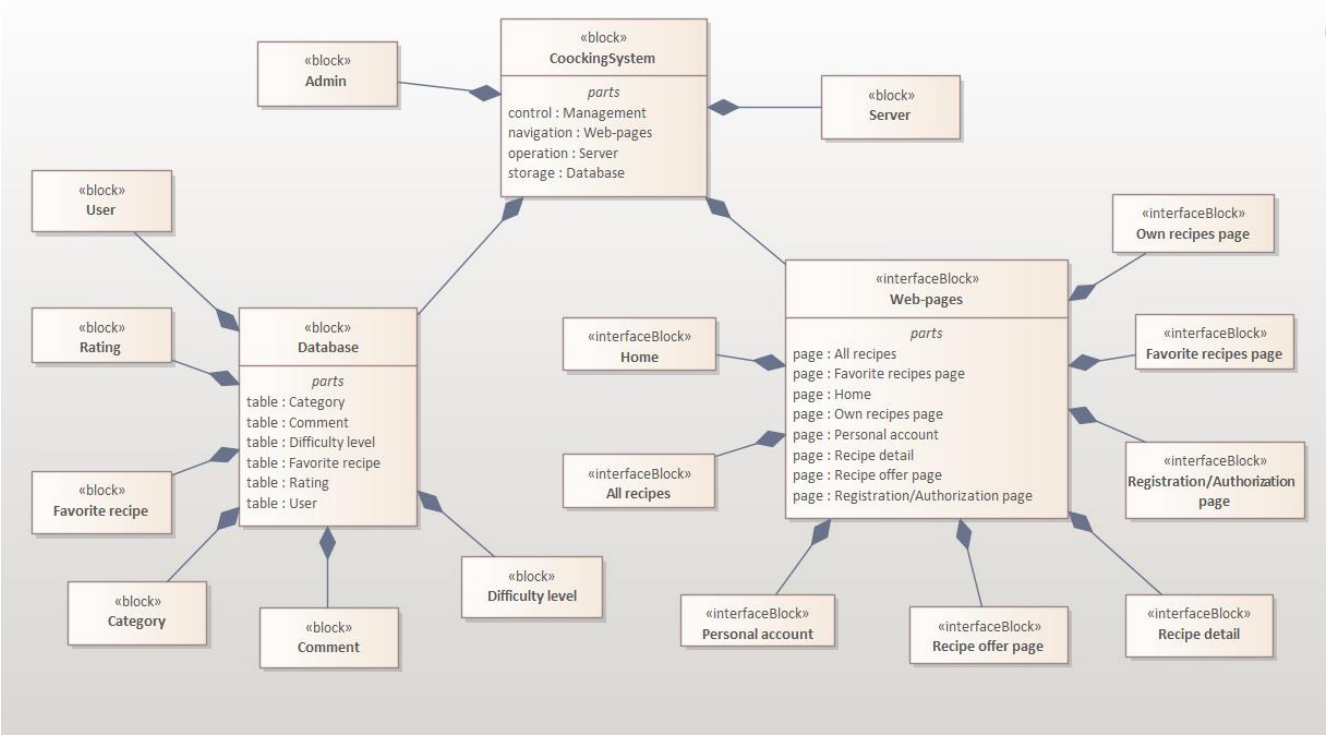


Рисунок 2.10 – Діаграма визначення блоків

Діаграми внутрішніх блоків

Після відкриття сторінки реєстрації, користувач заповнює форму, надаючи інформацію про себе. Після завершення заповнення форми, він надсилає запит на сервер для оновлення таблиці бази даних щодо нового користувача. Сервер отримує цей запит і передає його до бази даних, де дані

про користувача додаються. Після успішного оновлення даних, сервер викликає тригер, який повідомляє користувача про успішну операцію та перенаправляє його на головну сторінку.

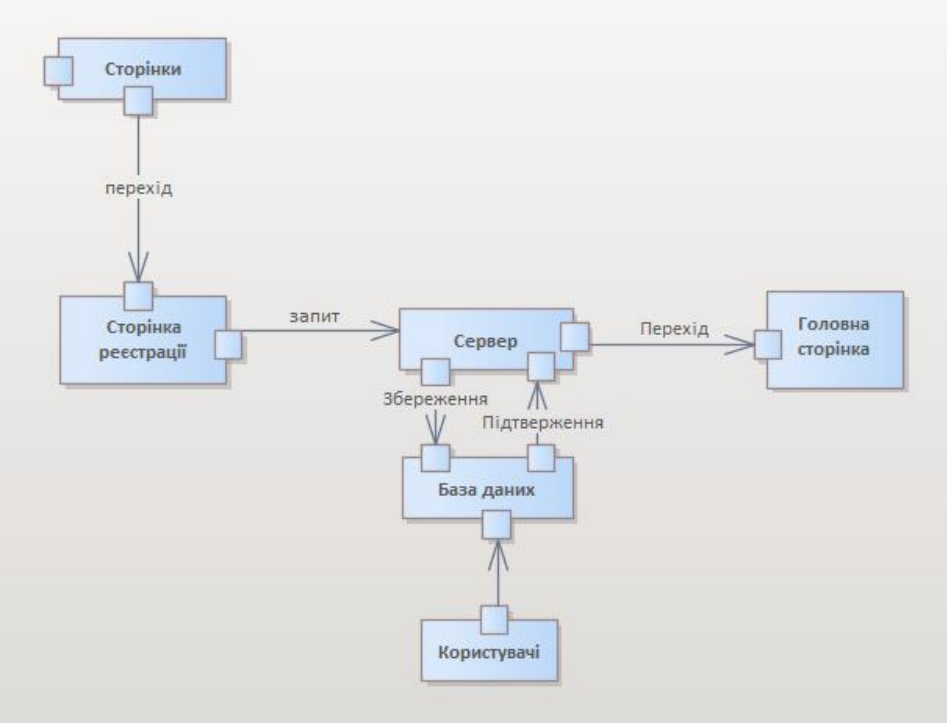


Рисунок 2.11 – Діаграма внутрішніх блоків обробки процесу заповнення форми “Реєстрація користувача”

Після відкриття сторінки “Пропозиція рецепту”, користувач заповнює форму, надаючи повну інформацію про рецепт. Після завершення заповнення форми, він надсилає запит на сервер для оновлення таблиці бази даних щодо нового рецепту. Сервер отримує цей запит і передає його до бази даних, де дані про рецепт додаються. Після успішного оновлення даних, сервер викликає тригер, який повідомляє користувача про успішну операцію та відображає повідомлення про успішну пропозицію рецепту.

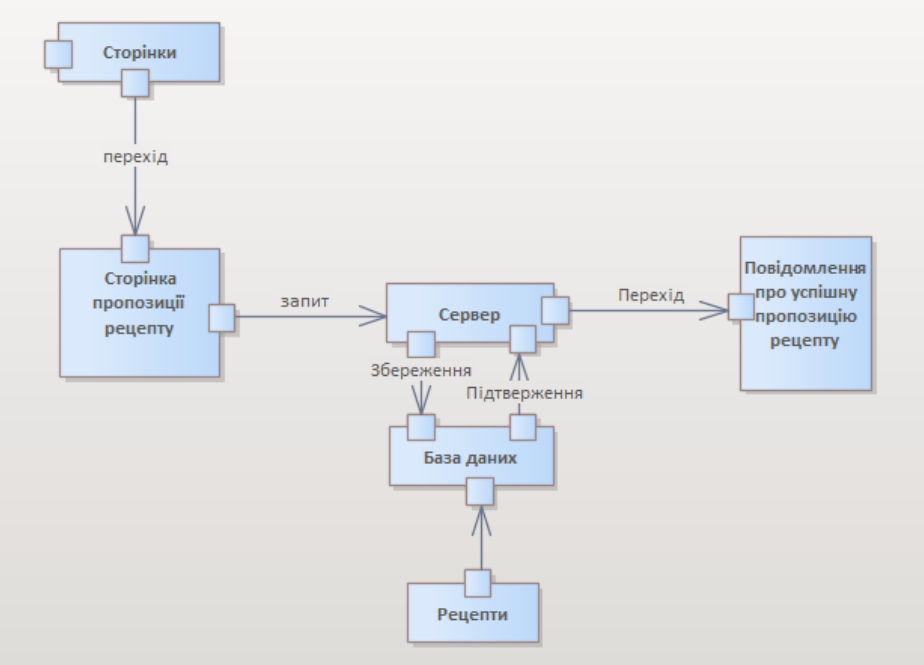


Рисунок 2.12 – Діаграма внутрішніх блоків обробки процесу заповнення форми “Пропозиції рецепту”

Діаграма класів

Діаграма класів - це структурна діаграма, яка відображає структуру системи за допомогою класів, їх взаємозв'язків та атрибутів. Кожен клас представляє об'єкт або сутність в системі, а взаємозв'язки між класами вказують на способи їх взаємодії.

Клас “Recipe” виступає у ролі моделі, та містить у собі всі необхідні атрибути які використовуються для рецептів.

Клас “Rating” виступає у ролі моделі, та містить у собі всі необхідні атрибути які використовуються для оцінки рецептів.

Клас “DifficultyLevel” виступає у ролі моделі, та містить у собі всі необхідні атрибути які необхідні для позначення складності рецептів.

Клас “Favorite” виступає у ролі моделі, та містить у собі всі необхідні атрибути для позначення улюблених рецептів.

Клас “Category” виступає у ролі моделі, та містить у собі всі необхідні атрибути для категорій.

Класи “CategoryRecipesViev”, “RecipeSearchViev” та “AllRecipe” мають схожі атрибути і методи та потрібні для виведення всіх необхідних рецептів згідно з заданої ситуації.

Клас “RecipeDetailViev” потрібний для відображення детальної інформації про рецепт.

Клас “ToggleFavoriteViev” потрібний для додавання рецепту у список улюблених.

Клас “RegisterViev” потрібний для відображення форми реєстрації.

Клас “LoginForm” потрібний для відображення форми аутентифікації.

Клас “HomeRecipe” потрібний для відображення головної сторінки.

Клас “CreateRecipeView” потрібний для того щоб користувачі мали можливість пропонувати нові рецепти на платформі.

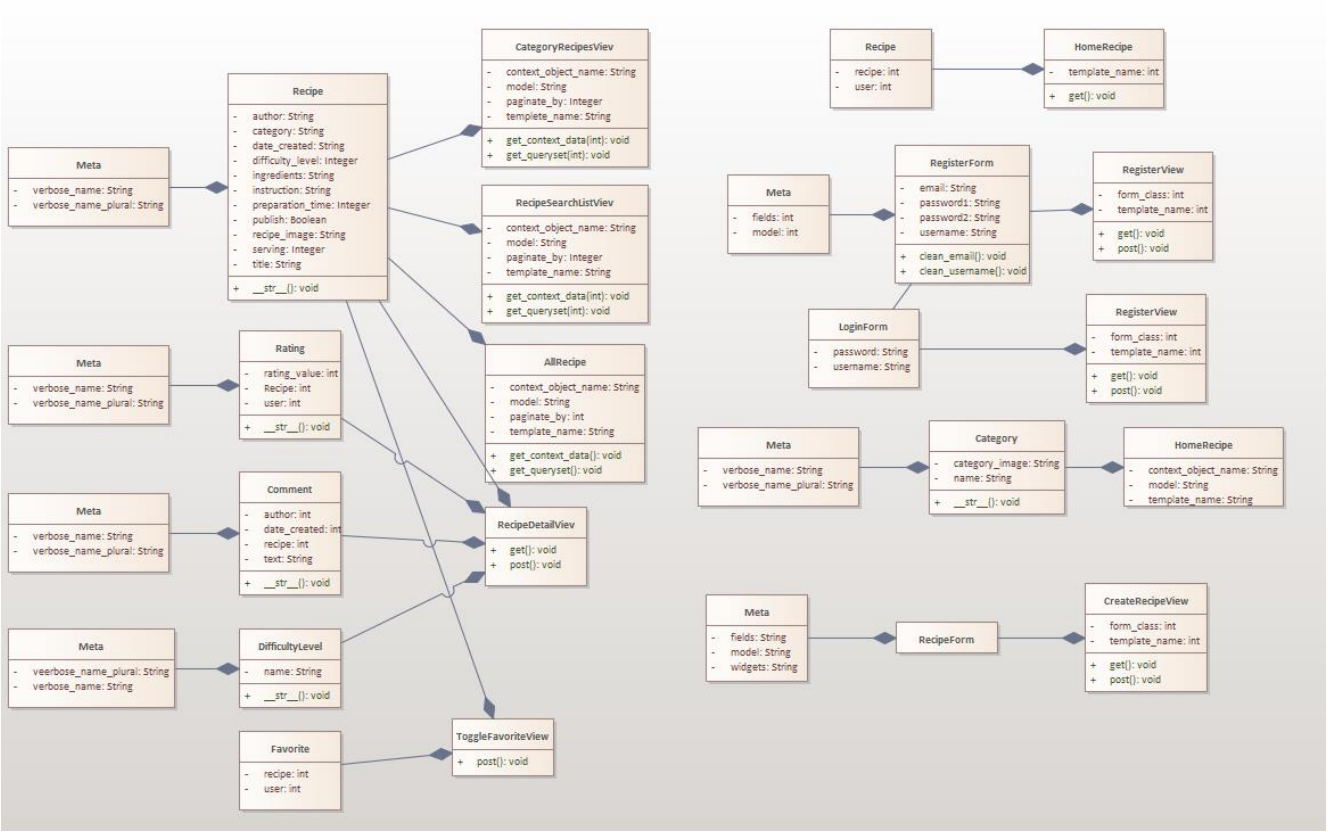


Рисунок 2.13 – Діаграма класів

2.3.3. Розподіл вимог за компонентами системи

Діаграма трасувань є важливим інструментом в інженерії вимог, що допомагає відстежити зв'язки між різними елементами системи та її вимогами. Вона надає можливість відображення, які вимоги стосуються конкретних компонентів, функцій чи інших елементів системи, і навпаки, які елементи системи відповідають конкретним вимогам.

Ця діаграма допомагає забезпечити повноту і консистентність вимог до системи, а також виявити потенційні прогалини або конфлікти між різними вимогами. Наприклад, вона дозволяє визначити, які компоненти системи повинні бути реалізовані для задоволення певних вимог, і перевірити, чи всі вимоги враховані при проектуванні та розробці системи.

Діаграма трасувань також допомагає у визначенні обсягу тестування та розробці тест-кейсів. Вона дозволяє встановити, які вимоги потрібно перевірити за допомогою конкретних тестів і які функціональності повинні бути покриті тестами.

Узагальнюючи, діаграма трасувань допомагає забезпечити високу якість програмного забезпечення шляхом ефективного управління вимогами, аналізу зв'язків між різними елементами системи та вимогами, а також планування тестування для впевненості у відповідності системи вимогам.

Загальний вигляд діаграми трасування для веб-застосунку для обміну кулінарними рецептами та порадами можна побачити на рисунку 4.4.

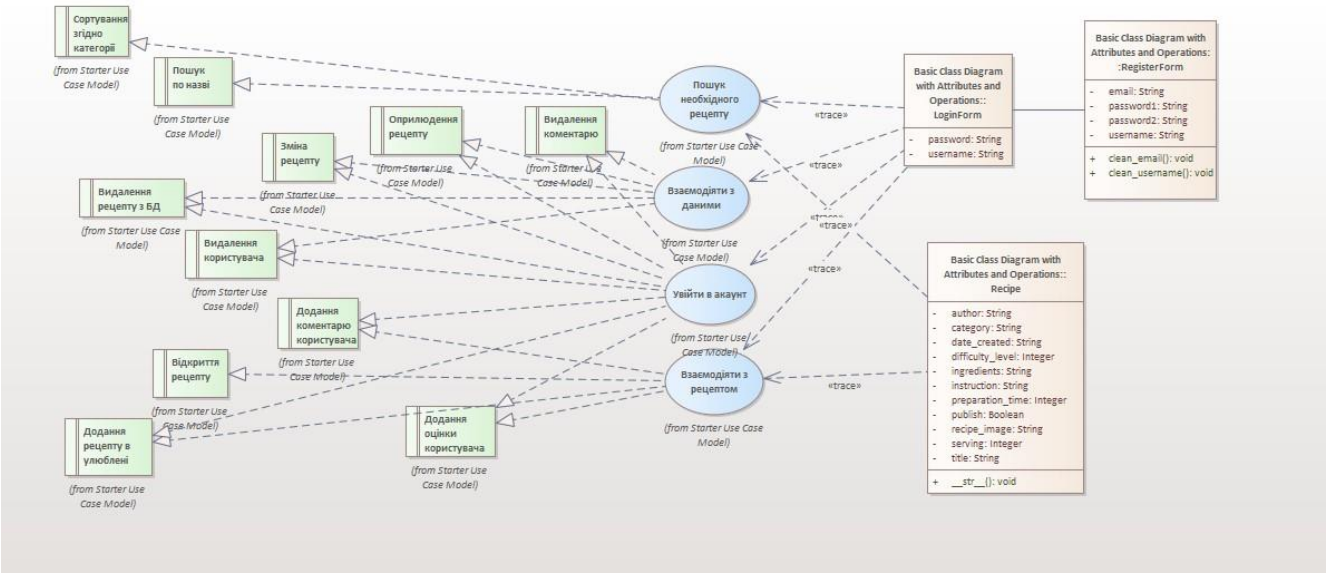


Рисунок 2.14 – Діаграма трасування вимог

Для глибшого розуміння взаємозв'язків, які відображені на діаграмі трасування, була розроблена матриця взаємозв'язків, яка відображає ті ж самі зв'язки (див. рис. 4.5).

	Пошук необхідного рецепту	Взаємодія з рецептом	Увійти в акаунт	Взаємодіяти з даними
Пошук по назві	X			
Сортування згідно категорії	X			
Відкриття рецепту		X		
Додання оцінки користувача		X	X	
Додання рецепту в улюблені		X	X	
Додання коментаря		X	X	
Видалення коментарю			X	X
Оприлюднення рецепту			X	X
Зміна рецепту			X	X
Видалення рецепту з БД			X	X
Видалення користувача			X	X

Рисунок 2.15 – Матриця взаємозв’язків

РОЗДІЛ 3. ПРОЕКТУВАННЯ СИСТЕМИ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

Інформаційне забезпечення веб-платформи для обміну кулінарними рецептами і порадами включає кілька ключових складових, організованих у єдиній базі даних для забезпечення ефективної роботи системи. Основні компоненти цієї бази даних включають таблиці користувачів, де зберігається інформація про зареєстрованих користувачів, їхні особисті дані, налаштування, уподобання та історія активності. Інша таблиця містить усі рецепти, додані користувачами, їхні інгредієнти, інструкції, фотографії та відео. Коментарі та оцінки користувачів зберігаються в окремих таблицях, що дозволяє відстежувати відгуки та рейтинги для кожного рецепту. Також є таблиця категорій, яка містить інформацію про різні категорії та підкатегорії рецептів, що полегшує їх пошук і організацію.

Принципи організації інформаційного забезпечення передбачають централізоване зберігання даних з можливістю швидкого доступу та обробки. Для цього використовується система керування базами даних (СКБД) SQLite, яка забезпечує легку інтеграцію, низькі вимоги до ресурсів і достатню продуктивність для невеликих та середніх за розміром проектів. Вибір SQLite обґрунтовується її здатністю ефективно обробляти локальні бази даних, простотою налаштування та відсутністю необхідності в окремому сервері баз даних, що спрощує розгортання та обслуговування системи.

Тип носія даних переважно цифровий, що зберігається на локальних серверах або у хмарних сховищах з високим рівнем безпеки. Методи контролю інформації включають регулярне резервне копіювання, перевірку цілісності даних та шифрування конфіденційної інформації. Вимоги до надійності і достовірності інформації включають забезпечення безперебійного доступу до

даних, захист від несанкціонованого доступу та регулярне оновлення систем безпеки.

Загальна схема інформаційного забезпечення включає серверну інфраструктуру, що забезпечує зберігання та обробку даних, клієнтські пристрої для доступу користувачів до платформи та програмне забезпечення, яке забезпечує інтерфейс між користувачами і базами даних. Перелік конкретних елементів, які використовуються при функціонуванні інформаційної системи, включає сервери баз даних, веб-сервери, мережеве обладнання та резервні сховища.

Організація збору і передавання первинної інформації на веб-платформі обміну кулінарними рецептами і порадами базується на кількох ключових джерелах та носіях інформації, що забезпечують ефективну та своєчасну обробку даних.

Основними джерелами інформації є користувачі платформи, які подають дані через веб-інтерфейс. Ці дані включають реєстраційну інформацію, рецепти, коментарі, оцінки та іншу активність користувачів. Користувачі мають можливість подавати ці дані в режимі реального часу під час їхньої взаємодії з платформою.

Адміністратори та модератори платформи відіграють важливу роль у контролі якості та достовірності інформації. Вони відповідають за перевірку та затвердження нових рецептів, модерацію коментарів і рейтингів, а також за видалення неприйнятного контенту. Це забезпечує підтримку високих стандартів якості та безпеки на платформі.

Для кожного вхідного повідомлення вказуються джерела інформації. Наприклад, джерелом інформації для нових рецептів є зареєстровані користувачі, які додають рецепти через свої акаунти. Адміністратори та модератори забезпечують своєчасну перевірку та обробку цих даних.

Періодичність надання інформації залежить від типу даних та активності користувачів. Дані про нові рецепти, коментарі та оцінки надходять у реальному часі.

Основним носієм інформації є база даних SQLite, яка зберігає всі дані про користувачів, рецепти, коментарі, оцінки та категорії. Веб-інтерфейс платформи забезпечує користувачам зручний спосіб введення та оновлення інформації. Інформація зберігається у відповідних таблицях бази даних, що дозволяє ефективно керувати та обробляти великі обсяги даних.

Для забезпечення надійності та достовірності даних використовуються механізми аутентифікації та авторизації користувачів, регулярні резервні копії бази даних, а також системи моніторингу та перевірки коректності введених даних. Це дозволяє підтримувати високу якість інформації та забезпечує безперебійну роботу платформи.

Основними об'єктами класифікації у базі даних є користувачі, рецепти, категорії рецептів, рівні складності приготування, коментарі, рейтинги та обрані рецепти. Для користувачів використовується унікальний ідентифікаційний номер (ID), який є первинним ключем у базі даних. Цей ID автоматично генерується при створенні нового користувача і забезпечує унікальність кожного запису. Цифровий код, який автоматично генерується при реєстрації користувача, має структуру цілісного числа (наприклад, 1, 2, 3, ..., 1000, 1001 тощо).

Категорії рецептів класифікуються для зручності навігації та пошуку. Цифровий код для ідентифікації категорії має структуру цілісного числа (наприклад, 1 для "Випічка", 2 для "М'ясні страви").

Для рівнів складності приготування, категорій рецептів, рецептів, коментарів, рейтингів та обраних рецептів також використовуються унікальні ідентифікаційні номери. Ці числа генеруються автоматично при створенні нового запису відповідної таблиці.

Правильна організація та структура інформаційних масивів є критично важливою для забезпечення ефективної роботи системи, швидкого доступу до даних і надійного зберігання інформації. Нижче наведено опис ключових масивів у вигляді таблиць, що містять назви полів, типи даних і короткий опис кожного поля.

Одним із ключових масивів, що забезпечують роботу веб-застосунку, є масив RECIPE, який включає всі існуючі рецепти. Ці рецепти є основним матеріалом для обміну кулінарною інформацією між користувачами. Кожен рецепт містить детальну інформацію про інгредієнти, інструкції з приготування, поради та фотографії готової страви. Структура рецепту включає такі поля, як назва страви, складники, покрокові інструкції, автор рецепту, дата створення, категорія страви.

Масив USER включає інформацію про всіх користувачів платформи. Це дозволяє ідентифікувати користувачів та відстежувати їхню активність.

Масив COMMENT містить коментарі користувачів до рецептів. Ці коментарі відіграють важливу роль у формуванні спільноти користувачів та обговоренні кулінарних питань. Коментарі дозволяють обмінюватися враженнями, порадами та досвідом, а також задавати питання авторам рецептів або іншим користувачам. Документи коментарів повинні бути структуровані таким чином, щоб забезпечити зручність читання та модерації.

Масив RATING включає рейтинги, які користувачі надають рецептам. Це допомагає визначити найпопулярніші та найкорисніші рецепти на платформі.

Масив FAVORITE_RECIPE зберігає улюблені рецепти користувачів. Це дозволяє користувачам швидко знаходити та зберігати рецепти, які їм сподобалися, для подальшого використання.

Масив DIFFICULTY_LEVEL включає рівні складності приготування страв. Це допомагає користувачам оцінити, наскільки складно приготувати ту чи іншу страву, і обрати рецепти відповідно до своїх навичок та досвіду.

Масив CATEGORY містить категорії рецептів, що допомагає організувати рецепти за різними критеріями, такими як тип страви, регіональна кухня, дієтичні обмеження тощо. Це полегшує користувачам пошук необхідних рецептів і орієнтацію серед великого обсягу інформації.

Опис масивів:

Таблиця 3.1

Найменування масиву — масив всіх існуючих рецептів.
Ідентифікатор масиву — RECIPE.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 5000 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — RECIPE.

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	PK	01–32	так	ІДД	—
Назва рецепту	**title**	—	String	—	—	так	—	—
Інгрідієнти	**ingredients**	—	String	—	—	так	—	—
Інструкція	**instruction**	—	String	—	—	так	—	—
Час готування	**preparation_time**	—	String	—	—	так	—	—
Кількість порцій	**serving**	—	Number	—	—	так	—	—
Складність приготування	**difficulty_level**	—	Object	FK	—	так	—	DIFFLEVEL
Категорія страви	**category**	—	Object	FK	—	так	—	CATEGORY
Фото страви	**recipe_image**	—	Image	—	—	так	—	—

Таблиця 3.2

Найменування масиву — масив користувачів.

Ідентифікатор масиву — USER.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 40 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — USER.

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	РК	01–32	так	ІДД	—
Нік користувача	**username**	—	String	—	—	так	—	—
Чи є користувач адміністратором?	**is_superuser**	—	Bool	—	—	так	—	—
Електронна пошта	**email**	—	String	—	—	так	—	—
Пароль	**password**	—	String	—	—	так	—	—
Останній логін	**last_login**	—	Date	—	—	так	—	—
Перший логін	**date_joined**	—	Date	—	—	так	—	—

Таблиця 3.3

Найменування масиву — масив коментарів.

Ідентифікатор масиву — COMMENT.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 500 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — COMMENT.

Продовження таблиці 3.3

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	PK	01–32	так	ІДД	—
Текст коментаря	**text**	—	String	—	—	так	—	—
Автор коментаря	**author**	Act_u	Object	FK	—	так	—	USERS
Рецепт коментаря	**recipe**	—	Object	FK	—	так	—	RECIPES
Дата створення	**date_created**	—	Date	—	—	так	—	—

Таблиця 3.4

Найменування масиву — масив рейтингів.
Ідентифікатор масиву — RATING.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 30 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — RATING.

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	PK	01–32	так	ІДД	—
Оцінка користувача	**rating**	u_rat_d	Number	—	—	так	—	—
Автор оцінки	**user**	u	Object	FK	—	так	—	USERS
Рецепт	**recipe**	—	Object	FK	—	так	—	RECIPES

Таблиця 3.5

Найменування масиву — масив улюблених рецептів.

Ідентифікатор масиву — FAVORITE_RECIPE.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 30 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — FAVORITE_RECIPE.

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	PK	01–32	так	ІДД	—
Рецепт	**recipe**	—	Object	FK	—	так	—	RECIPES
Користувач	**user**	—	Object	FK	—	так	—	USERS

Таблиця 3.6

Найменування масиву — масив складності приготування.

Ідентифікатор масиву — DIFFCULTY_LEVEL.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 30 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — DIFFCULTY_LEVEL.

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	PK	01–32	так	ІДД	—
Назва складності	**name**	—	String	—	—	так	—	—

Таблиця 3.7

Найменування масиву — масив категорій рецептів.
Ідентифікатор масиву — CATEGORY.
Найменування носія інформації — БД.
Максимальний обсяг масиву — 10 000 записів.
Довжина запису — 30 символів (або байт).
Метод організації — послідовний (прямий).
Ключі упорядкування — код структурного підрозділу, табельний номер.
Ідентифікатор індексного масиву — CATEGORY.

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи систематичні зв'язки
				Первинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Унікальний ідентифікатор	**id**	—	Number	РК	01–32	так	ІДД	—
Назва категорії	**name**	—	String	—	—	так	—	—
Фотографія категорії	**category_image**	—	Image	—	—	так	—	—

Для реалізації проекту було обрано систему керування базами даних (СКБД) SQLite. Цей вибір обґрунтовано декількома ключовими факторами, які роблять SQLite оптимальним рішенням для даного проєкту.

SQLite є чудовим вибором завдяки своїй простоті, легкості у використанні та ефективності. Однією з головних причин для вибору саме SQLite є її вбудованість та відсутність необхідності в окремому серверному процесі. Це означає, що базу даних можна легко розгорнути разом з додатком без додаткових налаштувань серверного програмного забезпечення, що значно спрощує процес розробки та розгортання.

Важливою перевагою SQLite є її компактність та економічне використання ресурсів. База даних зберігається у вигляді одного файлу на диску, що робить її легкою у керуванні та переносі. Це особливо корисно для платформи, оскільки дозволяє уникнути складних процедур резервного

копіювання та відновлення даних. Крім того, таке рішення значно знижує витрати на інфраструктуру, що є вагомим фактором для стартапів та невеликих проектів.

SQLite має чудову продуктивність для невеликих та середніх проектів, що робить її ідеальним вибором для веб-платформи. Вона здатна обробляти тисячі запитів на секунду без значних затримок, що забезпечує швидкий доступ до рецептів та коментарів користувачів. Це означає, що можна гарантувати швидкий відгук системи навіть при великій кількості одночасних користувачів, що є критично важливим для забезпечення позитивного досвіду користувачів.

Крім того, SQLite підтримує повний набір функцій SQL, що дозволяє розробникам використовувати всі переваги реляційних баз даних. Це включає складні запити, транзакції та цілісність даних, що забезпечує надійність та консистентність даних у системі. Для платформи це означає, що можна реалізувати всі необхідні функції, такі як пошук рецептів, фільтрація за категоріями та управління коментарями, з використанням потужних можливостей SQL.

Ще однією важливою причиною вибору SQLite є її висока портативність. База даних, збережена у вигляді одного файлу, може бути легко перенесена з одного середовища на інше, що робить її ідеальною для розробки та тестування.

SQLite також відома своєю надійністю та стабільністю. Вона широко використовується у різних додатках, від мобільних до настільних, і має довгу історію безвідмовної роботи. Це означає, що можна покладатися на SQLite як на надійне рішення для зберігання та управління даними, не турбуючись про можливі збої або втрату даних. Її стабільність підтверджена численними проектами та організаціями, які використовують її у своїх системах.

Останньою, але не менш важливою причиною вибору SQLite є її підтримка великим співтовариством розробників та наявність обширної

документації. Це забезпечує доступ до широкого спектра ресурсів, включаючи навчальні матеріали, приклади коду та рішення поширених проблем. Для команди розробників це означає, що завжди можна знайти необхідну інформацію та отримати допомогу від спільноти у разі потреби, що значно прискорює процес розробки та впровадження нових функцій.

Таким чином, вибір SQLite для веб-платформи обміну кулінарними рецептами та порадами є виправданим і раціональним. Її вбудованість, компактність, висока продуктивність, повна підтримка SQL, портативність, надійність та підтримка співтовариством роблять її ідеальним рішенням для цього проекту. Використання SQLite дозволить створити стабільну, ефективну та зручну платформу для користувачів, забезпечуючи швидкий доступ до інформації та надійне зберігання даних.

Інфологічна модель бази даних використовується на етапі аналізу вимог до системи, коли важливо з'ясувати, які дані потрібно зберігати і як вони взаємодіють між собою. Вона є основою для подальшого проектування фізичної моделі бази даних, в якій вже враховуються деталі технічної реалізації.

Інфологічна модель може бути представлена у вигляді схеми бази даних, діаграми сутностей-зв'язків (ER-діаграми) або іншими методами візуалізації. Вона дозволяє команді проектувальників баз даних та розробникам програмного забезпечення зрозуміти потреби бізнесу та відобразити їх у відповідній структурі даних, що сприяє ефективнішій і раціональній реалізації системи.

Для розробки інфологічної моделі, яка відображає логічну структуру бази даних, було використано онлайн-застосунок Draw.io (див. рис. 3.1). Цей інструмент надає зручний інтерфейс для створення діаграм, що дозволяє візуалізувати взаємозв'язки між сутностями та атрибутами в базі даних.

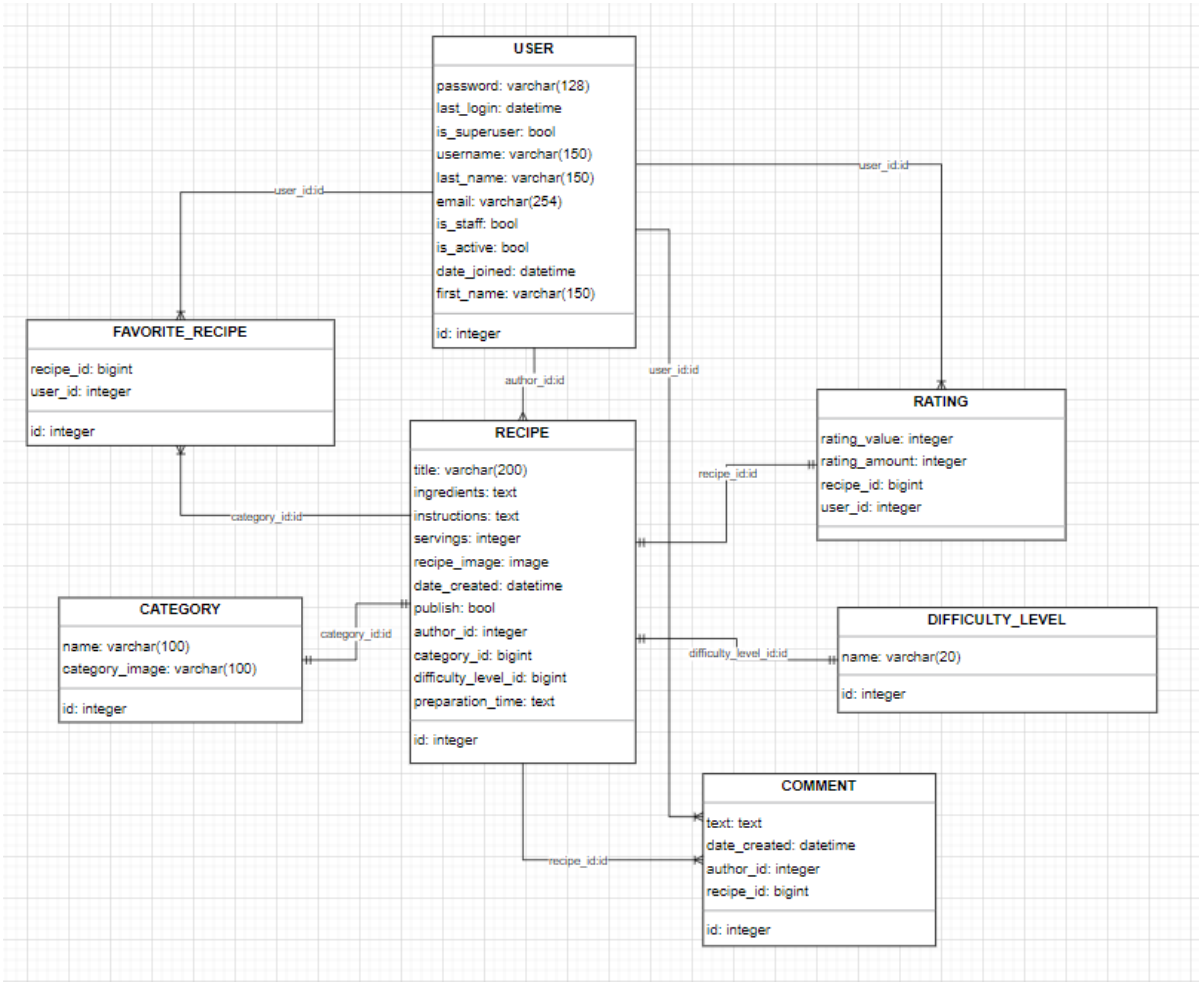


Рисунок 3.1 – Схема інфологічної моделі БД

3.2. Технічне забезпечення

Веб-платформа для обміну кулінарними рецептами і порадами буде функціонувати на віддаленому сервері, розташованому в дата-центрі. Ядро системи, база даних та необхідні файли будуть зберігатися на цьому сервері. Доступ користувачів до системи буде здійснюватися через веб-сайт за допомогою браузера з доступом до мережі Інтернет. Віддалений сервер забезпечить високу доступність та безперебійну роботу платформи. Система функціонуватиме у глобальній мережі Інтернет, що дозволить користувачам з усього світу переглядати рецепти, залишати коментарі та додавати власні рецепти.

Комплекс технічних засобів (КТЗ) включає серверну інфраструктуру, мережеве обладнання та користувацькі пристрої. Серверна інфраструктура

складається з основного сервера, на якому розміщене програмне забезпечення платформи, база даних (SQLite), а також резервного сервера для забезпечення відмовостійкості. Основний сервер відповідає за обробку запитів користувачів, збереження та обробку даних, а резервний сервер автоматично приймає на себе функції основного у разі його виходу з ладу. Мережеве обладнання включає маршрутизатори та комутатори, які забезпечують надійний та безпечний доступ до серверів, а також захист від зовнішніх атак та несанкціонованого доступу. Користувацькі пристрої, такі як комп'ютери, планшети та смартфони, використовуються для доступу до веб-сайту платформи. Всі пристрої мають можливість підключатися до Інтернету через дротові або бездротові мережі. Схему мережі передачі даних наведено нижче (див. рис. 3.2)

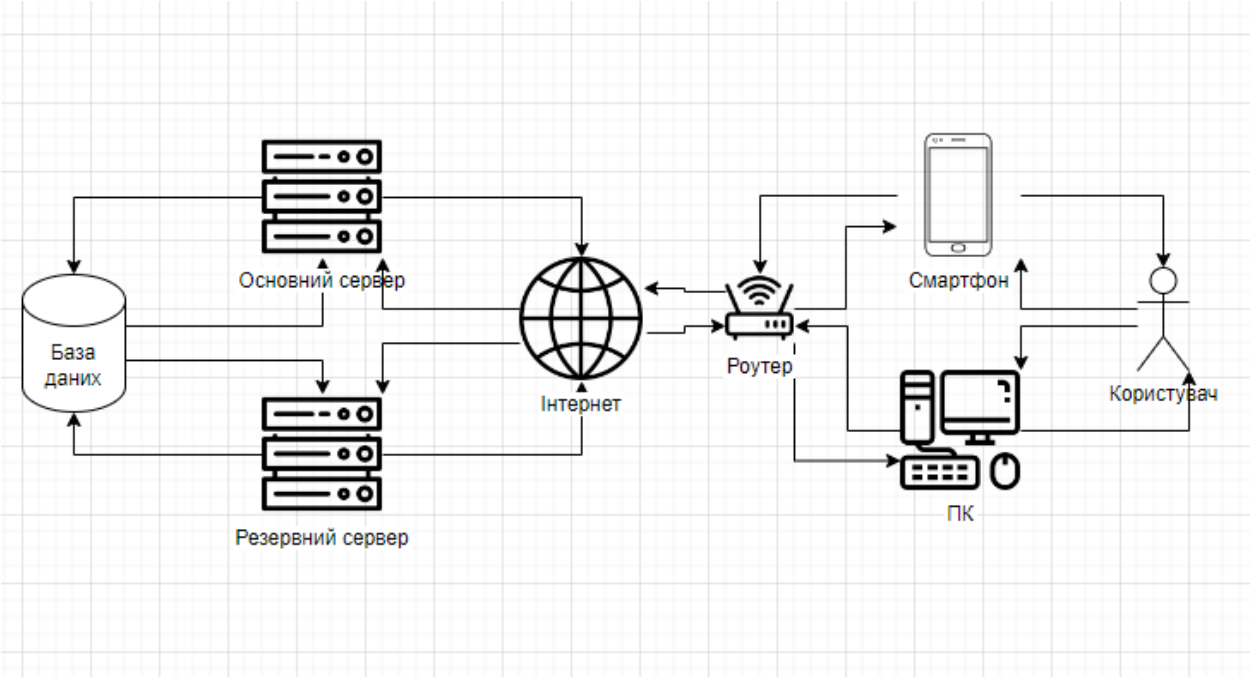


Рисунок 3.2 – Схема мережі передачі даних

Для забезпечення роботи платформи обрано віддалений серверний хостинг, який зможе забезпечити високу доступність і надійність системи. Хостингом може виступати будь-який провайдер, що пропонує сервери з наступними, або кращим технічними характеристиками: процесор з частотою

3,5 ГГц, 16 ГБ оперативної пам'яті, 500 ГБ SSD-дискового простору. Сервер повинен бути розташований у дата-центрі з високим рівнем захисту і дотриманням технічних умов експлуатації. Обмін даними з іншими автоматизованими системами здійснюватися через API, що б дозволило інтеграцію з іншими сервісами та платформами.

Вибір такої структури обумовлений необхідністю забезпечення безперервної роботи платформи та можливістю масштабування у разі збільшення навантаження. Мінімально допустима кількість автоматизованих робочих місць (АРМ) для підтримки функціонування системи включає одне АРМ для адміністратора платформи, який здійснює моніторинг, управління контентом та користувачами.

Для забезпечення безпеки технічних засобів від механічних, теплових, електромагнітних та інших впливів, а також від несанкціонованого доступу, використовується комплексний підхід. Сервери розміщені в дата-центрі, який відповідає стандартам безпеки і має системи контролю доступу, відеоспостереження, охоронні системи, а також засоби захисту від пожежі та затоплень. Доступ до серверів здійснюється через захищені канали зв'язку з використанням SSL/TLS шифрування.

Вибір типу серверів обґрунтований нефункціональними вимогами системи, такими як висока доступність, безперервність роботи, масштабованість та безпека. Вирішення щодо типу серверів та їх розміщення буде прийнято з урахуванням потенційного навантаження, вартості транзакції та інших технічних вимог.

Технічні засоби, розміщені в обчислювальній мережі об'єкта, включають сервери, маршрутизатори та комутатори. Використовуються стандартні мережеві комутатори та маршрутизатори, які забезпечують швидкий та безперебійний обмін даними між користувачами та сервером.

Периферійні технічні засоби в даній інформаційній системі не використовуються, оскільки платформа працює у веб-середовищі і доступна

через браузер на різних користувацьких пристроях (смартфони, планшети, ноутбуки). Для відображення інформації використовується стандартний веб-інтерфейс, який забезпечує зручний доступ до всіх функцій платформи.

Чисельність персоналу, що забезпечує функціонування технічних засобів, мінімальна. Основні функції адміністрування та підтримки здійснює один системний адміністратор, оскільки більшість технічних процесів автоматизовані, а серверне обладнання розміщене у дата-центрі провайдера, який забезпечує технічне обслуговування та підтримку інфраструктури.

Автоматизоване робоче місце (АРМ) адміністратора платформи включає персональний комп'ютер або ноутбук з доступом до Інтернету, а також спеціалізоване програмне забезпечення для управління системою. Адміністратор використовує веб-інтерфейс для моніторингу роботи системи, додавання нових рецептів, редагування існуючих рецептів, модерації коментарів та управління користувачами. Веб-інтерфейс надає зручний доступ до всіх необхідних функцій, забезпечуючи ефективне виконання завдань.

Для забезпечення коректної роботи АРМ було визначено мінімальні технічні вимоги, а саме:

- 2х ядерний процесор з базовою частотою 2.5 ГГц;
- 4 ГБ оперативної пам’яті;
- 500 ГБ об’єму пам’яті на жорсткого диску;
- Порт Ethernet.

Для повноцінної роботи АРМ також необхідне використання периферійних технічних засобів, таких як монітори, клавіатури, миші та інші. Монітор з діагоналлю 21 дюйм забезпечить комфортний перегляд інформації та зручну роботу з платформою. Клавіатура повинна мати стандартний розмір для забезпечення зручного введення тексту. Миша повинна бути оптичною з мінімальною кількістю кнопок для ергономічного використання.

3.3. Програмне забезпечення

Програмне забезпечення — це сукупність інструкцій, які керують апаратним забезпеченням комп'ютера та виконують конкретні завдання. Його можна розділити на системне та прикладне програмне забезпечення. Системне програмне забезпечення включає операційні системи, такі як Windows, macOS, Linux, які керують ресурсами комп'ютера та забезпечують основу для роботи прикладних програм. Прикладне програмне забезпечення — це програми, які виконують конкретні користувацькі задачі, наприклад, текстові редактори, електронні таблиці, графічні редактори та браузери.

Загалом, програмне забезпечення є невід'ємною частиною сучасного світу, значно полегшуючи різні аспекти нашого життя, від професійної діяльності до розваг. Розвиток цієї галузі постійно триває, відкриваючи нові можливості та вдосконалюючи вже існуючі технології.

Схема програмного забезпечення складається з декількох шарів, кожен з яких відповідає за певні функції та забезпечує безперебійну роботу системи. Нижче представлена укрупнена схема ПЗ, яка включає базове програмне забезпечення, прикладне програмне забезпечення та програмну документацію.

Базове програмне забезпечення є фундаментом для роботи всієї системи. Воно забезпечує основне функціонування комп'ютера і надає середовище для виконання прикладних програм. Для нашої платформи базове ПЗ включає:

- Операційна система: Windows 10;
- Фреймворк: Django;
- База даних: SQLite;
- Середовище виконання: Python;

Windows є однією з найпопулярніших операційних систем, розроблених Microsoft. Вона широко використовується на персональних комп'ютерах і серверах. Windows відома своїм дружнім графічним інтерфейсом і великою кількістю підтримуваних програм і драйверів, що робить її популярним вибором серед користувачів і розробників. Операційна система забезпечує

високу сумісність із різними апаратними засобами та підтримує широкий спектр програмних продуктів, включаючи серверні програми, бази даних, веб-сервіси тощо. Завдяки інтеграції з Azure та іншими продуктами Microsoft, такими як Visual Studio, Windows забезпечує потужне середовище для розробки, тестування та розгортання веб-програм.

Django є одним із найбільш популярних і потужних фреймворків для веб-розробки, створених на мові програмування Python. Рішення використовувати Django для веб-платформи базується на ряді ключових переваг, які він надає, забезпечуючи ефективну розробку, масштабованість і безпеку проекту.

По-перше, одним із головних аргументів на користь Django є його принцип "не повторюй себе" (DRY - Don't Repeat Yourself). Цей принцип дозволяє зменшити кількість повторюваного коду і зосередитися на написанні логіки додатку, що значно прискорює процес розробки. Використання повторно застосовуваних компонентів і шаблонів у Django дозволяє легко керувати складними проектами, зберігаючи при цьому чистоту і зрозумілість коду. Завдяки цьому принципу, розробники можуть уникнути дублювання коду, що не тільки спрощує підтримку і оновлення системи, але й знижує кількість помилок, які можуть виникнути через повторюваність коду.

Інша важлива перевага полягає в тому, що Django має вбудовані засоби для забезпечення безпеки веб-додатків. Фреймворк включає механізми захисту від поширених веб-атак, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS), міжсайтові підробки запитів (CSRF) та інші. Це дозволяє значно знизити ризик безпеки, забезпечуючи надійний захист даних користувачів і стабільну роботу системи. Вбудовані засоби безпеки Django дозволяють розробникам зосередитися на функціональності додатку, знаючи, що основні аспекти безпеки вже покриті фреймворком.

Ще однією вагомою причиною вибору Django є його масштабованість. Фреймворк здатний підтримувати високу навантаженість, завдяки чому

платформа може розширюватися і обслуговувати зростаючу кількість користувачів без втрати продуктивності. Django використовується в багатьох великих проектах, таких як Instagram, Disqus і Mozilla, що підтверджує його здатність справлятися з великими обсягами даних і високим трафіком. Це означає, що проект може починати з невеликої бази користувачів і поступово масштабуватися до мільйонів користувачів без необхідності переходу на інший фреймворк чи переписування коду.

Додатково, Django забезпечує швидкий і зручний процес розробки завдяки своєму вбудованому адміністративному інтерфейсу. Адміністратори можуть легко керувати контентом платформи, користувачами і налаштуваннями через автоматично згенеровану панель адміністрування. Це дозволяє значно скоротити час на створення та налаштування власних адміністративних інтерфейсів. Адміністративний інтерфейс Django є дуже гнучким і дозволяє легко налаштовувати його під конкретні потреби нашої платформи, що є великим плюсом для швидкої і ефективної роботи адміністративного персоналу.

Ще одним аспектом, що робить Django привабливим вибором, є його велика і активна спільнота. Django має детальну документацію і багато ресурсів для навчання, що значно полегшує процес освоєння фреймворку як для початківців, так і для досвідчених розробників. Крім того, активна спільнота розробників постійно працює над оновленнями і покращеннями фреймворку, що забезпечує його актуальність і відповідність сучасним вимогам веб-розробки. Завдяки великій кількості навчальних матеріалів, форумів і груп підтримки, розробники можуть швидко знайти відповіді на свої запитання і вирішити технічні проблеми.

З технічної точки зору, Django пропонує потужний ORM (Object-Relational Mapping), який дозволяє працювати з базами даних більш ефективно і інтуїтивно зрозуміло. Завдяки ORM, розробники можуть взаємодіяти з базами даних, використовуючи Python-код, що зменшує кількість помилок і

підвищує продуктивність роботи. Це також полегшує міграції баз даних і зміни в структурі даних, що є важливим для динамічно змінюваних проектів. ORM Django підтримує роботу з різними типами баз даних, такими як SQLite, PostgreSQL, MySQL, що надає додаткову гнучкість у виборі СУБД.

Окрім того, Django забезпечує гнучкість і модульність завдяки своїй архітектурі. Розробники можуть легко інтегрувати сторонні бібліотеки і пакети, розширюючи функціональні можливості платформи. Наприклад, існує багато готових рішень для аутентифікації, платежів, обробки зображень та інших задач, які можна легко додати до проекту, використовуючи Django. Модульність Django дозволяє створювати власні розширення і компоненти, які легко інтегруються у вже існуючу структуру проекту.

Одним із важливих аспектів є також продуктивність Django. Завдяки оптимізованим механізмам кешування, обробки запитів і ефективному використанню ресурсів серверу, Django забезпечує високу швидкість роботи веб-додатків. Це особливо важливо для платформи, де користувачі очікують швидкого завантаження сторінок і миттєвого відгуку на їхні дії. Продуктивність фреймворку дозволяє знизити витрати на серверні ресурси і забезпечити позитивний користувацький досвід.

Не можна не згадати також про тестування. Django має вбудовані інструменти для тестування, які дозволяють розробникам автоматизувати процес перевірки функціональності додатку. Це включає юніт-тести, інтеграційні тести і тести продуктивності, що допомагає виявляти і виправляти помилки на ранніх стадіях розробки. Використання тестування забезпечує високу якість коду і стабільність роботи системи, що є критично важливим для будь-якого веб-додатку.

Python є високорівневою мовою програмування, яка завоювала популярність завдяки своїй читабельності та простоті у вивченні. Мова була створена з акцентом на зручність використання і зниження складності коду, що дозволяє писати зрозумілий і компактний код.

Python підтримує різні парадигми програмування, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування. Це надає гнучкість у виборі підходу, який найкраще відповідає специфіці проекту та особистим вподобанням. Наприклад, об'єктно-орієнтоване програмування (ООП) дозволяє моделювати реальні об'єкти і їхні взаємодії, що може бути корисним для великих і складних систем. Процедурне програмування, зі свого боку, підходить для розробки структурованих програм, де основна увага приділяється функціям і процедурі їх виконання. Функціональне програмування акцентує увагу на використанні функцій як основних будівельних блоків програми, що дозволяє створювати більш модульний і тестований код.

Однією з ключових переваг Python є його велика стандартна бібліотека, яка включає модулі для роботи з файлами, інтернет-протоколами, управлінням процесами, серіалізації даних, математичними обчисленнями та багатьма іншими завданнями. Це дозволяє швидко знаходити та використовувати готові рішення для багатьох задач, що значно скорочує час розробки. Зокрема, для проекту з розробки веб-платформи для обміну кулінарними рецептами та порадами, стандартна бібліотека Python надає інструменти для роботи з базами даних, управління користувачами та обробки запитів, що є критично важливим для створення ефективного та надійного веб-застосунку.

Крім того, існує багато зовнішніх бібліотек і фреймворків, таких як Django для веб-розробки, NumPy для наукових обчислень, Pandas для обробки даних і багато інших. Використання Django у проекті веб-платформи дозволяє створити потужний та масштабований додаток з мінімальними зусиллями. Django забезпечує високий рівень безпеки, надає вбудовані механізми для аутентифікації та управління доступом, що є важливим для захисту користувацьких даних та забезпечення надійної роботи платформи.

Python також відомий своєю міжплатформеністю, що дозволяє розробляти програми, які можуть працювати на різних операційних системах,

таких як Windows, macOS і Linux, без значних змін у коді. Це робить Python ідеальним вибором для проектів, які потребують кросплатформеної підтримки. Його простий і зрозумілий синтаксис сприяє швидкому розвитку програмного забезпечення і зменшує кількість помилок. Написаний на Python код легше підтримувати і розширювати, що робить його популярним вибором для довготривалих проектів.

Крім того, спільнота Python є однією з найактивніших у світі програмування, постійно створюючи нові інструменти, бібліотеки та ресурси для навчання. Це забезпечує постійну підтримку та розвиток мови, а також доступ до великої кількості документації та прикладів коду. Усе це робить Python ідеальним вибором для розробки різноманітних проектів, включаючи веб-платформи для обміну кулінарними рецептами та порадами. Цей вибір забезпечує високу якість, надійність та зручність для користувачів платформи, сприяючи ефективному обміну кулінарною інформацією.

Прикладне програмне забезпечення забезпечує виконання конкретних завдань користувача. На нашій платформі воно включає всі функції, що дозволяють користувачам взаємодіяти з системою та обмінюватися кулінарними рецептами та порадами.

Користувацький інтерфейс створюється за допомогою технологій фронтенду: HTML відповідає за структуру веб-сторінок, CSS – за їх стилізацію, а JavaScript – за динамічність та інтерактивність. Використання фреймворку Bootstrap дозволяє швидко створювати адаптивні та привабливі інтерфейси.

Обробка даних здійснюється за допомогою моделей Django (див. додаток Г), які визначають структуру бази даних і забезпечують зручний спосіб маніпулювання даними через ORM (Object-Relational Mapping). Логіка обробки запитів реалізована у вигляді Django Views (див. додаток Д), які отримують дані з моделей, передають їх в шаблони для відображення і повертають відповіді користувачам.

Функціональність користувачів включає реєстрацію та авторизацію, що забезпечує безпеку і персоналізацію доступу до платформи. Зареєстровані користувачі можуть оцінювати рецепти, коментувати їх та додавати до улюблених, що забезпечує інтерактивність і залучення користувачів, дозволяючи їм взаємодіяти з контентом і один з одним. Адміністратори мають можливість додавати, редагувати та видаляти рецепти, що дає їм розширені права для керування контентом платформи. Переглянути панель адміністратора можна у додатках (див. додаток Б).

3.4 Результати реалізації інформаційної системи

Ця предметна галузь зосереджена на ефективному управлінні та документообігу в сфері інформаційної інтернет-підтримки кулінарії. Основні економічні задачі полягають у створенні ефективного механізму обміну кулінарною інформацією, включаючи рецепти, відгуки, поради та інші форми контенту. Від цього залежить розвиток та успішність кулінарної платформи, яка повинна відповідати високим стандартам якості та задовольняти потреби користувачів. Впровадження цифрових рішень дозволяє оптимізувати обмін інформацією, знижуючи витрати на адміністрування та підвищуючи ефективність комунікацій.

В рамках кваліфікаційного бакалаврського проекту було запропоновано низку інноваційних рішень, спрямованих на покращення процесів обміну кулінарною інформацією. Суть цих розробок полягає у створенні інтерактивної платформи, яка включає можливості для користувачів не лише отримувати, але й ділитися своїми кулінарними знаннями. Було розроблено ефективну методику систематизації та узагальнення кулінарного контенту, що включає класифікацію рецептів за різними параметрами, а також механізми інтерактивної взаємодії, такі як коментарі та оцінювання.

Розроблений прототип веб-застосунку став ключовим елементом даної роботи. Він забезпечує зручний доступ до великого обсягу кулінарних знань,

пропонуючи користувачам можливість обміну рецептами, їх коментування та оцінювання. Особлива увага була приділена створенню ефективного інтерфейсу, який дозволяє легко знаходити необхідну інформацію та сприяє активній участі користувачів.

Пілотне впровадження розробленого прототипу було здійснено локально на персональному комп'ютері, що дозволило зібрати цінні відгуки та провести тестування функціоналу в контрольованому середовищі. Для цього було написано тест кейси в Testrail, які допомогли детально перевірити роботу всіх модулів платформи. Локальне тестування включало збір даних про взаємодію з платформою, а також аналіз продуктивності та стабільності роботи застосунку.

Аналіз зібраних даних показав високий рівень задоволеності користувачів і підтвердив ефективність розроблених рішень. Було виявлено декілька областей для покращення, що дозволило внести необхідні корективи до прототипу. Локальне тестування забезпечило глибоке розуміння функціонування платформи і виявило ключові аспекти, що потребують подальшої оптимізації.

Розроблена платформа має значний потенціал для практичного використання, зокрема у сфері кулінарії. Основні переваги включають підвищення ефективності обміну кулінарною інформацією, що дозволяє користувачам швидко знаходити необхідні рецепти та поради. Важливою складовою економічної ефективності є монетизація платформи через рекламу. Розроблена модель монетизації забезпечує стабільний дохід та економічну вигоду для платформи, залучаючи рекламодавців та надаючи користувачам корисні рекламні пропозиції.

Забезпечення високого рівня безпеки та конфіденційності даних користувачів є ще одним важливим аспектом, який сприяє збереженню їхньої довіри та лояльності. Це дозволяє платформі зберігати свою репутацію та

приваблювати нових користувачів, що є ключовим фактором для її успішного функціонування.

На основі проведених досліджень та локального пілотного впровадження, розроблений прототип платформи готовий до реального використання. Основні рекомендації для подальшого розвитку включають регулярне тестування та вдосконалення платформи з метою виявлення та усунення можливих недоліків. Планується розширення функціоналу на основі зворотного зв'язку від користувачів, що дозволить підвищити привабливість та зручність платформи.

Також важливо розробити ефективну маркетингову стратегію для залучення нових користувачів та підвищення популярності платформи. Проведення додаткових досліджень та вдосконалень, таких як впровадження нових економіко-математичних методів, алгоритмів та технологій проектування, дозволить забезпечити успіх платформи на конкурентному ринку кулінарних веб-ресурсів.

Таким чином, розроблена платформа має високий потенціал для успішного впровадження та може значно покращити процеси обміну кулінарною інформацією, забезпечуючи високу якість контенту та задоволення користувачів.

ВИСНОВКИ

У процесі виконання кваліфікаційного бакалаврського проекту було досягнуто основної мети – систематизації та узагальнення сучасного досвіду з розробки нових рішень в галузі обміну кулінарною інформацією та розробки власного прототипу веб-застосунку. Дослідження охоплювало теоретичні, методичні та прикладні аспекти моделювання, проектування та програмування модулів, спрямованих на підвищення ефективності обміну кулінарною інформацією за допомогою веб-технологій.

Особливу увагу було приділено процесам ефективного управління та документообігу в галузі інформаційної інтернет-підтримки кулінарії. Проведений аналіз економічного змісту задач показав, що оптимізація цих процесів сприяє зниженню витрат, покращенню якості послуг та підвищенню загальної продуктивності.

Розроблений веб-застосунок було локально впроваджено та протестовано, що дозволило виявити та усунути недоліки, а також підтвердити відповідність розробки заявленим вимогам. Пілотне впровадження на локальному рівні показало високу ефективність функціоналу та зручність використання платформи для кінцевих користувачів.

Одним із ключових досягнень проекту стало створення інтерактивного інтерфейсу, що забезпечує зручність навігації та доступність інформації для користувачів з різним рівнем підготовки. Було розроблено механізми для обміну рецептами та коментарями, що сприяє активній участі користувачів та створенню динамічної спільноти.

Особливу увагу було приділено безпеці та конфіденційності даних користувачів, що є важливим фактором для збереження їхньої довіри та забезпечення надійного функціонування платформи. Запроваджені заходи дозволили створити безпечне середовище для обміну інформацією.

Узагальнюючи результати роботи, можна зробити висновок, що проект досягнув поставлених цілей, продемонструвавши високу ефективність та практичну цінність розроблених рішень. Основні рекомендації включають подальше вдосконалення функціоналу платформи на основі зворотного зв'язку від користувачів та впровадження нових технологій для підвищення конкурентоспроможності веб-застосунку.

Усі етапи виконання проекту, від аналізу потреб аудиторії до програмування та тестування функціоналу, було здійснено з урахуванням найкращих практик і сучасних методів розробки інформаційних систем. Цей процес сприяв набуттю глибоких знань та навичок у сфері розробки інформаційних систем, що є важливим для подальшої професійної діяльності.

Таким чином, розроблений веб-застосунок має високу практичну цінність і може бути успішно використаний як ефективний інструмент для обміну кулінарною інформацією, сприяючи створенню активної кулінарної спільноти та покращенню якості обслуговування користувачів. Переглянути загальний вигляд веб-застонку можна у додатках (див. додаток А).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сайт “Allrecipes”. [Електронний ресурс] – Режим доступу до ресурсу: https://www.allrecipes.com/. Дата звернення: 30.11.2023.

2. Сайт "Smachno.ua/ " [Електронний ресурс] – Режим доступу до ресурсу: https://smachno.ua/ua/. Дата звернення: 30.11.2023.

3. Репозиторій проекту на «GitHub» [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/farikan2018/RecipeSite. Дата звернення: 02.06.2024.

4. Прототип проекту у «Figma» [Електронний ресурс] – Режим доступу до ресурсу: https://www.figma.com/file/QYBN4zWRuKRkW2slxBrVkO/Laboratory-work-3.1?type=design&node-id=0-1&mode=design&t=VQV1mVWoUoI6P6Ux-0. Дата звернення: 14.04.2024.

5. Сайт “Visual Paradigm”. [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/. Дата звернення: 14.04.2024.

6. Сайт "LucidChart" [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-class-diagram. Дата звернення: 14.04.2024.

7. Сайт "SysML" [Електронний ресурс] – Режим доступу до ресурсу: https://sysml.org/sysml-faq/what-is-block-definition-diagram.html. Дата звернення: 14.04.2024.

8. Сайт "Sparx System" [Електронний ресурс] – Режим доступу до ресурсу: https://sparxsystems.com/. Дата звернення: 14.04.2024.

9. Сайт "Django documentation" [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/5.0/. Дата звернення: 02.06.2024.

10. Сайт "SQLite" [Електронний ресурс] – Режим доступу до ресурсу: https://sqlite.org/. Дата звернення: 02.06.2024.

11. Сайт "Apache" [Електронний ресурс] – Режим доступу до ресурсу: https://httpd.apache.org/. Дата звернення: 02.06.2024.

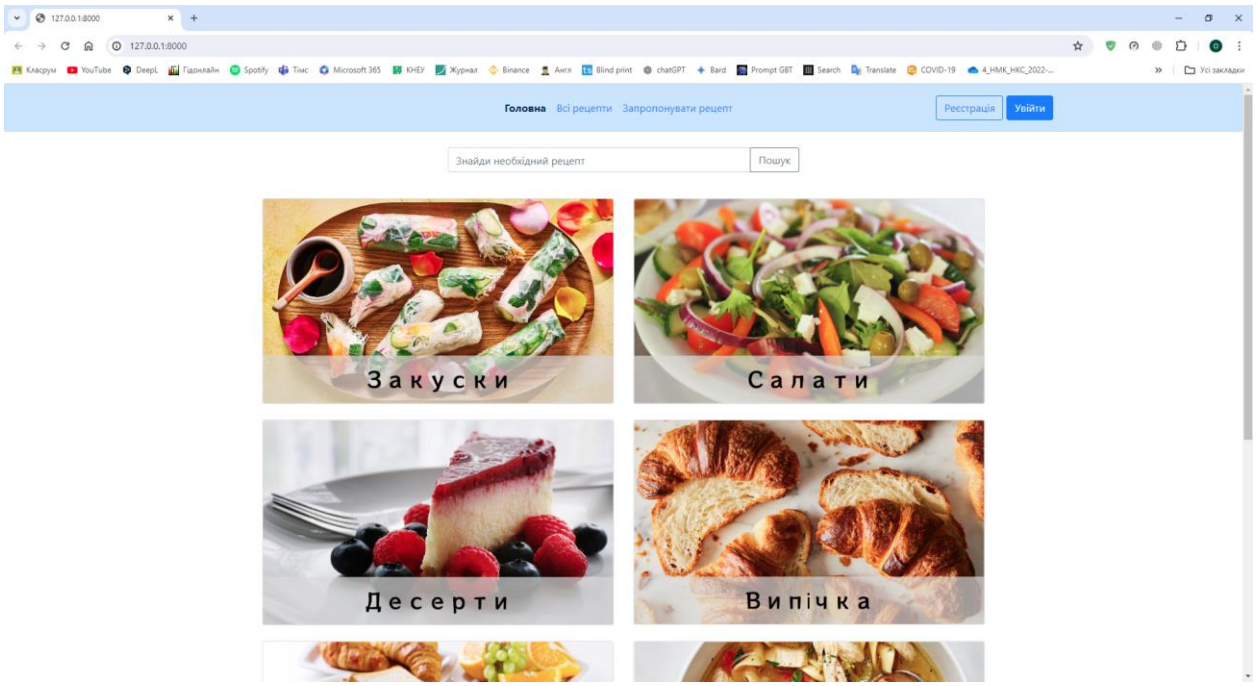
12. Сайт "Python" [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/. Дата звернення: 02.06.2024.

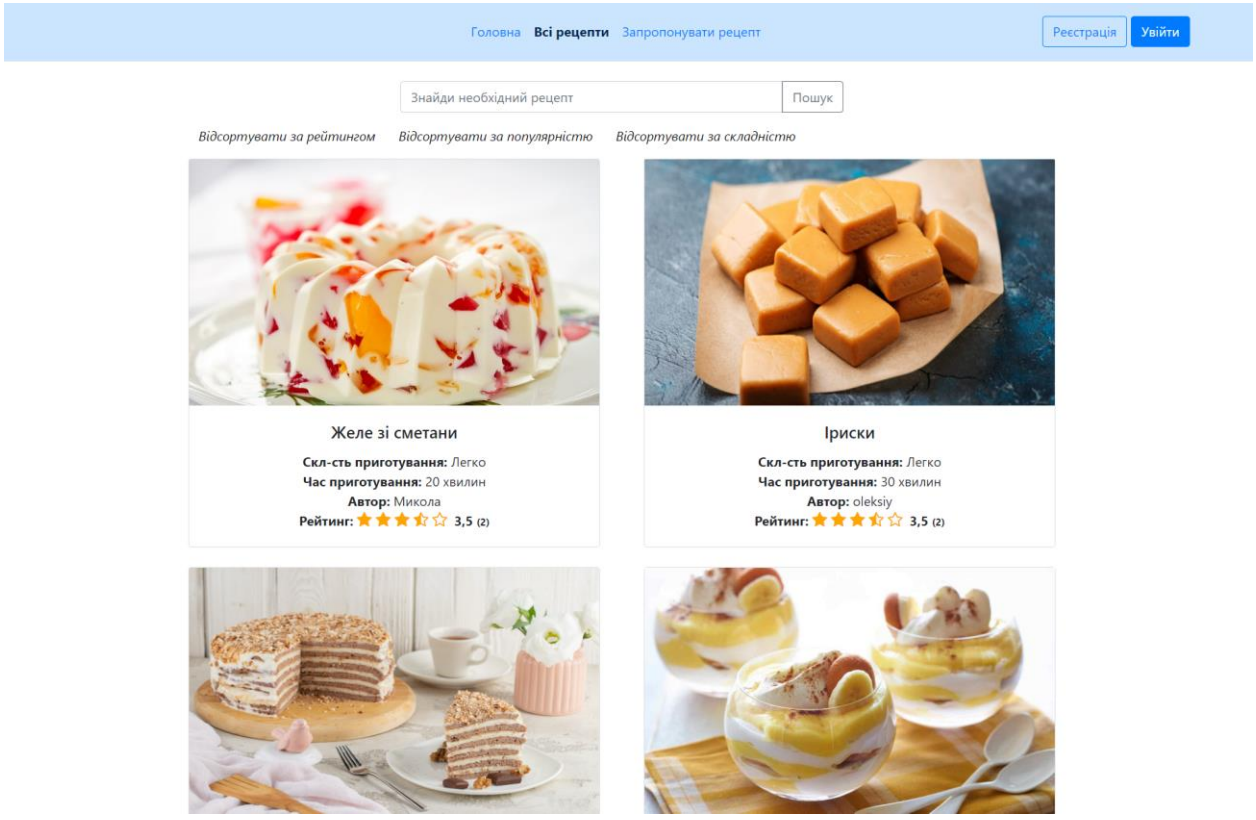
13. Сайт “FOOD LIKE ART.” " [Електронний ресурс] – Режим доступу до ресурсу: https://foodlikeart.blogspot.com/ Дата звернення: 02.06.2024.

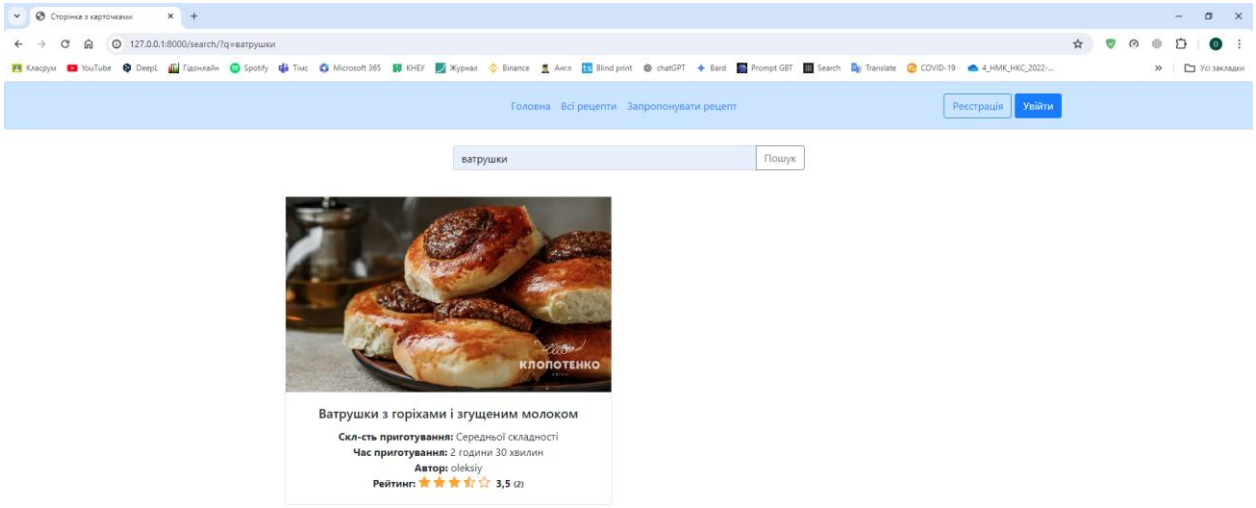
14. Сайт “Клопотенко” " [Електронний ресурс] – Режим доступу до ресурсу: https://klopotenko.com/ Дата звернення: 02.06.2024.

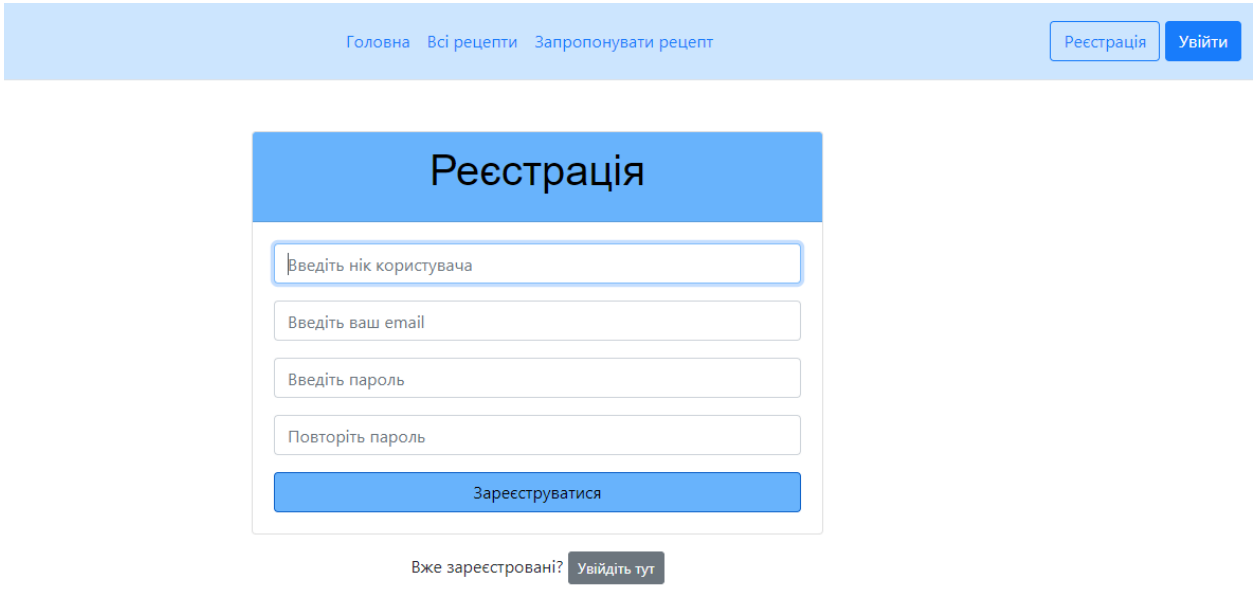
ДОДАТКИ

Додаток А – Загальний вигляд розробленого прототипу веб-застосунку






Сторінка з картками
127.0.0.1:8000/search/?q=ватрушки
Головна
Всі рецепти
Запропонувати рецепт
Реєстрація
Увійти
ватрушки
Пошук
КЛОПОТЕНКО
Ватрушки з горіхами і згущеним молоком
Скл-сть приготування: Середньої складності
Час приготування: 2 години 30 хвилин
Автор: oleksiy
Рейтинг: 3,5 (2)


Головна
Всі рецепти
Запропонувати рецепт
Реєстрація
Увійти
Реєстрація
Введіть нік користувача
Введіть ваш email
Введіть пароль
Повторіть пароль
Зареєструватися
Вже зареєстровані?
Увійдіть тут

Авторизація

Введіть нік користувача

Введіть пароль

Увійти

Немає акаунту? Зареєструйтесь тут

Головна Всі рецепти Запропонувати рецепт

Ласкаво просимо oleksiy | Вийти

Желе зі сметани

Складність приготування: Легко
Час приготування: 20 хвилин
Категорія: Десерти
Автор: Микола
Дата створення: 23 вересня 2023 р. 18:18
Рейтинг: 3,5(2)

Видалити з улюблених

Інгредієнти

15 г желатину
50 мл води
350 г сметани жирністю 20 %
50 г цукру

Інструкція

КРОК 1
15 г желатину розчиніть у 50 мл холодної води та залиште його набухати на 10 - 15 хв.

КРОК 2
350 г сметани кімнатної температури викладіть у миску і змішайте з 50 г цукру.

КРОК 3
Набухлий желатин розтопіть або на водяній бані, або, 1 хвилину прогрівши його в мікрохвильовці.

350 г сметани жирністю 20 %
50 г цукру

набухати на 10 - 15 хв.

КРОК 2
350 г сметани кімнатної температури викладіть у миску і змішайте з 50 г цукру.

КРОК 3
Набухлий желатин розтопіть або на водяній бані, або, 1 хвилину прогрівши його в мікрохвильовці.

КРОК 4
До сметани з цукром додайте желатин, розмішайте. Суміш сметани та желатину розлийте по піалах та поставте їх до холодильника застигати приблизно на 1 годину.

Додати коментар

Додати коментар

oleksiy
24 вересня 2023 р. 11:59

Микола твої рецепти найкращі!!!

Особистий кабінет

Головна Всі рецепти Запропонувати рецепт Ласкаво просимо oleksiy | Вийти

Мої рецепти

Іриски

Скл-сть приготування: Легко
Час приготування: 30 хвилин
Автор: oleksiy
Рейтинг: 3,5 (2)

Торт сметанник

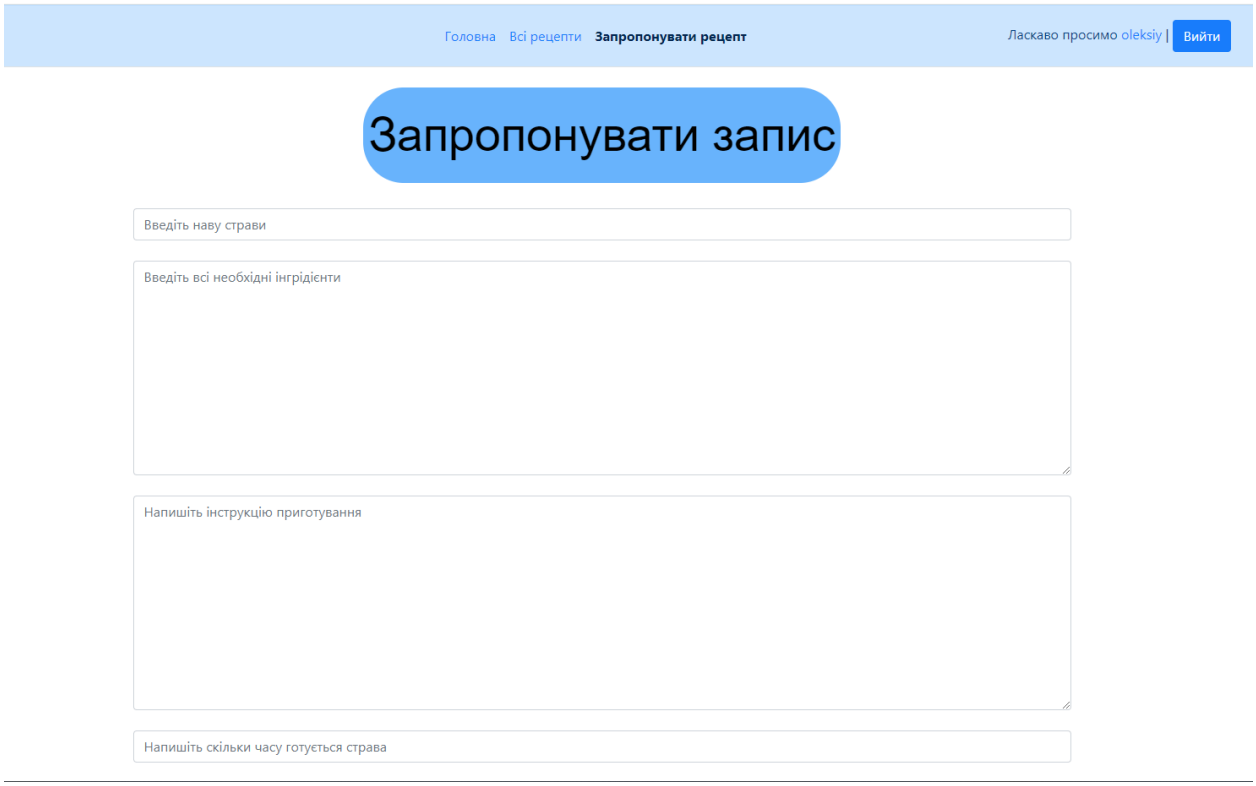
Скл-сть приготування: Складно
Час приготування: 2 години
Автор: oleksiy
Рейтинг: 0 (0)

Головна Всі рецепти Запропонувати рецепт Ласкаво просимо oleksiy | Вийти

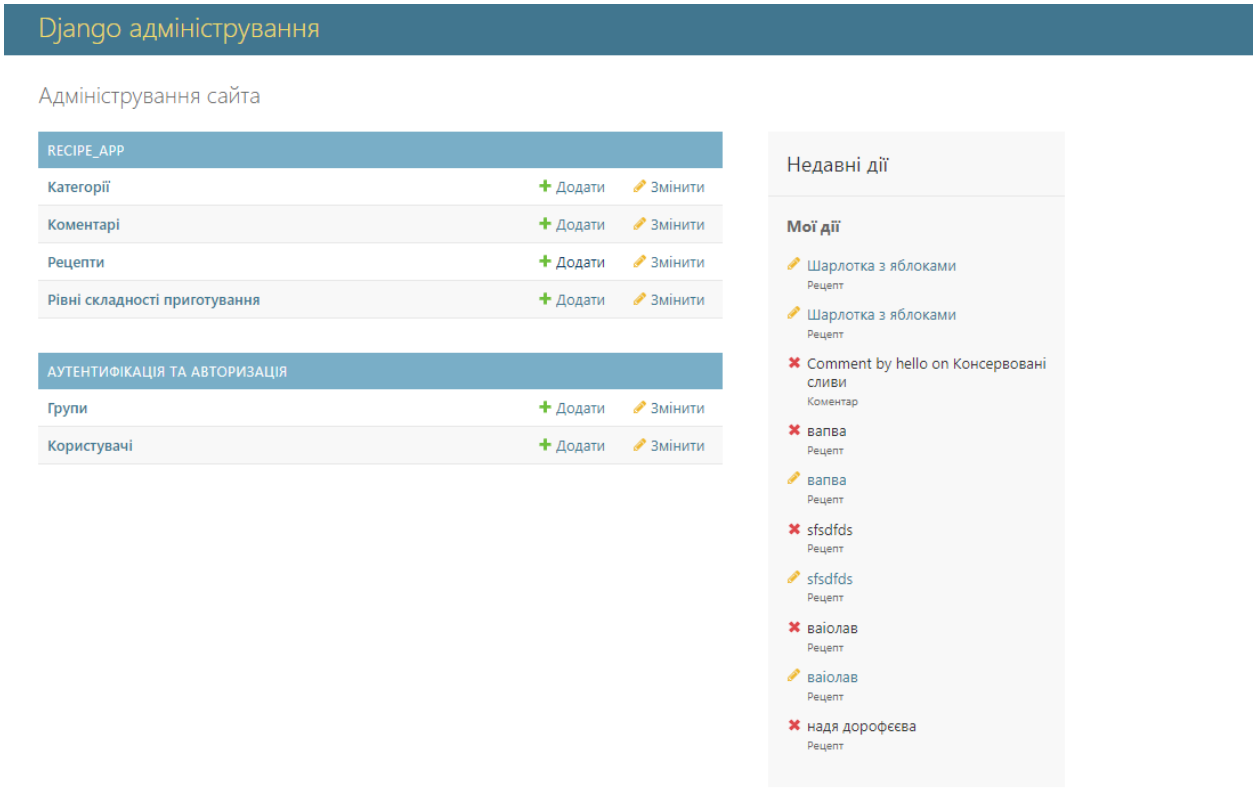
Улюблені рецепти

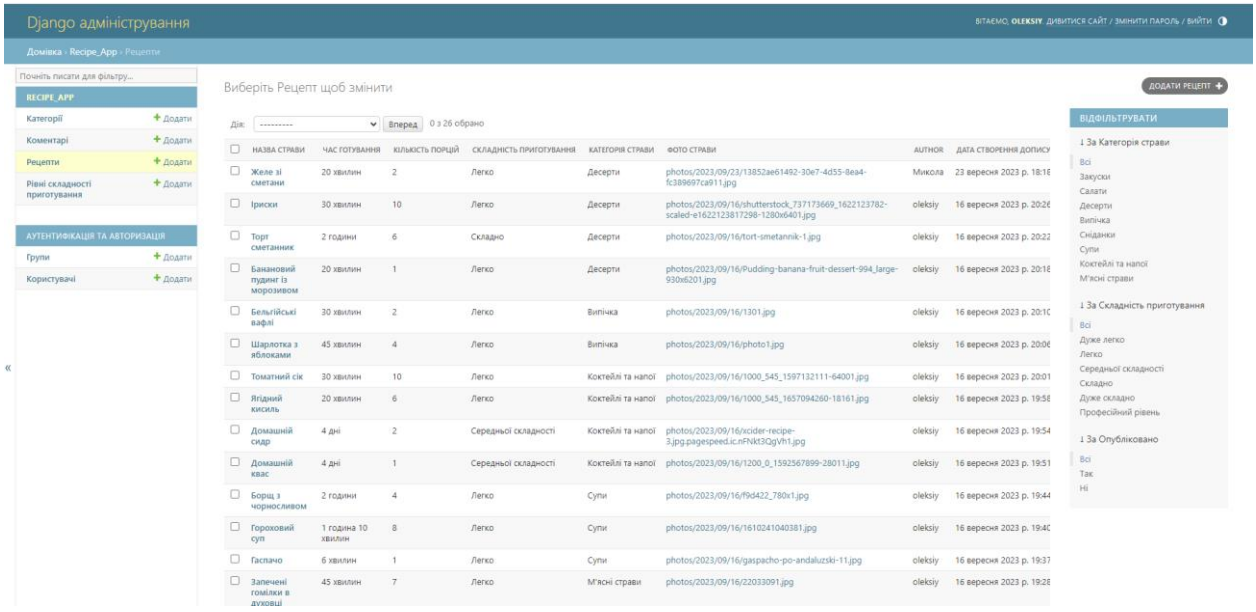
Желе зі сметани

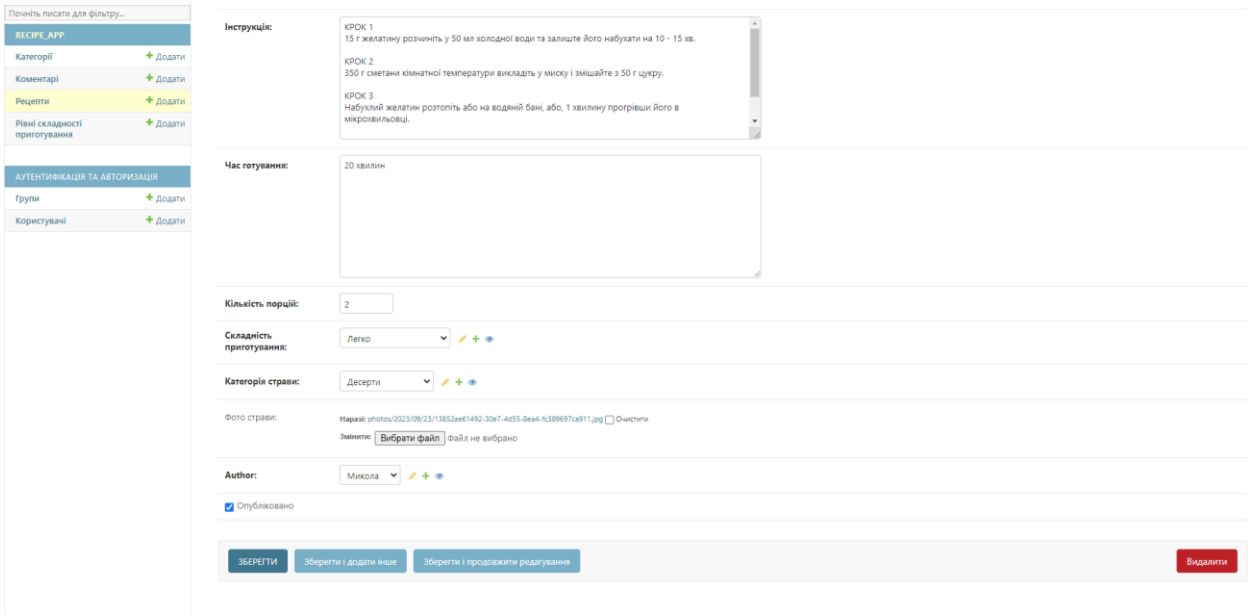
Скл-сть приготування: Легко
Час приготування: 20 хвилин
Автор: Микола
Рейтинг: 3,5 (2)

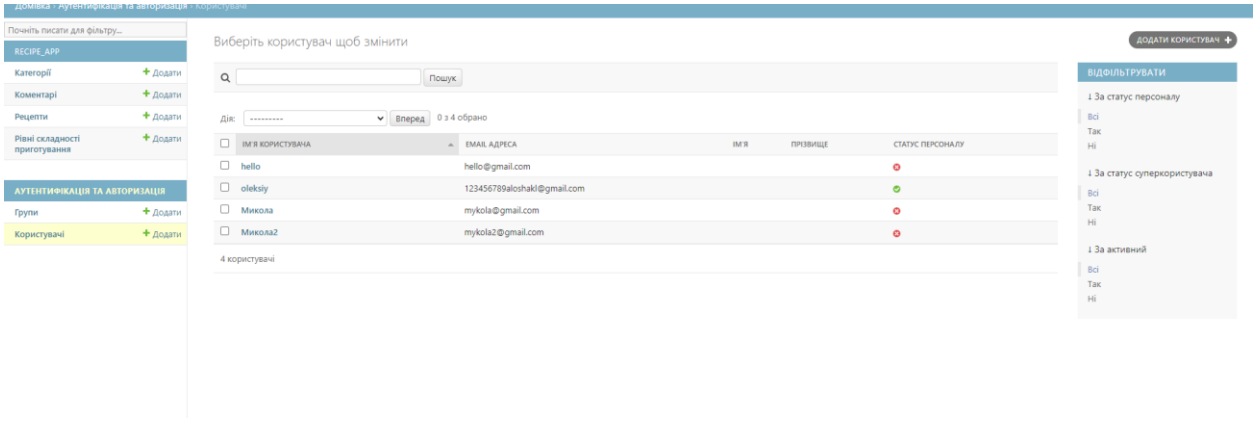

Головна
Всі рецепти
Запропонувати рецепт
Ласкаво просимо oleksiy |
Вийти
Запропонувати запис
Введіть назву страви
Введіть всі необхідні інгрідієнти
Напишіть інструкцію приготування
Напишіть скільки часу готується страва

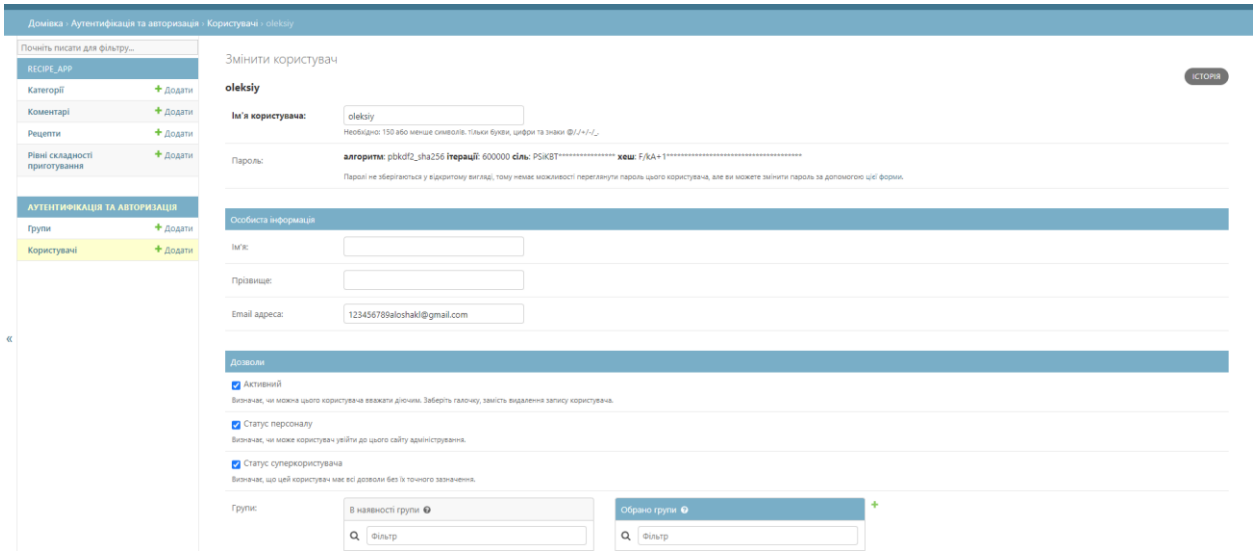
Додаток Б – Панель адміністратора



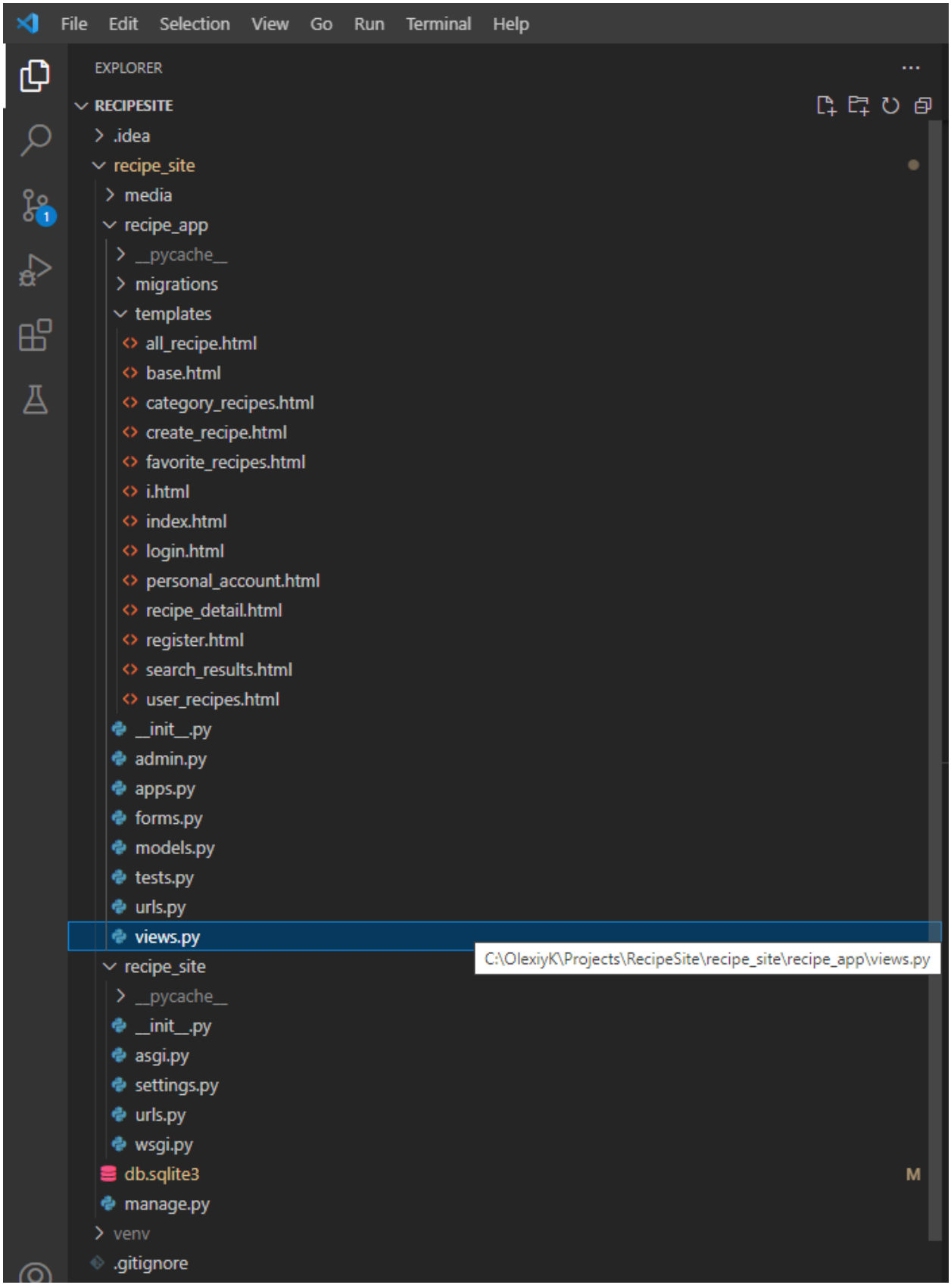



Почніть писати для фільтру...
RECIPE_APP
Категорії + Додати
Коментарі + Додати
Рецепти + Додати
Рівні складності приготування + Додати
АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ
Групи + Додати
Користувачі + Додати
Інструкція:
КРОК 1
15 г желатину розчиніть у 50 мл холодної води та залиште його набухати на 10 - 15 хв.
КРОК 2
350 г сметани кімнатної температури викладіть у миску і змішайте з 50 г цукру.
КРОК 3
Набухлий желатин розтопіть або на водяній бані, або, 1 хвилину прогрівши його в мікрохвильовці.
Час готування:
20 хвилин
Кількість порцій:
2
Складність приготування:
Легко
Категорія страви:
Десерти
Фото страви:
Наразі: photos/2023/09/23/13852ae61492-30e7-4d55-8ea4-fc389697ca911.jpg Очистити
Змінити: Вибрати файл Файл не вибрано
Author:
Микола
Опубліковано
ЗБЕРЕГТИ
Зберегти і додати інше
Зберегти і продовжити редагування
Видалити


Почніть писати для фільтру...
RECIPE_APP
Категорії + Додати
Коментарі + Додати
Рецепти + Додати
Рівні складності приготування + Додати
АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ
Групи + Додати
Користувачі + Додати
Виберіть користувач щоб змінити
ДОДАТИ КОРИСТУВАЧ +
Пошук
Дія: --------- Вперед 0 з 4 обрано
ІМ'Я КОРИСТУВАЧА
EMAIL АДРЕСА
ІМ'Я
ПРІЗВИЩЕ
СТАТУС ПЕРСОНАЛУ
hello hello@gmail.com
oleksiy 123456789aloshakl@gmail.com
Микола mykola@gmail.com
Микола2 mykola2@gmail.com
4 користувачі
ВІДФІЛЬТРУВАТИ
За статус персоналу
Всі
Так
Ні
За статус суперкористувача
Всі
Так
Ні
За активний
Всі
Так
Ні


Домівка › Аутентифікація та авторизація › Користувачі › oleksiy
Почніть писати для фільтру...
RECIPE_APP
Категорії + Додати
Коментарі + Додати
Рецепти + Додати
Рівні складності приготування + Додати
АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ
Групи + Додати
Користувачі + Додати
Змінити користувач
ІСТОРІЯ
oleksiy
Ім'я користувача: oleksiy
Необхідно: 150 або менше символів. Тільки букви, цифри та знаки @/./+/-/_.
Пароль: алгоритм: pbkdf2_sha256 ітерацій: 600000 сіль: PSIKBT********** хеш: F/kA+1**********
Паролі не зберігаються у відкритому вигляді, тому немає можливості переглянути пароль цього користувача, але ви можете змінити пароль за допомогою цієї форми.
Особиста інформація
Ім'я:
Прізвище:
Email адреса: 123456789aloshakl@gmail.com
Дозволи
Активний
Визначає, чи можна цього користувача вважати діючим. Зберіть галочку, замість видалення запису користувача.
Статус персоналу
Визначає, чи може користувач увійти до цього сайту адміністрування.
Статус суперкористувача
Визначає, що цей користувач має всі дозволи без їх точного зазначення.
Групи:
В наявності групи
Фільтр
Обрано групи
Фільтр

Додаток В – Загальна структура проекту



Додаток Г – Код моделі проекту

```
from django.db import models

from django.db import models
from django.contrib.auth.models import User
from django.contrib.auth.models import AbstractUser, Group, Permission
from django.utils.translation import gettext as _

# Модель для складності рецепту
class DifficultyLevel(models.Model):
    name = models.CharField('Рівень складності приготування', max_length=20)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Рівень складності приготування'
        verbose_name_plural = 'Рівні складності приготування'

# Модель для категорій рецептів
class Category(models.Model):
    name = models.CharField('Назва категорії',max_length=100)
    category_image = models.ImageField(upload_to='photos/%Y/%m/%d/',
verbose_name='Фото страви', blank=True, null=True,)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Категорія'
        verbose_name_plural = 'Категорії'

# Модель для рецептів
```

```
class Recipe(models.Model):
    title = models.CharField('Назва страви', max_length=200)
    ingredients = models.TextField('Інгрідієнти')
    instructions = models.TextField('Інструкція')
    preparation_time = models.TextField('Час готування')
    servings = models.PositiveIntegerField('Кількість порцій')
    difficulty_level = models.ForeignKey(DifficultyLevel, on_delete=models.CASCADE, verbose_name='Складність приготування')
    category = models.ForeignKey(Category, on_delete=models.CASCADE, verbose_name='Категорія страви')
    recipe_image = models.ImageField(upload_to='photos/%Y/%m/%d/', verbose_name='Фото страви', blank=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    date_created = models.DateTimeField('Дата створення допису', auto_now_add=True)
    publish = models.BooleanField('Опубліковано', default=False)

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = 'Рецепт'
        verbose_name_plural = 'Рецепти'
        ordering = ['-id']

# Модель для коментарів
class Comment(models.Model):
    text = models.TextField('Текст коментаря')
    author = models.ForeignKey(User, on_delete=models.CASCADE, verbose_name='Автор коментаря')
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE, verbose_name='Рецепт під яким залишили коментар')
    date_created = models.DateTimeField('Дата створення коментаря', auto_now_add=True)
```

```
        def __str__(self):
            return f'Comment by {self.author.username} on {self.recipe.title}'

        class Meta:
            verbose_name = 'Коментар'
            verbose_name_plural = 'Коментарі'

    # Модель для оцінок рецептів
    class Rating(models.Model):
        rating_value = models.PositiveIntegerField()
        recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
        user = models.ForeignKey(User, on_delete=models.CASCADE)

        def __str__(self):
            return f'{self.rating_value} stars for {self.recipe.title} by {self.user.username}'

        class Meta:
            verbose_name = 'Рейтинг'
            verbose_name_plural = 'Рейтинги'

    # Модель для обраних рецептів користувачів
    class FavoriteRecipe(models.Model):
        recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
        user = models.ForeignKey(User, on_delete=models.CASCADE)

        def __str__(self):
return f'{self.user.username} added {self.recipe.title} to favorites'
```

Додаток Д – Views проекту

```
from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse
from django.http import JsonResponse, HttpResponseBadRequest

from .models import *
from .forms import  *
from django.core.paginator import Paginator
from django.views import View
from django.views.generic import ListView, CreateView, FormView
from django.contrib.auth import login, logout
from django.contrib import messages
from django.views.decorators.http import require_POST
from django.contrib.auth.mixins import LoginRequiredMixin
from django.db.models import Sum
from django.urls import reverse_lazy
from django.http import HttpResponseRedirect
from django.db.models import Q


class HomeRecipe(ListView):
    model = Category
    template_name = 'index.html'
    context_object_name = 'categories'



class CategoryRecipesView(ListView):
    model = Recipe
    template_name = 'all_recipe.html'
    context_object_name = 'recipes'
    paginate_by = 10
```

```
        def get_queryset(self):
            category_id = self.kwargs['category_id']
            queryset = Recipe.objects.filter(category_id=category_id, publish=True)
            sorting = self.request.GET.get('sort_by', None)
            if sorting:
                queryset = sort_recipes(queryset, sorting)
            return queryset


        def get_context_data(self, **kwargs):
            context = super().get_context_data(**kwargs)
            recipe_list = context[self.context_object_name]
            add_ratings_to_recipes(recipe_list)
            context[self.context_object_name] = recipe_list
            stars = [1, 2, 3, 4, 5]
            context['stars'] = stars
            return context


    class RecipeSearchListView(ListView):
        model = Recipe
        template_name = 'search_results.html'
        context_object_name = 'recipes'
        paginate_by = 10

        def get_queryset(self):
            query = self.request.GET.get('q')
            if query:
                query_lower = query.lower()  # Переведення пошукового запиту в нижній регістр
                matching_recipes = Recipe.objects.filter(title__icontains=query_lower)
                if not matching_recipes:
                    matching_recipes = Recipe.objects.filter(title__icontains=query_lower.capitalize())
```

```
            if not matching_recipes:
                words = query_lower.split()
                for word in words:
                    matching_recipes |= Recipe.objects.filter(title__icontains=word)
                    if matching_recipes.count() >= 5:
                        break
            return matching_recipes
        return Recipe.objects.all()

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        recipe_list = context[self.context_object_name]
        add_ratings_to_recipes(recipe_list)
        context[self.context_object_name] = recipe_list
        stars = [1, 2, 3, 4, 5]
        context['stars'] = stars
        return context



class AllRecipe(ListView):
    model = Recipe
    template_name = 'all_recipe.html'
    context_object_name = 'recipes'
    paginate_by = 10

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        recipe_list = context[self.context_object_name]

        add_ratings_to_recipes(recipe_list)
        context[self.context_object_name] = recipe_list
        stars = [1, 2, 3, 4, 5]
        context['stars'] = stars
```

```
        return context

    def get_queryset(self):
        queryset = Recipe.objects.filter(publish=True)
        sorting = self.request.GET.get('sort_by', None)
        if sorting:
            queryset = sort_recipes(queryset, sorting)
        return queryset


def sort_recipes(queryset, sorting):
    if sorting == 'rating' or 'reverse_rating':
        recipe_ratings = [
            (recipe.id, *calculate_average_rating(recipe))
            for recipe in queryset
        ]
        # Сортуємо список рейтингів за середнім рейтингом
        if sorting == 'rating':
            recipe_ratings.sort(key=lambda x: x[1], reverse=True)
        elif sorting == 'reverse_rating':
            recipe_ratings.sort(key=lambda x: x[1], reverse=False)
        # Отримуємо список id рецептів відсортованих за рейтингом
        sorted_recipe_ids = [rating[0] for rating in recipe_ratings]
        # Змінюємо порядок рецептів в queryset, використовуючи відсортовані id
        queryset = sorted(queryset, key=lambda x: sorted_recipe_ids.index(x.id))

    if sorting == 'popular' or 'reverse_popular':
        recipe_ratings = [
            (recipe.id, Rating.objects.filter(recipe=recipe).count())
            for recipe in queryset
        ]
        # Сортуємо список рейтингів за загальною кількістю
        if sorting == 'popular':
            recipe_ratings.sort(key=lambda x: x[1], reverse=True)
```

```
            elif sorting == 'reverse_popular':
                recipe_ratings.sort(key=lambda x: x[1], reverse=False)
            # Отримуємо список id рецептів відсортованих за загальною кількістю рейтингів
            sorted_recipe_ids = [rating[0] for rating in recipe_ratings]
            # Змінюємо порядок рецептів в queryset, використовуючи відсортовані id
            queryset = sorted(queryset, key=lambda x: sorted_recipe_ids.index(x.id))

        if sorting == 'difficulty' or sorting == 'reverse_difficulty':
            recipe_ids = [recipe.id for recipe in queryset]
            q_objects = Q(id__in=recipe_ids)
            queryset = Recipe.objects.filter(q_objects)
            if sorting == 'difficulty':
                queryset = queryset.order_by('difficulty_level')
            elif sorting == 'reverse_difficulty':
                queryset = queryset.order_by('-difficulty_level')

        return queryset



    def register_view(request):
        if request.method == 'POST':
            form = RegisterForm(request.POST)
            if form.is_valid():
                user = form.save()
                login(request, user)
                messages.success(request, 'Ви успішно зареєструвались')
                return redirect('home')
            else:
                messages.error(request, 'Проблема під час реєстрації')
        else:
            form = RegisterForm()
        return render(request, 'register.html', {'form': form})
```

```
def login_view(request):
    if request.method == 'POST':
        form = LoginForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('home')
    else:
        form = LoginForm()
    return render(request, 'login.html', {'form': form})


def logout_view(request):
    logout(request)
    return redirect('home')


class RecipeDetailView(View):
    template_name = 'recipe_detail.html'

    def get(self, request, recipe_id):
        recipe = get_object_or_404(Recipe, pk=recipe_id)
        comments = recipe.comment_set.all().order_by('-date_created')
        form = CommentForm()
        rating_form = RatingForm()

        is_favorite = False  # Початково рецепт не улюблений

        # Логіка для обчислення середнього рейтингу
        average_rating, total_reviews, rating_residue = calculate_average_rating(recipe)

        if request.user.is_authenticated:
            if FavoriteRecipe.objects.filter(recipe=recipe, user=request.user).exists():
```

```
                is_favorite = True

            return render(request, self.template_name, {
                'recipe': recipe,
                'comments': comments,
                'form': form,
                'rating_form': rating_form,  # Додано форму оцінки
                'is_favorite': is_favorite,
                'average_rating': average_rating,
                'total_reviews': total_reviews,
                'rating_residue': rating_residue,
                'stars': [1,2,3,4,5],
            })

        def post(self, request, recipe_id):
            recipe = get_object_or_404(Recipe, pk=recipe_id)
            comments = recipe.comment_set.all()
            form = CommentForm(request.POST)
            rating_form = RatingForm(request.POST)

            if form.is_valid():
                comment = form.save(commit=False)
                comment.recipe = recipe
                comment.author = request.user
                comment.save()
                return redirect('recipe_detail', recipe_id=recipe_id)

            if rating_form.is_valid():
                rating = rating_form.cleaned_data['rating']

                existing_rating, created = Rating.objects.get_or_create(recipe=recipe,
user=request.user, defaults={'rating_value': rating})

                if not created:
```

```
                existing_rating.rating_value = rating
                existing_rating.save()

            total_reviews = Rating.objects.filter(recipe=recipe).count()
            total_rating = Rating.objects.filter(recipe=recipe).aggregate(Sum('rating_value'))['rating_value__sum']
            average_rating = round(total_rating / total_reviews, 2) if total_reviews > 0 else 0
            rating_residue = average_rating % 1

            # Повертаємо JsonResponse замість redirect
            return JsonResponse({'average_rating': average_rating, 'total_reviews': total_reviews, 'rating_residue': rating_residue})

        return render(request, self.template_name, {
            'recipe': recipe,
            'comments': comments,
            'form': form,
            'rating_form': rating_form,
        })


    def calculate_average_rating(recipe):
        reviews = Rating.objects.filter(recipe=recipe)
        total_reviews = reviews.count()
        if total_reviews > 0:
            total_rating = sum(review.rating_value for review in reviews)
            average_rating = round(total_rating / total_reviews, 2)
            rating_residue = average_rating % 1
        else:
            average_rating = 0
            rating_residue = 0
        return average_rating, total_reviews, rating_residue
```

```
        def add_ratings_to_recipes(recipes):
            for recipe in recipes:
                average_rating, total_reviews, rating_residue = calculate_average_rating(recipe)
                recipe.average_rating = average_rating
                recipe.total_reviews = total_reviews
                recipe.rating_residue = rating_residue



        def rate_recipe(request, recipe_id):
            if request.method == 'POST' and request.headers.get('x-requested-with') == 'XMLHttpRequest':
                rating = int(request.POST.get('rating', 0))

                if 1 <= rating <= 5:
                    recipe = get_object_or_404(Recipe, pk=recipe_id)
                    user = request.user

                    existing_rating, created = Rating.objects.get_or_create(recipe=recipe, user=user, defaults={'rating_value': rating})

                    if not created:
                        existing_rating.rating_value = rating
                        existing_rating.save()

                    total_reviews = Rating.objects.filter(recipe=recipe).count()
                    total_rating = Rating.objects.filter(recipe=recipe).aggregate(Sum('rating_value'))['rating_value__sum']
                    average_rating = round(total_rating / total_reviews, 2) if total_reviews > 0 else 0
                    rating_residue = average_rating % 1

                    return JsonResponse({'average_rating': average_rating, 'total_reviews': total_reviews, 'rating_residue': rating_residue})
                else:
                    return JsonResponse({'error': 'Invalid rating value'})
```

```
        else:
            return JsonResponse({'error': 'Invalid request'})


    def toggle_favorite(request, recipe_id):
        if request.user.is_authenticated:
            recipe = Recipe.objects.get(pk=recipe_id)
            user = request.user
            favorite, created = FavoriteRecipe.objects.get_or_create(user=user, recipe=recipe)
            if created:
                message = "Рецепт додано до улюблених."
            else:
                favorite.delete()
                message = "Рецепт видалено з улюблених."
            return JsonResponse({'message': message})
        else:
            return JsonResponse({'message': 'Потрібно увійти в систему, щоб додавати рецепти до улюблених.'})


    def create_recipe(request):
        if request.method == 'POST':
            form = RecipeForm(request.POST, request.FILES)
            if form.is_valid():
                recipe = form.save(commit=False)
                recipe.author = request.user
                recipe.save()
                return redirect('home')
        else:
            form = RecipeForm()

        return render(request, 'create_recipe.html', {'form': form})
```

```
class FavoriteRecipesView(View):
    template_name = 'favorite_recipes.html'

    def get(self, request):
        if request.user.is_authenticated:
            user = request.user
            favorite_recipes = FavoriteRecipe.objects.filter(user=user)
            # Отримання списку рецептів, які є улюбленими для даного користувача
            recipes = [fav.recipe for fav in favorite_recipes]
            add_ratings_to_recipes(recipes)
            return render(request, self.template_name, {'favorite_recipes': favorite_recipes, 'recipes': recipes, 'stars': [1,2,3,4,5]})
        else:
            return render(request, 'not_authenticated.html')



class UserRecipeListView(LoginRequiredMixin, ListView):
    model = Recipe
    template_name = 'user_recipes.html'
    context_object_name = 'recipes'
    paginate_by = 10

    def get_queryset(self):
        recipes = Recipe.objects.filter(author=self.request.user)
        add_ratings_to_recipes(recipes)
    return recipes
```

Додаток Е – код форм

```
from django import forms
from .models import *
from django.forms import ModelForm, TextInput, Textarea, Select, ClearableFileInput, EmailInput, NumberInput
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User
# from captcha.fields import CaptchaField


class LoginForm(AuthenticationForm):
    username = forms.CharField(label='', widget=(TextInput({"class": "form-control", "placeholder": "Введіть нік користувача"})))
    password = forms.CharField(label='', widget=(TextInput({"class": "form-control", "placeholder": "Введіть пароль"})))


class RegisterForm(UserCreationForm):
    username = forms.CharField(label='', widget=(TextInput({"class": "form-control", "placeholder": "Введіть нік користувача"})))
    email = forms.EmailField(label='', widget=(EmailInput({"class": "form-control", "placeholder": "Введіть ваш email"})))
    password1 = forms.CharField(label='', widget=(TextInput({"class": "form-control", "placeholder": "Введіть пароль"})))
    password2 = forms.CharField(label='', widget=(TextInput({"class": "form-control", "placeholder": "Повторіть пароль"})))

    def clean_username(self):
        username = self.cleaned_data['username']
        if User.objects.filter(username=username).exists():
            raise forms.ValidationError("Цей нікнейм вже використовується. Будь ласка, виберіть інший.")
        return username
```

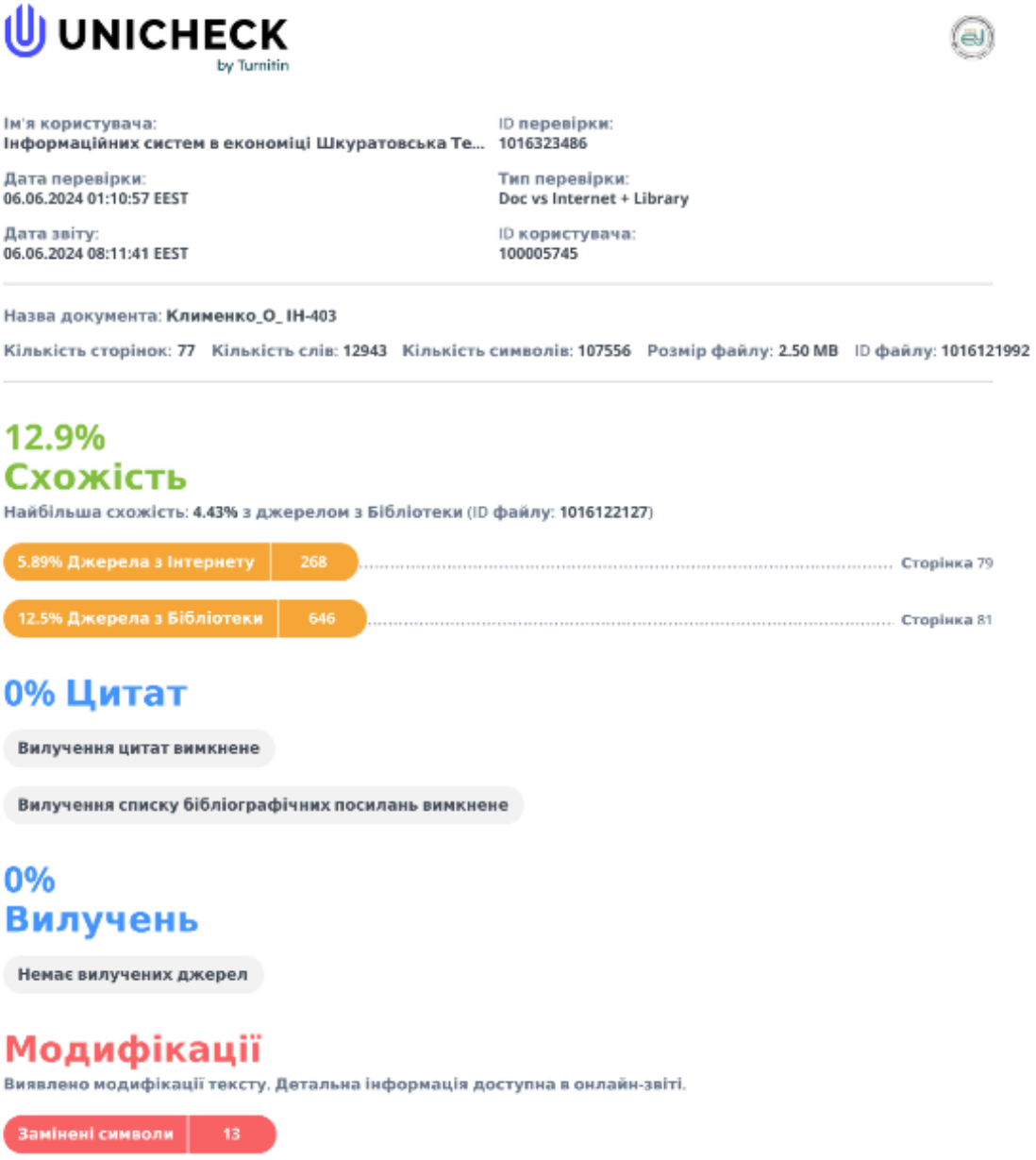
```
        def clean_email(self):
            email = self.cleaned_data['email']
            if User.objects.filter(email=email).exists():
                raise forms.ValidationError("Ця електронна пошта вже використовується. Будь ласка, введіть іншу.")
            return email

        class Meta:
            model = User
            fields = ['username', 'email', 'password1', 'password2']


    class RecipeForm(ModelForm):
        class Meta:
            model = Recipe
            fields = ['title', 'ingredients', 'instructions', 'instructions', 'preparation_time', 'servings', 'difficulty_level', 'category', 'recipe_image']
            widgets = {
                'title': TextInput({
                    "class": "form-control",
                    "placeholder": "Введіть наву страви"
                }),
                'ingredients': Textarea({
                    "class": "form-control",
                    "placeholder": "Введіть всі необхідні інгрідієнти"
                }),
                'instructions': Textarea({
                    "class": "form-control",
                    "placeholder": "Напишіть інструкцію приготування"
                }),
                'preparation_time': TextInput({
                    "class": "form-control",
                    "placeholder": "Напишіть скільки часу готується страва"
```

```
            }),
            'servings': NumberInput({
                "class": "form-control",
                "placeholder": "Напишіть скільки вийде порцій"
            }),
            'difficulty_level': Select({
                "class": "form-control"
            }),
            'category': Select({
                "class": "form-control"
            }),
            'recipe_image': ClearableFileInput({
                "class": "form-control",
            }),
        }

class CommentForm(forms.ModelForm):
    class Meta:
        model = Comment
        fields = ['text']



class RatingForm(forms.ModelForm):
    rating = forms.IntegerField(min_value=1, max_value=5, required=False)

    class Meta:
        model = Rating
        fields = ['rating']
```

Додаток Є – налаштування панелі адміністратора

```
from django.contrib import admin
from .models import *


class RecipeAdmin(admin.ModelAdmin):
    list_display = ['title', 'preparation_time', 'servings', 'difficulty_level', 'category', 'recipe_image', 'author', 'date_created', 'publish']
    list_editable = ['publish']
    list_filter = ['category', 'difficulty_level', 'publish']


class DifficultyLevelAdmin(admin.ModelAdmin):
    list_display = ['name']


class CategoryAdmin(admin.ModelAdmin):
    list_display = ['name']


class CommentAdmin(admin.ModelAdmin):
    list_display = ['recipe', 'author', 'text', 'date_created']


admin.site.register(Recipe, RecipeAdmin)
admin.site.register(DifficultyLevel, DifficultyLevelAdmin)
admin.site.register(Category, CategoryAdmin)
admin.site.register(Comment, CommentAdmin)
```

Додаток Ж – Результат Unicheck



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЕКТ

на тему

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ОБМІНУ КУЛІНАРНИМИ РЕЦЕПТАМИ ТА ПОРАДАМИ

здобувача Клименка Олексія Володимировича

Науковий керівник:
к.е.н., доцент
Гордієнко І.В.

Проект допущений до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент
Тішков Б.О.

Київ 2024