МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
_____
(підпис)
грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Впровадження штучного інтелекту для оптимізації технологічних процесів у кулінарній індустрії та покращення якості страв»
здобувачки групи ІН.мз – 31с Войтенко Каріна Костянтинівна

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Каріна ВОЙТЕНКО
(підпис)

Керівник Валентина БОРОВИК
доцент, кандидат технічних наук, -
доцент

_____
(підпис)

Суми – 2024

Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.мз-31с Войтенко Каріна Костянтинівна

1. Тема роботи: «Впровадження штучного інтелекту для оптимізації технологічних процесів у кулінарній індустрії та покращення якості страв»
затверджую наказом по СумДУ від *«05» грудня 2024 року № 1267-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 06 грудня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються з налаштування мовної моделі 3) Розробка інтелектуальної системи для оптимізації технологічних процесів у кулінарній індустрії 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	07.09.2024	
2	*Огляд технологій, що використовуються для налаштування мовної моделі*	28.09.2024	
3	*Розробка інтелектуальної системи для оптимізації технологічних процесів у кулінарній індустрії*	10.10.2024	
4	*Аналіз отриманих результатів*	15.11.2024	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	20.11.2024	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 59 стор., 20 рис., 1 додатоку, 23 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі підвищення ефективності кулінарної галузі через впровадження штучного інтелекту. Використання технологій штучного інтелекту дозволяє автоматизувати процеси створення рецептів, аналізу інгредієнтів, персоналізації меню та управління взаємодією з клієнтами, що сприяє підвищенню якості обслуговування.

Об’єкт дослідження – процес використання штучного інтелекту для автоматизації кулінарних процесів.

Мета роботи – розробка кулінарної платформи з використанням мовної моделі LLaMA, яка дозволить автоматизувати створення рецептів, персоналізувати обслуговування та покращити управління ресурсами.

Методи дослідження – алгоритми машинного навчання, методи обробки природної мови, інструменти для роботи з великими даними та інтеграція з CRM-системою.

Результати – розроблено кулінарну інформаційну технологію, що інтегрує мовну модель LLaMA для автоматизації процесів створення рецептів, аналізу інгредієнтів, перевірки дієтичних обмежень, та надання персоналізованих рекомендацій користувачам. Платформа дозволяє закладам громадського харчування підвищити ефективність роботи, покращити взаємодію з клієнтами та забезпечити відповідність страв індивідуальним потребам споживачів. Проведено тестування системи на основі реальних даних.

ШТУЧНИЙ ІНТЕЛЕКТ, КУЛІНАРНА ПЛАТФОРМА, LLaMA, PYTHON, MONGODB, MERN, CRM, МАШИННЕ НАВЧАННЯ.

ЗМІСТ

ВСТУП

Актуальність. Сьогодні питання дослідження та використання штучного інтелекту у кулінарній галузі є надзвичайно актуальним. Використання технологій штучного інтелекту для автоматизації процесів у кулінарії може значно підвищити якість обслуговування, ефективність управління ресурсами та персоналізацію споживчого досвіду. Такі інновації дозволяють знизити витрати, підвищити швидкість обробки запитів та забезпечити відповідність страв дієтичним обмеженням і вподобанням клієнтів. Розробка інтерактивної платформи на основі штучного інтелекту, яка дозволяє генерувати рецепти, аналізувати інгредієнти на наявність алергенів та створювати персоналізовані меню, матиме значний попит серед закладів громадського харчування, власників бізнесу, а також індивідуальних користувачів, які прагнуть покращити свій раціон.

Об’єкт дослідження. Процес використання штучного інтелекту для автоматизації кулінарних процесів.

Предмет дослідження. Методологія інтеграції мовної моделі LLaMA у платформу для кулінарії з метою автоматизації процесів та персоналізації обслуговування.

Гіпотеза. Використання штучного інтелекту та інтеграція мовної моделі LLaMA у кулінарну платформу дозволить автоматизувати створення рецептів, аналіз інгредієнтів та персоналізацію дієтичних рекомендацій, що забезпечить покращення якості обслуговування та підвищення ефективності кулінарних закладів.

Наукова новизна. На відміну від існуючих аналогів кулінарних платформ, розроблена система інтеграції мовної моделі LLaMA забезпечує автоматичне створення рецептів з урахуванням дієтичних обмежень, аналіз наявності алергенів та можливість персоналізації меню. Такий підхід дозволить користувачам отримувати більш точні рекомендації, а також автоматизувати процеси формування меню та управління запасами продуктів.

Структура. Дана робота складається зі вступу, аналізу публікацій, постановки задачі дослідження, вибору методології та інструментів для

вирішення поставленої проблеми, опису розробленого програмного забезпечення кулінарної платформи, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРОБЛЕМИ І ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд сучасних тенденцій у кулінарній індустрії: основні виклики та проблеми

На сучасному етапі розвитку кулінарна індустрія зазнає суттєвих змін як під впливом так і технологічного прогресу так і зміни очікувань споживачів. Основними трендами в цій сфері є активне впровадження інновацій персоналізація обслуговування, підвищений інтерес до здорового харчування, та екологічності. Водночас існують і серйозні виклики такі як зростання витрат на ресурси дефіцит кваліфікованого персоналу та необхідність забезпечення безпеки і високої якості харчових продуктів. Дослідження цих трендів і проблем є важливим для розуміння того що штучний інтелект (ШІ) може сприяти подоланню існуючих бар'єрів та забезпечити конкурентні переваги для учасників ринку[1].

Однією з ключових тенденцій яка стрімко набирає популярність, є впровадження технологічних інновацій .Перш за все це пов'язано з автоматизацією кулінарних процесів, що включає використання роботизованих кухонь автоматичних систем для приготування їжі та чат-ботів для прийому замовлень.

Рисунок 1.1 – Зростання ринку автоматизації [2]

На діаграмі, яка ілюструє зростання ринку автоматизації в кулінарній індустрії з 2018 по 2023 рік. Як видно з графіка, розмір ринку демонструє стійке зростання, що підкреслює тенденцію до активного впровадження автоматизованих технологій у сфері громадського харчування.

Такі інновації дають можливість оптимізувати виробничий процес, зменшити витрати на обслуговуючий персонал та підвищити загальну ефективність роботи закладів громадського харчування. Наприклад, роботизовані кухні здатні одночасно готувати кілька страв, значно скорочуючи час очікування замовлення для клієнтів. У свою чергу, чат-боти можуть обробляти замовлення в автоматичному режимі, що знижує навантаження на персонал і підвищує рівень обслуговування.

Сучасні технології також активно використовуються для управління замовленнями продуктів, моніторингу запасів і контролю витрат на сировину. Дедалі більше ресторанів впроваджують програми для моніторингу запасів, які використовують алгоритми ШІ для прогнозування необхідних закупівель на основі історичних даних [2]. Це сприяє зменшенню харчових відходів, оскільки система допомагає уникати надмірних закупівель продуктів, що швидко псуються.

Сучасні споживачі очікують від ресторанів не лише якісних страв, але й унікального персонального підходу, який враховує їхні індивідуальні смаки, алергії або дієтичні обмеження. Тренд на персоналізацію стає дедалі популярнішим у кулінарній індустрії, що мотивує заклади громадського харчування активно впроваджувати системи штучного інтелекту. Алгоритми ШІ, здатні аналізувати переваги клієнтів та формувати персональні меню, стали основним інструментом для підвищення рівня обслуговування [3].

На основі попередніх замовлень та історії покупок клієнтів, системи ШІ можуть рекомендувати їм страви, які відповідають їхнім смакам та потребам. Це підвищує рівень задоволеності споживачів та дозволяє ресторанам ефективніше використовувати свої ресурси.

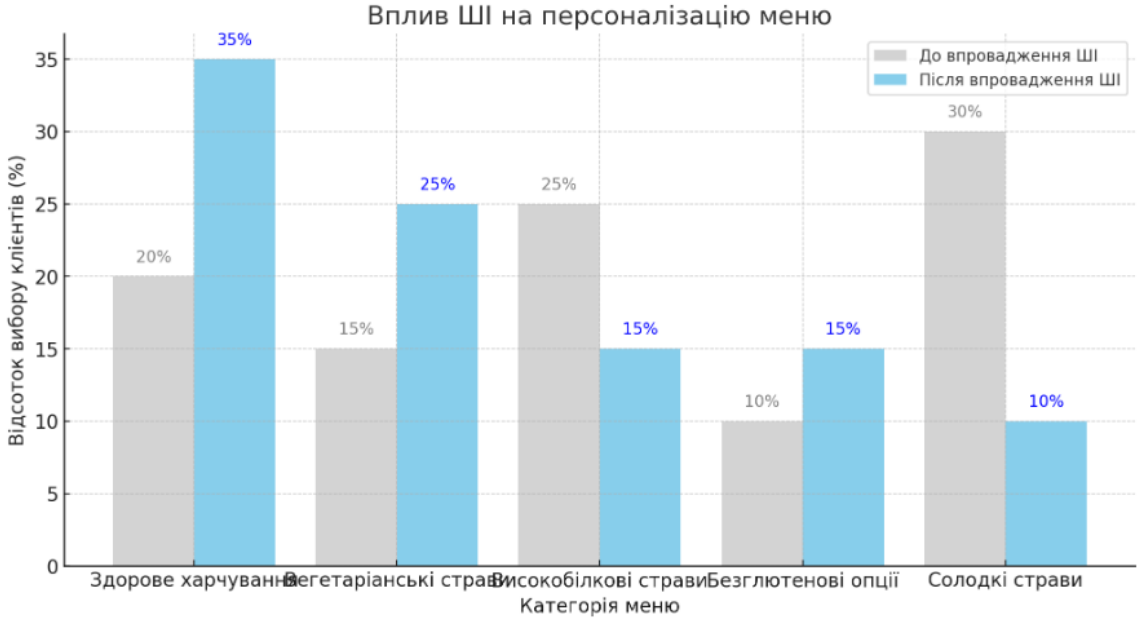

Рисунок 1.2 – Вплив ШІ на персоналізацію меню [4]

На цьому графіку зображено, як ШІ допомагає в персоналізації меню, орієнтуючись на вподобання клієнтів. Після впровадження ШІ зріс вибір категорій, пов'язаних із здоровим харчуванням та вегетаріанськими стравами, оскільки ШІ аналізує дані про вподобання і пропонує відповідні опції, що підвищує задоволеність клієнтів і адаптує меню до їхніх потреб.

Такий індивідуальний підхід не лише підвищує лояльність клієнтів, а й сприяє зростанню повторних продажів, оскільки споживачі надають перевагу тим закладам, які відповідають їхнім вимогам і пропонують унікальний досвід [5].

Попит на здорове харчування та екологічно чисті продукти стрімко зростає, і це стає важливим чинником для кулінарної індустрії. Споживачі, зокрема молодь, надають перевагу стравам з органічних інгредієнтів, які не містять штучних добавок. Крім того, зростає інтерес до концепції **"нуль відходів"**, яка передбачає мінімізацію харчових відходів у процесі приготування їжі. Впровадження ШІ у цьому контексті може забезпечити точний розрахунок обсягів інгредієнтів для страв, що допоможе зменшити кількість харчових відходів.

Також інноваційні рішення допомагають контролювати якість продуктів на різних етапах їх обробки, зберігання та доставки.. Штучний інтелект у

поєднанні з системами моніторингу дозволяє відстежувати стан інгредієнтів у режимі реального часу та визначати, коли вони стають непридатними для використання. Це важливий крок до підвищення безпеки харчових продуктів та зниження ризиків, пов'язаних із недоброякісними стравами.

Крім позитивних змін, кулінарна індустрія також стикається з низкою викликів, які стають дедалі актуальнішими. Одним з основних викликів є постійне зростання витрат на ресурси, що включає як сировину, так і оплату праці. Це змушує підприємства шукати способи зменшення витрат, і штучний інтелект може стати ефективним рішенням у цьому напрямку. Використання автоматизованих систем дозволяє знизити витрати на обслуговування, оптимізувати управління запасами та запобігти марнотратству.

Іншим суттєвим викликом є дефіцит кваліфікованого персоналу. У зв'язку з цим дедалі більше ресторанів починають впроваджувати роботизовані системи для виконання стандартних операцій, таких як приготування певних страв, обслуговування клієнтів, а також прибирання. Така автоматизація дозволяє закладам громадського харчування підвищити продуктивність і зменшити залежність від людських ресурсів.

Третім значним викликом є забезпечення високої якості та безпеки харчових продуктів. Це особливо актуально в умовах пандемії та зростаючих вимог до гігієни. Споживачі очікують, що ресторани надаватимуть продукти високої якості, безпечні для здоров'я. ШІ допомагає здійснювати контроль на різних етапах підготовки продуктів, від закупівлі інгредієнтів до подачі страв. Наприклад, спеціальні датчики можуть виявляти відхилення у свіжості або якості продуктів та попереджати персонал про необхідність заміни інгредієнтів.

Штучний інтелект вже довів свою ефективність у багатьох сферах, і кулінарна індустрія - не виняток. Системи ШІ використовуються для прогнозування попиту на певні страви, оптимізації процесів закупівлі, автоматизації приготування їжі та підвищення якості обслуговування [6].

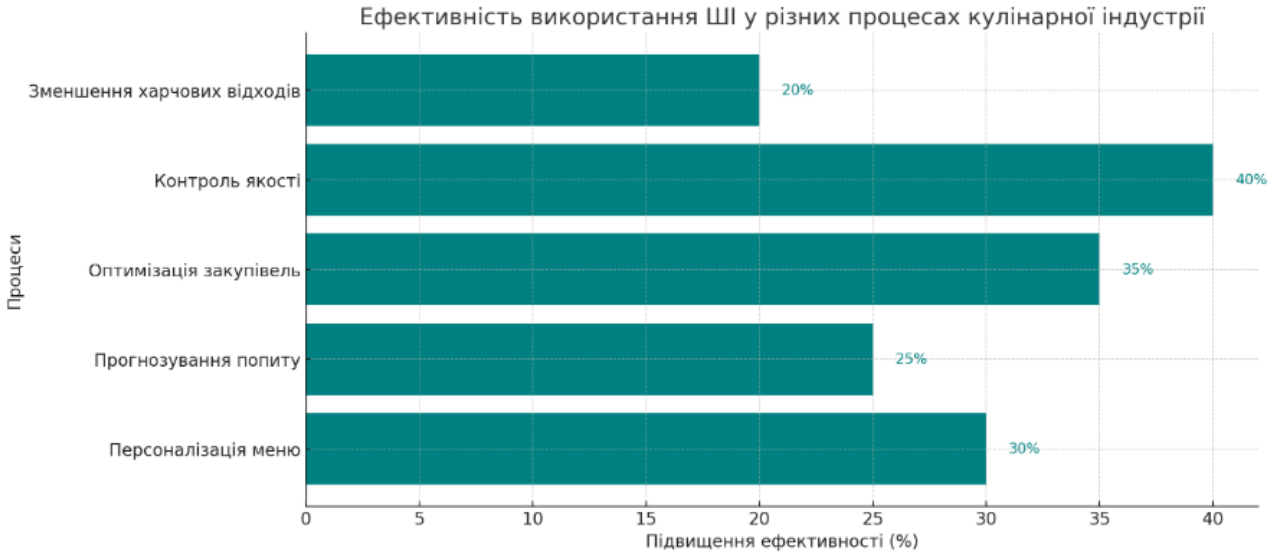

Рисунок 1.3 – Ефективність використання ШІ [7]

На цьому графіку зображено ефективність використання ШІ в різних процесах кулінарної індустрії. Найбільше підвищення ефективності досягається в контролі якості та оптимізації закупівель, де ШІ забезпечує стабільність і знижує ризик помилок. Персоналізація меню та прогнозування попиту також мають значний вплив, покращуючи адаптацію до потреб клієнтів і знижуючи витрати. Зменшення харчових відходів, хоч і має менший показник, все ж є важливим аспектом екологічності та економії.

1.2 Стан ринку кулінарії: аналіз попиту, пропозиції, конкурентного середовища

Аналіз сучасного ринку кулінарних послуг є важливою складовою для розуміння перспектив і можливостей впровадження інноваційних технологій, зокрема штучного інтелекту (ШІ), який здатний оптимізувати процеси та підвищити ефективність [8]. Попит на кулінарні послуги не тільки зростає, але й змінюється у своїй структурі, що пов’язано зі змінами в споживчих перевагах, демографічними тенденціями та високою конкуренцією на ринку. У сучасних умовах кулінарна індустрія характеризується складною системою взаємовідносин між попитом і пропозицією, високою конкуренцією, а також необхідністю швидко реагувати на зміни та адаптуватися до нових умов. Цей підпункт присвячений детальному аналізу попиту, структури пропозиції та

конкурентного середовища на ринку кулінарних послуг.

Поточні споживчі тенденції у сфері кулінарії показують, що попит на кулінарні послуги стає дедалі більш диференційованим і залежить від різноманітних факторів, таких як здорове харчування, персоналізація обслуговування, екологічність продуктів тощо. У сучасних умовах спостерігається зростання інтересу до страв, які відповідають концепціям здорового способу життя. Особливо це стосується продуктів, що не містять глютену, лактози, а також органічних страв. Це обумовлюється підвищеною увагою до власного здоров'я з боку молодих споживачів, які віддають перевагу екологічно чистим та якісним продуктам [9].

Ще однією важливою особливістю сучасного попиту є запит на персоналізацію обслуговування. Сучасні споживачі, особливо молоде покоління, очікують, що кулінарні заклади враховуватимуть їхні індивідуальні побажання та потреби. Це включає не лише вибір інгредієнтів для страв, але й різні способи їх приготування, що відповідають смакам та уподобанням клієнтів. Крім того, стрімке поширення онлайн-сервісів для замовлення та доставки їжі посилило попит на швидкість і зручність обслуговування, адже споживачі дедалі частіше обирають заклади, які пропонують можливість швидкого отримання замовлень, зокрема за допомогою цифрових платформ [10].

Пропозиція на ринку кулінарних послуг є різноманітною та включає кілька основних сегментів, таких як ресторани, кафе, фаст-фуд заклади, кейтерингові компанії, кулінарні онлайн-сервіси тощо. Кожен із цих сегментів має свою цільову аудиторію, особливості та вимоги, а також різні можливості для впровадження технологічних інновацій. Ресторани преміум-класу, наприклад, приділяють більше уваги якості продуктів та обслуговуванню, а також можуть використовувати ШІ для підвищення рівня персоналізації, в той час як фаст-фуд заклади орієнтовані на швидкість обслуговування та стандартизацію процесів, що також можна оптимізувати завдяки автоматизації та роботизації.

Щодо основних типів послуг, які пропонуються на ринку, варто зазначити, що вони включають традиційні страви, органічну їжу, дієтичні та веганські страви, а також різні види доставки. Пандемія COVID-19 значно вплинула на

структуру ринку, зокрема, підвищивши популярність сервісів доставки їжі та безконтактного обслуговування. Цей тренд залишився актуальним і після пандемії, що підштовхує заклади харчування до впровадження автоматизованих процесів, які дозволяють ефективно обслуговувати клієнтів із мінімальними затратами часу [11].

Ринок кулінарних послуг є висококонкурентним, і на ньому діють як великі мережі ресторанів, так і малі незалежні заклади. Основними гравцями на ринку є міжнародні та локальні мережі ресторанів, що мають значні ресурси для впровадження інновацій, а також дрібні гравці, які намагаються утримати свою частку за рахунок унікальних пропозицій або особистого підходу до клієнтів. Успішні кейси застосування ШІ серед великих гравців свідчать про його ефективність, що змушує малі заклади також розглядати можливість впровадження цих технологій, аби залишатися конкурентоспроможними.

Впровадження ШІ та інших інноваційних рішень стає ключовим фактором конкурентоспроможності. Деякі компанії вже активно використовують ШІ для персоналізації меню, аналізу переваг клієнтів та автоматизації процесів обслуговування. Наприклад, система на основі ШІ може автоматично аналізувати історію замовлень клієнта і пропонувати страви, що відповідають його попереднім виборам. Це підвищує рівень задоволеності клієнтів і сприяє повторним продажам. Конкурентні переваги, які отримують компанії завдяки ШІ, створюють нові виклики для інших учасників ринку, оскільки без впровадження інноваційних рішень багато закладів можуть втратити своїх клієнтів [12].

Для нових компаній, які бажають вийти на ринок кулінарних послуг, існують певні бар’єри. Основними з них є високі початкові витрати на відкриття закладу, складність у залученні клієнтів у конкурентному середовищі та необхідність підтримки високих стандартів якості. Додатково, витрати на впровадження нових технологій, таких як штучний інтелект, також можуть бути значними. ШІ може допомогти знизити операційні витрати за рахунок автоматизації процесів і підвищення ефективності обслуговування. Наприклад, використання систем на основі ШІ для управління запасами дозволяє

оптимізувати процес закупівлі продуктів, що знижує витрати та запобігає марнотратству.

Нестача кваліфікованого персоналу також є важливим викликом для нових учасників ринку, особливо в умовах сучасних демографічних змін. Деякі заклади змушені використовувати роботизовані системи для виконання стандартних операцій, таких як приготування страв або обслуговування клієнтів. Такі автоматизовані рішення дозволяють знизити витрати на персонал і підвищити продуктивність закладу, що є суттєвою перевагою для нових гравців, які намагаються конкурувати з великими мережами [13].

Загальний аналіз ринку кулінарних послуг показує, що він є динамічним, з високим рівнем конкуренції та різноманітними споживчими запитами. Штучний інтелект має значний потенціал для підвищення ефективності в цій галузі, адже він дозволяє оптимізувати управління ресурсами, забезпечити персоналізований підхід до клієнтів і підвищити якість обслуговування. Основними сферами для застосування ШІ є автоматизація процесів на кухні, управління запасами та моніторинг якості продуктів. Упровадження таких рішень дозволить не лише задовольнити потреби сучасних споживачів, але й зміцнити позиції на конкурентному ринку, підвищуючи лояльність клієнтів та забезпечуючи високу якість послуг.

1.3 Аналіз аналогічних проєктів

У процесі розробки проєкту було знайдено кілька аналогів, які мають схожий функціонал і спрямовані на вирішення задач у кулінарній індустрії. Серед них можна виділити платформи Yummly, Mealime, Whisk та MyFitnessPal. Кожна з них пропонує унікальні можливості, такі як персоналізовані рекомендації рецептів, планування меню, моніторинг калорійності або створення кулінарних книг. Ці сервіси використовуються для спрощення кулінарних процесів та забезпечення зручності для користувачів. Однак жодна з платформ повністю не відповідає концепції інтеграції мовної моделі та багаторівневого доступу, які є основними особливостями розроблюваного проєкту.

Yummly — це платформа, створена для персоналізованого підбору рецептів, яка використовує алгоритми машинного навчання для врахування індивідуальних уподобань. Система аналізує інгредієнти, алергени та дієтичні обмеження, щоб надавати користувачам рекомендації, які максимально відповідають їхнім потребам. Однією з переваг платформи є можливість інтеграції зі смарт-пристроями, що дозволяє автоматизувати процес приготування страв. Крім того, Yummly забезпечує простий у використанні інтерфейс, який дозволяє шукати рецепти за інгредієнтами, категоріями або навіть способами приготування. Однак платформа має обмежений набір функцій, орієнтованих на аналіз дієт чи калорійності, а також не пропонує багаторівневого доступу чи спеціального захисту від шкідливої інформації.

Mealime - це програма, орієнтована на планування меню та оптимізацію покупок. Вона пропонує користувачам можливість швидкого створення персоналізованого тижневого меню, що враховує дієтичні обмеження та алергії. Її головною перевагою є простота використання та інтеграція з функцією створення списків покупок, які можна синхронізувати із супермаркетами. Mealime також надає рецепти з коротким часом приготування, що робить платформу популярною серед зайнятих людей. Проте система обмежена функціями аналізу складних дієт або розширеного моніторингу калорійності. Інтеграція з мовними моделями чи можливість інтерактивного введення інформації про кулінарію відсутня.

Whisk - це платформа, яка дозволяє зберігати, створювати та обмінюватися рецептами. Її особливістю є можливість користувачів додавати рецепти з різних джерел та створювати персоналізовані кулінарні книги. Whisk також автоматично генерує списки покупок на основі обраних рецептів, що зручно для планування покупок. Однак система не має спеціальних функцій, спрямованих на аналіз алергенів чи створення дієт. Вона зосереджена на зборі рецептів і спрощенні процесу приготування їжі, що робить її більш обмеженою в порівнянні з багатофункціональними платформами. Крім того, захист від шкідливої інформації чи багаторівневий доступ у Whisk не передбачено.

MyFitnessPal - це популярний інструмент для моніторингу калорійності та

аналізу раціону, орієнтований на здорове харчування. Платформа дозволяє користувачам відстежувати споживання їжі, планувати раціон і синхронізувати дані з фітнес-трекерами. Система надає доступ до величезної бази продуктів із їхньою калорійністю, що робить її потужним інструментом для аналізу харчування. Проте MyFitnessPal не призначений для створення рецептів або інтерактивного введення інформації про кулінарію. Інтерфейс орієнтований на відстеження раціону, але не підтримує багаторівневого доступу чи захисту від шкідливої інформації, що може бути недоліком для використання в професійних середовищах.

Для більш детального аналізу було створено порівняльну таблицю, яка демонструє переваги та недоліки кожної платформи, а також підкреслює унікальні характеристики вашої програми. У таблиці порівнюються такі аспекти, як можливість інтерактивного введення даних, персоналізація на основі великих обсягів даних, інтеграція з API, захист від шкідливої інформації та підтримка багаторівневого доступу. Це дозволяє чітко виділити сильні сторони вашої розробки у порівнянні з існуючими аналогами.

Таблиця

Таблиця 1.1 Порівняння моделей

Назва	Ключове спрямування	Особливості	Переваги	Недоліки
Yummly	Пошук і персоналізація рецептів	Інтеграція зі смарт-пристроями	Зручний інтерфейс, персоналізація	Відсутність багаторівневого доступу
Mealime	Планування меню	Створення списків покупок	Простота використання	Відсутність розширеного аналізу дієт
Whisk	Збереження та організація рецептів	Створення персональних кулінарних книг	Легке збереження рецептів	Відсутність аналізу алергенів
MyFitness Pal	Моніторинг харчування	Відстеження калорій	Детальний аналіз калорійності	Відсутність створення рецептів
Culinary guru	Інтерактивна кулінарна платформа з ШІ	Інтеграція моделі LLaMA, багаторівневий доступ	Комплексний підхід, сучасний захист	Високі вимоги до якості даних

Особливості додані до таблиці:

- Ключове спрямування кожної платформи;
- Особливості, які розкривають інноваційні або ключові функції кожного аналога.

Ця розширена таблиця демонструє сильні сторони вашої програми, зокрема її здатність інтегрувати сучасну мовну модель для вирішення складних задач кулінарії та забезпечувати високий рівень захисту інформації й доступності.

1.4 Постановка задачі

Метою даного дослідження є розробка інтегрованої вебплатформи на основі технологій штучного інтелекту для автоматизації кулінарних процесів, персоналізації обслуговування клієнтів та створення нових бізнес-моделей у кулінарній індустрії. Розробка здійснюється в рамках кваліфікаційної роботи

магістра за спеціальністю 122 «Комп'ютерні науки».

Розроблений програмний продукт має задовольняти наступні вимоги:

- створення рецептів із врахуванням дієтичних потреб та алергенів;
- аналіз складу страв та автоматизоване формування персональних меню;
- оцінка калорійності страв на основі введених інгредієнтів;
- інтеграція засобів управління взаємовідносинами з клієнтами для збереження історії взаємодій та підвищення лояльності;
- автоматизація процесів управління запасами, планування закупівель та моніторингу витрат.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Виконати аналіз ринку кулінарної індустрії, визначити основні проблеми та актуальність використання штучного інтелекту.
2. Порівняти існуючі рішення на ринку та оцінити їхні можливості щодо інтеграції CRM та штучного інтелекту.
3. Обрати стек технологій для розробки вебплатформи (MERN: MongoDB, Express, React, Node.js) та інтеграції мовної моделі LLaMA.
4. Реалізувати модель інформаційної системи з використанням штучного інтелекту для автоматизації процесів та надання персоналізованих рекомендацій.

Предметом дослідження є використання штучного інтелекту для автоматизації кулінарних процесів і персоналізації обслуговування в галузі ресторанного бізнесу.

Наукова новизна дослідження полягає у впровадженні інтегрованої системи, яка поєднує можливості штучного інтелекту з функціями CRM для вирішення ключових викликів кулінарної індустрії. Зокрема, відмінністю від існуючих аналогів є можливість використання мовної моделі для створення персоналізованих рекомендацій, що враховують індивідуальні потреби клієнтів, їхні дієтичні обмеження та алергени.

Практична значимість розробленої платформи полягає у підвищенні ефективності роботи кулінарних підприємств, зниженні витрат на управління

запасами та автоматизації обслуговування клієнтів, що сприятиме зростанню лояльності та конкурентоспроможності закладів харчування.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ ТА ВИБІР ТЕХНОЛОГІЙ

2.1 Типи мовних моделей: порівняння та вибір відповідної (LLaMA)

Мовні моделі є однією з центральних технологій штучного інтелекту, особливо у галузі обробки природної мови (NLP). Їхня здатність до генерації тексту, аналізу даних і взаємодії з користувачами відкриває нові можливості для реалізації складних програмних рішень. У сучасних умовах існує широкий вибір мовних моделей, кожна з яких орієнтована на певні завдання і має унікальні технічні особливості. У цьому розділі розглянуто типи мовних моделей, проведено порівняння їхніх характеристик і обґрунтовано вибір моделі LLaMA для інтеграції в проєкт.

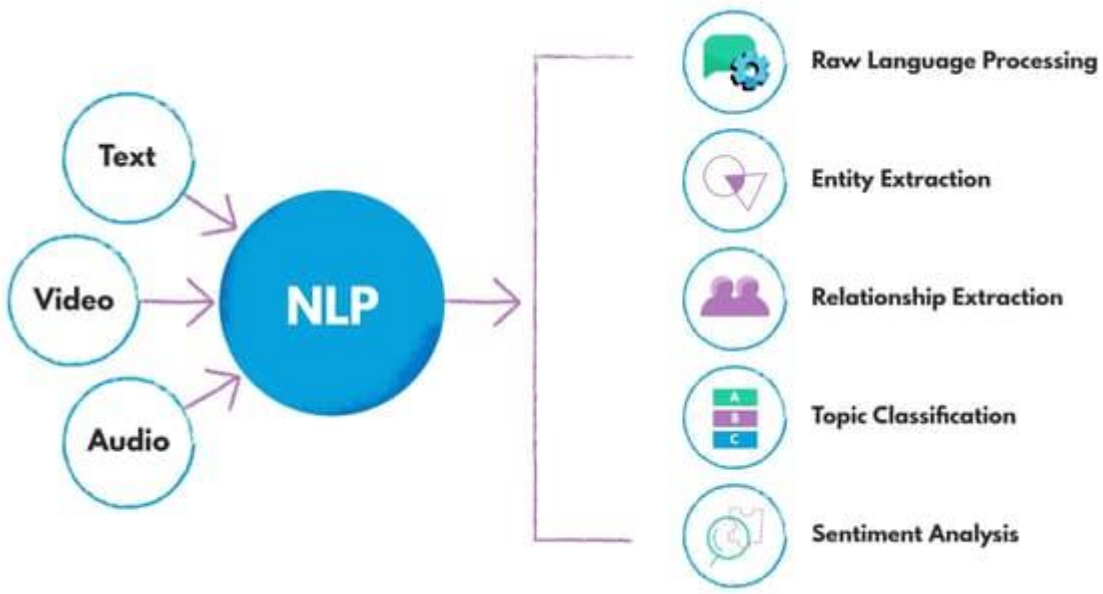

Рисунок 2.1 – Процес обробки природної мови (NLP) [14]

Процес обробки природної мови (NLP): Вхідні дані у вигляді тексту, відео та аудіо обробляються за допомогою NLP для виконання різних завдань, таких як: попередня обробка мови (Raw Language Processing), вилучення сутностей (Entity Extraction), вилучення взаємозв'язків (Relationship Extraction), класифікація тем (Topic Classification) та аналіз настроїв (Sentiment Analysis).

Однією з перших значних мовних моделей була GPT (Generative Pre-trained Transformer), розроблена компанією OpenAI. Вона є генеративною моделлю, яка здатна створювати зв’язний текст на основі заданого контексту. GPT працює на

основі трансформерів, архітектура яких дозволяє моделі ефективно обробляти великі обсяги текстових даних, враховуючи довготривалі залежності між словами. GPT здобула популярність завдяки своїй універсальності: вона використовується для написання текстів, перекладу, створення чат-ботів і багатьох інших завдань. Однак її недоліками є висока обчислювальна складність і потреба у великих ресурсах для тренування, що може бути обмеженням для інтеграції у спеціалізовані платформи з обмеженими ресурсами.

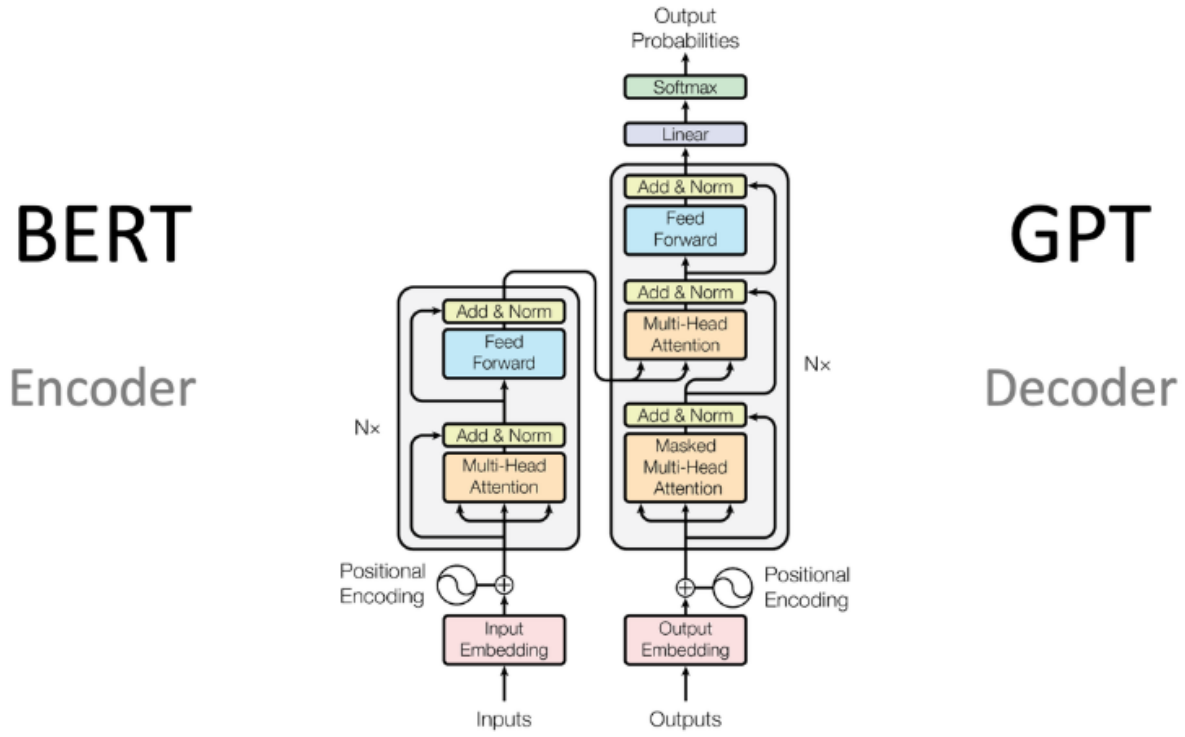

Рисунок 2.2 – Архітектура трансформерів [15]

Іншою важливою моделлю є BERT (Bidirectional Encoder Representations from Transformers), створена Google. BERT побудована для задач розуміння тексту, таких як класифікація, виявлення сутностей і відповіді на запитання. Її унікальність полягає у двонаправленій природі, що дозволяє враховувати контекст як зліва, так і справа від поточного слова. Це забезпечує моделі високу точність у завданнях аналізу тексту. Проте BERT менш ефективна в генеративних завданнях, таких як створення нових текстів, що обмежує її використання у контексті інтерактивних платформ, які вимагають генерації креативного контенту.

LLaMA (Large Language Model Meta AI) є новітньою мовною моделлю,

розробленою Meta AI, яка поєднує переваги ефективності й функціональності. Ця модель створена з акцентом на компактність і продуктивність, що робить її ідеальною для використання у середовищах із обмеженими ресурсами. LLaMA базується на архітектурі трансформерів і демонструє високу якість генерації тексту, порівнянну з більш великими моделями, такими як GPT-3. Її ключовою перевагою є можливість навчання на відносно невеликих наборах даних і низькі апаратні вимоги для розгортання. Це дозволяє інтегрувати модель у спеціалізовані програми, такі як кулінарні платформи, що працюють із великими обсягами структурованих і неструктурованих даних.

Серед інших важливих моделей варто згадати T5 (Text-to-Text Transfer Transformer) і XLNet. T5 є універсальною моделлю, яка перетворює всі задачі NLP у формат "текст-в-текст", що робить її дуже гнучкою. Водночас XLNet, будучи покращеною версією BERT, демонструє високі показники в багатьох задачах завдяки своєму інноваційному підходу до врахування контексту. Однак, як і BERT, ці моделі орієнтовані більше на аналіз, ніж на генерацію тексту, що обмежує їхню корисність у проєктах, які потребують створення контенту.

У процесі порівняння мовних моделей було враховано кілька ключових критеріїв. Перш за все, розглядалася здатність моделі працювати з великими обсягами даних кулінарного контенту, таких як рецепти, інгредієнти, дієти та алергени. Другим важливим аспектом була ефективність моделі в умовах обмежених апаратних ресурсів, що є актуальним для інтеграції в спеціалізовану вебплатформу. Третім критерієм стала можливість легкої адаптації моделі до специфічних завдань, включаючи аналіз і генерацію тексту, пов’язаного з кулінарією.

На основі аналізу обрано модель LLaMA, яка найкраще відповідає потребам проєкту. Її архітектура забезпечує високу точність і продуктивність, дозволяючи інтерактивно працювати з користувачами. Модель може бути додатково навчена на спеціалізованих наборах даних, які включають мільйони рецептів, інформацію про алергени, калорійність та дієтичні обмеження. Завдяки цьому LLaMA здатна генерувати релевантні й точні відповіді на запити користувачів, пропонувати рецепти відповідно до заданих умов і навіть

аналізувати введену інформацію на предмет відповідності дієтичним вимогам.

Ще однією перевагою LLaMA є її компактність, яка дозволяє запускати модель на відносно простих обчислювальних системах. Це робить її доступною для інтеграції у вебплатформи, які можуть мати обмежені серверні ресурси. Крім того, модель підтримує API-інтеграцію, що спрощує її використання в багатофункціональних системах із різними рівнями доступу. Наприклад, адміністративний рівень може включати розширені функції аналізу даних, тоді як користувацький рівень зосереджується на створенні та підборі рецептів.

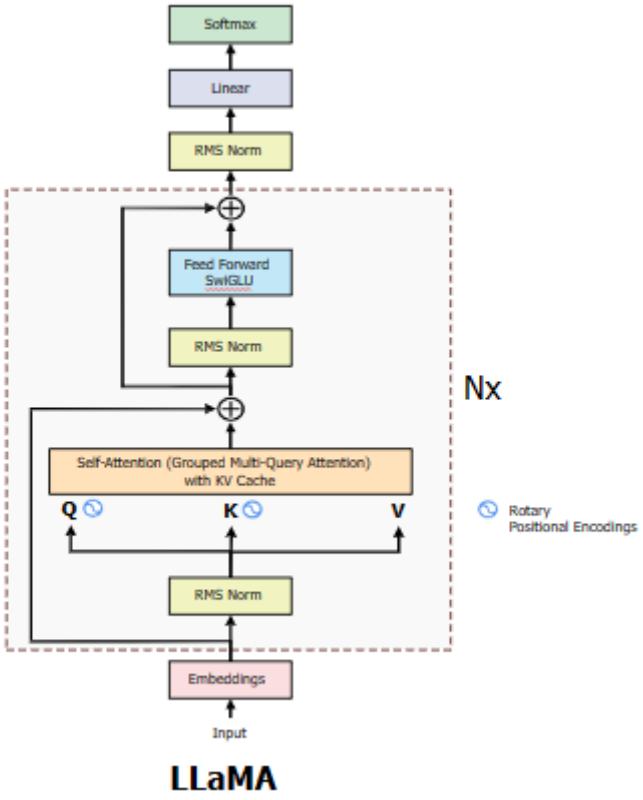

Рисунок 2.3 – Архітектура лама [16]

Особливу увагу під час вибору моделі приділено аспекту безпеки. LLaMA має вбудовані механізми фільтрації контенту, що знижує ризик генерування шкідливої або небажаної інформації. Це особливо важливо в контексті кулінарної платформи, де точність і надійність інформації є критичними. Наприклад, помилка в аналізі алергенів може мати серйозні наслідки, тому система повинна забезпечувати максимальну точність і захист.

Таким чином, вибір LLaMA як мовної моделі для інтеграції в кулінарну

платформу є обґрунтованим рішенням, яке базується на її універсальності, продуктивності та відповідності технічним і функціональним вимогам. Модель здатна забезпечити високу якість обслуговування користувачів, автоматизувати рутинні завдання, такі як створення рецептів і аналіз дієт, а також підтримувати інтерактивну взаємодію в реальному часі. У поєднанні з можливістю адаптації до специфічних потреб і ефективною інтеграцією через API, LLaMA стає центральним компонентом платформи, спрямованої на вдосконалення кулінарної індустрії.

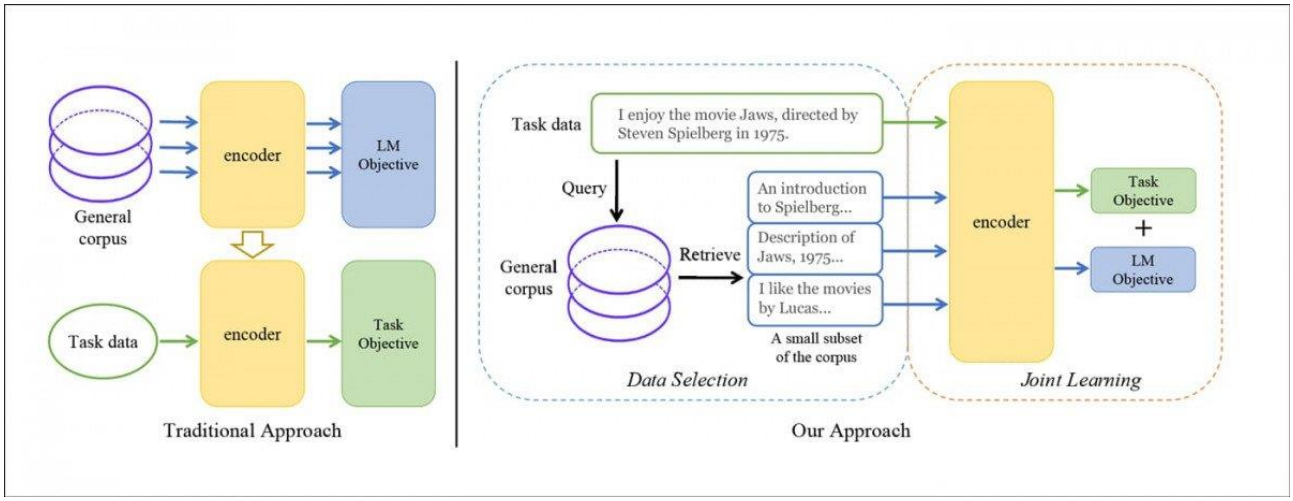

Рисунок 2.4 – Візуалізація створення мовних моделей [17]

У таблиці представлено порівняльний аналіз п'яти основних мовних моделей: GPT, BERT, LLaMA, T5 та XLNet. Кожна з моделей має свої унікальні характеристики, переваги та недоліки, що визначають їхню придатність для виконання різних задач у сфері обробки природної мови.

Таблиця 2.1 Порівняння мовних моделей

Модель	Призначення	Переваги	Недоліки
GPT	Генерація тексту	Висока якість генерації	Високі обчислювальні вимоги
BERT	Аналіз тексту	Висока точність контекстного аналізу	Не підходить для генерації тексту

Продовження таблиці 2.1

LLaMA	Генерація та аналіз тексту	Ефективність на менших ресурсах, адаптація під конкретні задачі	Менший обсяг попереднього навчання
T5	Генерація тексту та аналіз	Гнучкість у роботі з різними задачами	Високі обчислювальні витрати
XLNet	Аналіз тексту	Покращений контекстний аналіз	Складність і високі ресурси для обчислень

GPT є потужною мовною моделлю для генерації тексту, що забезпечує високу якість і природність відповіді, однак її основний недолік полягає у великих вимогах до обчислювальних ресурсів. BERT, у свою чергу, надає високу точність у задачах аналізу тексту завдяки своєму двонаправленому підходу до обробки контексту, але не підходить для генерації тексту, що обмежує її застосування. T5 забезпечує гнучкість у роботі з різними задачами " Text-to-Text Transfer Transformer ", але має високі обчислювальні витрати, що ускладнює її інтеграцію в середовища з обмеженими ресурсами. XLNet, вдосконалена версія BERT, покращує контекстний аналіз і є ефективною у завданнях, пов'язаних з обробкою тексту, але також потребує значних ресурсів [18].

Особливу увагу варто звернути на модель LLaMA, яка є найбільш придатною для інтеграції в кулінарну платформу завдяки її здатності працювати з меншими апаратними ресурсами, зберігаючи високу продуктивність. Важливою перевагою LLaMA є можливість легко адаптувати модель до конкретних задач кулінарної індустрії, таких як створення рецептів або аналіз алергенів. Це дозволяє використовувати LLaMA навіть на серверах середнього класу або локальних комп'ютерах, що робить її економічно вигідною для впровадження в різноманітних комерційних і некомерційних проектах. Завдяки своїй гнучкості та можливості донавчання під конкретні задачі, LLaMA дозволяє забезпечити персоналізоване обслуговування користувачів і автоматизувати багато рутинних процесів, що є необхідними для підвищення ефективності

роботи кулінарної платформи.

2.2 Процес навчання мовної моделі та оптимізації

Навчання мовної моделі є складним і багатоступеневим процесом, який вимагає ретельної підготовки даних, оптимізації параметрів і ефективного використання доступних апаратних ресурсів. У цьому випадку навчання моделі LLaMA було виконане на власному ПК із графічною картою RTX 4070 та процесором Ryzen 5 5600X, що є технічно потужним, але все ж має обмежені ресурси порівняно з великими серверними установками. Цей процес передбачав кілька ключових етапів, кожен з яких був спрямований на адаптацію моделі до завдань кулінарної платформи.

Першим етапом було формування навчального корпусу даних. Вибір якісних і репрезентативних даних має вирішальне значення для ефективного навчання моделі. У даному випадку навчання було засноване на великому обсязі кулінарної інформації, що включає мільйони рецептів, дані про інгредієнти, дієти, алергени та калорійність.

```
        додати часник і обсмажити до золотистого кольору. Додати помідори, сіль, перець
        та тушкувати 10-15 хвилин. Змішати спагетті з соусом, додати базилік перед
        подачею.
88613
88614   Назва рецепту: Оладки з кабачків
88615   Категорія: Закуски
88616   Інгредієнти:
88617   - Кабачок: 1 шт
88618   - Яйце: 1 шт
88619   - Борошно: 3 ст. л.
88620   - Сіль, перець: за смаком
88621   - Олія для смаження: 2 ст. л.
88622   Опис приготування: Кабачок натерти на тертці та злегка віджати сік. Додати яйце,
        борошно, сіль і перець, перемішати. На розігріту сковороду налити олію та
        викладати ложкою кабачкову масу, формуючи оладки. Смажити з обох боків до
        золотистої скоринки.
88623
88624   Назва рецепту: Фруктовий салат
88625   Категорія: Десерти
88626   Інгредієнти:
88627   - Яблуко: 1 шт
88628   - Банан: 1 шт
88629   - Апельсин: 1 шт
88630   - Мед: 1 ст. л.
88631   - Лимонний сік: 1 ч. л.
88632   Опис приготування: Яблуко та банан нарізати кубиками, апельсин очистити та
        розділити на дольки. Змішати всі фрукти, додати мед і лимонний сік. Перемішати та
        подати до столу.
88633
```

Рисунок 2.5 – Масив даних для навчання моделі

Джерела даних включали відкриті бази рецептів, кулінарні книги, наукові статті про харчування та дані з платформ, що займаються плануванням дієт. Дані пройшли ретельну підготовку, яка включала очищення від зайвого тексту, форматування та видалення дублювань. Оскільки кулінарна інформація має специфічну структуру, були розроблені спеціальні алгоритми для автоматичної категоризації даних, таких як розподіл інгредієнтів за категоріями, виділення кроків приготування та маркування дієтичних обмежень [19].

Алгоритм автоматичної категоризації кулінарних даних, розділяє інформацію про рецепти на відповідні категорії, щоб спростити процес їх аналізу і полегшити пошук та обробку кулінарної інформації. Кулінарні дані мають складну структуру, що включає різноманітні категорії інгредієнтів, кроки приготування, дієтичні обмеження та складається з багатьох етапів. Кожний етап алгоритму можна описати формулою.

Формула для автоматичної категоризації даних а саме категоризація інгредієнтів:

$$C_i = f(I_n) \tag{2.1}$$

Де C_i— категорія інгредієнта, f — функція класифікації, а (I_n)— конкретний інгредієнт.

Функція класифікації f може використовувати ключові слова для визначення, до якої категорії належить інгредієнт, наприклад, "Овочі", "Фрукти", "М'ясо", "Заправка".

Виділення кроків приготування:

$$S_k = Splint(T_o, '.') \tag{2.2}$$

де S_k — набір кроків приготування, T_o — текст опису приготування, а $Splint$ — функція розбиття тексту на кроки за роздільником (тут — крапка і пробіл).

Маркування дієтичних обмежень:

$$D_m = g(I) \tag{2.3}$$

де D_m — набір дієтичних обмежень, g — функція маркування обмежень на основі наявних інгредієнтів I. Функція g перевіряє, чи містять інгредієнти певні компоненти, такі як глютен або продукти тваринного походження, і додає

відповідне маркування ("Без глютену", "Вегетаріанське" тощо).

Ця формула дає змогу алгоритму автоматично категоризувати інгредієнти, виділяти кроки приготування та маркувати дієтичні обмеження, забезпечуючи точність і зручність у роботі з кулінарною інформацією. Та прогнати тестові дані через алгоритм, щоб оптимізувати та спростити навчання мовної моделі, і скоротити час та навантаження.

Другим етапом стала підготовка моделі до навчання. Модель LLaMA спочатку завантажується в базовій конфігурації, що містить попередньо натреновані параметри на загальному корпусі даних. У цьому проєкті було використано донавчання, тобто адаптацію моделі до спеціалізованого завдання кулінарії. Для цього необхідно було налаштувати модель на роботу з великими текстовими корпусами, які містять специфічну лексику та структуру. Гіперпараметри, такі як розмір навчального пакета, швидкість навчання та кількість епох, були налаштовані з урахуванням апаратних обмежень ПК. Графічна карта RTX 4070, незважаючи на високу продуктивність, має обмежену пам’ять, тому пакет даних для навчання був оптимізований, щоб уникнути перевантаження системи.

Навчання розпочалося з базового прогону через навчальний набір даних, що дозволило моделі адаптуватися до нового контексту. Для цього використовувалися алгоритми градієнтного спуску, які поступово оновлюють параметри моделі, щоб мінімізувати помилки прогнозу.

Градієнтний спуск — це один із найпоширеніших методів оптимізації для навчання мовних моделей, таких як LLaMA. Його мета полягає в тому, щоб знайти набір параметрів моделі, який мінімізує функцію втрат, забезпечуючи якнайкраще узгодження між вхідними даними та передбаченням моделі. Розглянемо основні етапи градієнтного спуску для вашої кулінарної моделі.

Ініціалізація параметрів**.** На початку навчання модель має випадкові значення параметрів (ваг), які будуть поступово коригуватися для зменшення похибок. Ініціалізація визначає початкову точку, з якої модель починає навчання.

Обчислення функції втрат. Функція втрат $(\mathrm{L}(\theta))$

визначає, наскільки добре модель робить прогнози. Для кулінарної моделі функція втрат може оцінювати відповідність генерованих описів рецептів до наявних даних або правильність класифікації інгредієнтів.

Обчислення градієнта. Градієнт функції втрат по кожному параметру визначає напрямок, у якому потрібно змінювати параметр, щоб зменшити похибку. Він обчислюється як похідна функції втрат відносно кожного параметра:

$$\nabla \mathrm{L}(\theta) = \left(\frac{bL}{b\theta_1}, \frac{bL}{b\theta_2}, \ldots, \frac{bL}{b\theta_n} \right) \tag{2.4}$$

де θ — це набір параметрів (ваг) моделі.

Оновлення параметрів. Параметри моделі оновлюються на основі градієнта і заданого кроку навчання (a). Крок навчання визначає, наскільки значними будуть зміни параметрів на кожній ітерації:

$$\theta = \theta - a\nabla \mathrm{L}(\theta) \tag{2.5}$$

У контексті кулінарної моделі це означає поступове покращення прогнозів моделі щодо опису рецептів, розподілу інгредієнтів або аналізу дієтичних обмежень.

Повторення ітерацій. Процес повторюється для кожного набору навчальних даних — кулінарних рецептів, поки функція втрат не досягне мінімального значення або не буде досягнуто заданої кількості ітерацій. Градієнтний спуск дозволяє моделі поступово знаходити найкращі параметри, які забезпечать оптимальне виконання задач.

Особливості градієнтного спуску для LLaMA. Однією з ключових особливостей моделі LLaMA є її здатність працювати з меншими обчислювальними ресурсами. Це важливо, оскільки градієнтний спуск може бути дорогим з точки зору обчислювальних витрат. Для LLaMA ефективність градієнтного спуску досягається завдяки спеціальним оптимізаційним

алгоритмам, таким як Adam або RMSProp, які допомагають контролювати розмір кроку та враховують попередню інформацію про градієнти.

Таким чином, градієнтний спуск для кулінарної моделі дозволяє поступово покращувати параметри моделі, щоб забезпечити якісні передбачення і категоризацію рецептів та інгредієнтів. Цей підхід дозволяє моделі адаптуватися до складних запитів користувачів і забезпечувати персоналізовані результати, враховуючи специфічні потреби кулінарної

2.3 Регуляризація моделі

Однією з головних технік, яка використовувалася під час навчання, була регуляризація, яка запобігає перенавчанню моделі. Регуляризація забезпечує збалансованість між здатністю моделі точно відтворювати навчальні дані та її здатністю до узагальнення, що особливо важливо для роботи з новими, невідомими запитами користувачів.

Особливістю навчання, була інтеграція спеціальних обмежень пов’язаних із контентом, а саме модель мала навчитися уникати шкідливої інформації та зокрема недостовірних даних про харчові обмеження чи помилкових рекомендацій. Для цього було розроблено спеціальні фільтри та сценарії, які оцінювали відповіді моделі під час навчання. Якщо модель генерувала текст, який не відповідав заданим стандартам, такі відповіді блокувалися та враховувалися як помилки, що підлягають корекції.Основна ідея полягає в тому, щоб забезпечити автоматичну перевірку та блокування невідповідних відповідей, які згодом враховуються як помилки для корекції. Нижче наведені основні етапи цього процесу та відповідні формули [20].

Оцінка релевантності відповіді**.** Фільтр релевантності визначає, чи відповідають інгредієнти і опис приготування заданій категорії страви.

Формула для оцінки релевантності:

$$R(i) = \begin{cases} 1, \text{якщо всі інгрідієнти } I \text{ відповідають категорії страви } C \\ 0, \quad \text{якщо } I \text{ містить невідомі інгрідінти} \end{cases} \quad (2.6)$$

Де $R(i)$ — релевантність інгредієнтів, I — список інгредієнтів а C — категорія страви. Відповідь блокується, якщо $R(i) = 0$

Перевірка точності інструкцій**.** Фільтр точності аналізує кожен крок

приготування на логічну послідовність і узгодженість з інгредієнтами.

Формула для аналізу послідовності:

$$A(k) = \sum_{n-1}^{N} P(i_n, k_n) \tag{2.7}$$

Де $A(k)$ — узгодженість інструкцій, $P(i_n, k_n)$ — функція, яка повертає 1, якщо інгредієнт i_n використовується у відповідному кроці k_n, і 0 — якщо ні. Якщо $A(k) = N$, то відповідь блокується.

Перевірка дієтичних обмежень**.** Фільтр дієтичних обмежень перевіряє, чи дотримуються вимоги, зазначені для кожного рецепту.

Формула для перевірки дієтичних обмежень:

$$D(i) = \begin{cases} 1, \text{якщо інгрідієнтине містять заборонених компонентів} \\ 0, \quad \text{якщо містять заборонених компонентів} \end{cases} \tag{2.8}$$

Де $D(i)$— дотримання дієтичних обмежень, а i — інгредієнти рецепту. Якщо $D(i) = 0$, відповідь блокується.

Оцінка стилю мови фільтр стилю мови перевіряє відповідність стилю тексту кулінарним стандартам та правильність формулювань.

Формула для стилю мови:

$$S(t) = \begin{cases} 1, \text{якщо текст вірний стилю і не містить помилок} \\ 0, \text{якщо текст містить помилки або не відповідає стандаотам} \end{cases} \tag{2.9}$$

де $S(t)$ — відповідність стилю мови, а t — текст відповіді. Якщо $S(t) = 0$, відповідь вважається некоректною і блокується.

Семантична відповідність рецепту фільтр семантичної відповідності перевіряє, чи всі частини рецепту (інгредієнти, кроки приготування) відповідають загальній концепції страви.

Формула семантичної відповідності:

$$M(i,k) = \begin{cases} 1, \text{якщо всі частини рецепту логічно узгоджені між собою} \\ 0, \quad \text{якщо є невідомі між інгрієнтами та ероками} \end{cases} \tag{2.10}$$

де $M(i,k) =$ — семантична відповідність між інгредієнтами (i) та кроками приготування (k). Якщо $M(i,k) = 0$, відповідь блокується.

Загальний процес оцінки під час навчання моделі всі фільтри застосовуються до кожної відповіді. Якщо хоч один із фільтрів повертає нуль, відповідь блокується і вважається помилкою, що підлягає корекції.

Формула оцінки відповіді:

$$E = R(i) \times A(k) \times D(i) \times S(t) \times M(i,k) \quad (2.11)$$

де $E = 1$ вказує на прийняття відповіді, а E=0E = 0E=0 вказує на блокування і необхідність корекції.

Цей технічний процес дозволяє забезпечити відповідність текстів, згенерованих моделлю, стандартам якості, релевантності, дієтичним вимогам та стилю, що підвищує точність і коректність кінцевого результату.

Важливим аспектом навчання було використання техніки перевірки на валідаційному наборі даних. Валідація дозволяє оцінити, наскільки добре модель узагальнює свої знання на даних, які не входили до навчального набору. Це забезпечує об’єктивну оцінку якості моделі та дозволяє уникати перенавчання. У процесі валідації модель перевірялася на здатність генерувати рецепти відповідно до заданих умов, таких як дієтичні обмеження чи специфічні інгредієнти. Також перевірялися її відповіді на запити про аналіз алергенів і калорійність страв.

```
88691   - Сіль, перець: за смаком (Приправи)
88692   - Олія для смаження: 2 ст. л. (Заправка)
88693   Опис приготування:
88694   1. Кабачок натерти на тертці та злегка віджати сік.
88695   2. Додати яйце, борошно, сіль і перець, перемішати.
88696   3. На розігріту сковороду налити олію та викладати ложкою кабачкову масу, формуючи ол
88697   4. Смажити з обох боків до золотистої скоринки.
88698   Дієтичні обмеження: Містить глютен
88699
88700   Назва рецепту: Фруктовий салат
88701   Категорія: Десерти
88702   Інгредієнти:
88703   - Яблуко: 1 шт (Фрукти)
88704   - Банан: 1 шт (Фрукти)
88705   - Апельсин: 1 шт (Фрукти)
88706   - Мед: 1 ст. л. (Заправка)
88707   - Лимонний сік: 1 ч. л. (Заправка)
88708   Опис приготування:
88709   1. Яблуко та банан нарізати кубиками, апельсин очистити та розділити на дольки.
88710   2. Змішати всі фрукти, додати мед і лимонний сік.
88711   3. Перемішати та подати до столу.
88712   Дієтичні обмеження: Вегетаріанське, без глютену
```

Рисунок 2.6 – Дані після алгоритму оптимізації

На наступному етапі проводилася оптимізація продуктивності моделі. Це включало скорочення кількості параметрів і адаптацію до апаратних можливостей ПК. Використовувалися методи, які дозволяють зменшити обсяг

обчислень без значного зниження якості результатів. Наприклад, були застосовані квантовані моделі, які зберігають основну функціональність, але спрощують математичні операції. Такий підхід був необхідним для забезпечення стабільної роботи моделі в умовах обмежених ресурсів.

Після завершення навчання модель була протестована на тестовому наборі даних, який включав як стандартні кулінарні запити, так і нестандартні сценарії, що моделюють поведінку реальних користувачів. Наприклад, перевірялися запити на створення складних рецептів із нестандартними інгредієнтами, аналіз меню на відповідність певним дієтам, а також здатність моделі враховувати алергени. Результати тестування були проаналізовані, і модель була дооптимізована для корекції виявлених недоліків.

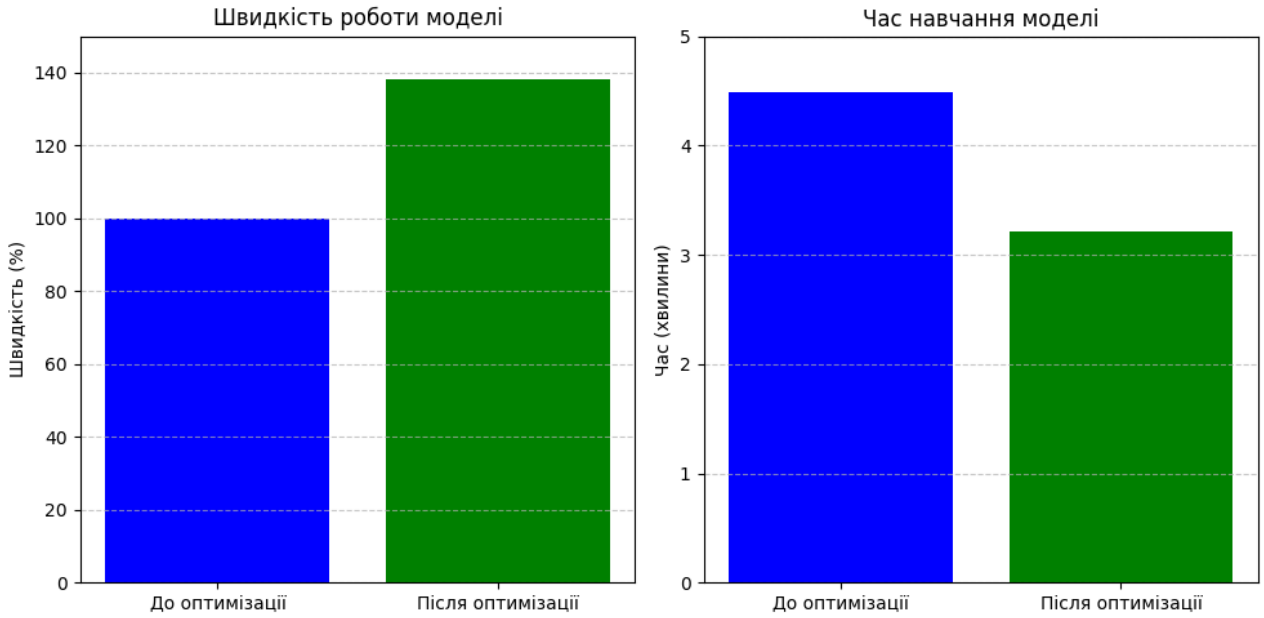

Рисунок 2.7 – Графік оптимізації

На графіках показано порівняння швидкості роботи та часу навчання моделі до і після оптимізації. Лівий графік демонструє зміну швидкості роботи моделі: після оптимізації швидкість зросла на 38%, що видно зі збільшення стовпчика з 100% до 138%. Це означає, що продуктивність моделі значно покращилась після проведених змін, що дозволяє обробляти більше даних за одиницю часу.

Правий графік ілюструє зменшення часу навчання моделі з 4.49 хвилин до 3.21 хвилин після оптимізації. Це зниження часу навчання вказує на ефективніше

використання обчислювальних ресурсів, що дозволяє суттєво скоротити час підготовки моделі до роботи. Такі покращення роблять модель більш ефективною, що є важливим для прискорення процесів її застосування у реальних умовах.

2.4 Інтеграція та продуктивне впровадження

Інтеграція мовної моделі в технологічну платформу є одним із ключових етапів її впровадження в реальні умови. Цей процес передбачає поєднання можливостей моделі з функціоналом платформи для забезпечення максимальної продуктивності та зручності використання кінцевими користувачами. У випадку інтеграції моделі LLaMA у кулінарну платформу було реалізовано кілька важливих технічних і організаційних завдань, кожне з яких спрямоване на створення ефективного та стабільного середовища для роботи моделі. Створено та детально описано етапи інтеграції, які включали налаштування API, адаптацію до архітектури платформи, забезпечення продуктивності та масштабованості, а також контроль якості роботи.

Процес інтеграції розпочався з налаштування API для забезпечення комунікації між мовною моделлю та основними компонентами платформи [21]. API є ключовою частиною архітектури, яка дозволяє платформі надсилати запити до моделі та отримувати відповіді у стандартизованому форматі. Це налаштування включало вибір протоколу взаємодії (наприклад, REST або gRPC), створення точок входу для запитів та визначення форматів даних для обміну. Основною вимогою було забезпечення швидкої та безперебійної передачі даних, що є критичним для роботи в реальному часі. Наприклад, запити на створення рецептів чи аналіз алергенів оброблятися без затримок, щоб користувачі отримували відповіді у зручному для них темпі. Було також впроваджено механізми кешування для повторюваних запитів, що зменшило навантаження на сервери та прискорило обробку даних.

```python
            return jsonify({"error": "Помилка при видаленні рецепту"}), 500

    # Ендпоінт для оновлення рецепту
    @app.route('/api/recipe/<recipe_id>', methods=['PUT'])
    def update_recipe(recipe_id):
        data = request.json
        if not data:
            return jsonify({"error": "Необхідні дані для оновлення відсутні"}), 400

        try:
            result = recipes_collection.update_one(
                {"_id": ObjectId(recipe_id)},
                {"$set": data}
            )
            if result.matched_count > 0:
                logging.info(f"Рецепт з ID {recipe_id} оновлено")
                return jsonify({"message": "Рецепт успішно оновлено"}), 200
            else:
                return jsonify({"error": "Рецепт не знайдено"}), 404
        except Exception as e:
            logging.error(f"Помилка при оновленні рецепту: {e}")
            return jsonify({"error": "Помилка при оновленні рецепту"}), 500

    # Фоновий процес для моніторингу оновлень

    def monitor_updates():
        while True:
            try:
                logging.info("Перевірка оновлень бази даних...")
                # Наприклад, перевіряти нові або змінені рецепти
                time.sleep(60)  # Перевіряти кожну хвилину
            except Exception as e:
                logging.error(f"Помилка під час моніторингу оновлень: {e}")

    # Запуск фонового потоку моніторингу
    monitor_thread = Thread(target=monitor_updates)
    monitor_thread.daemon = True
    monitor_thread.start()

    # Основна точка входу
    if __name__ == "__main__":
        try:
            app.run(host="0.0.0.0", port=int(api_port), debug=True)
        except Exception as e:
            logging.error(f"Помилка при запуску сервера: {e}")
```

Рисунок 2.8 – Інтиграція по API

Далі процес інтеграції передбачав адаптацію моделі до архітектури платформи, розробленої на стеку MERN (MongoDB, Express, React, Node.js). Ця архітектура забезпечує високу гнучкість та ефективність роботи, однак вимагає налаштування моделі на роботу з конкретними компонентами. Наприклад, модуль на базі Node.js був використаний для реалізації взаємодії з API моделі, що дозволило легко інтегрувати її в серверну частину платформи. Крім того, було розроблено фронтенд на React, який забезпечує зручний інтерфейс для користувачів, дозволяючи їм вводити запити, отримувати відповіді та взаємодіяти з функціоналом платформи. На цьому етапі важливим завданням стало налаштування відповідності між вихідними даними моделі та вимогами інтерфейсу. Наприклад, відповіді моделі формувалися у структурованому вигляді, щоб їх легко можна було відобразити у форматі списків, таблиць або інтерактивних елементів на вебсторінці.

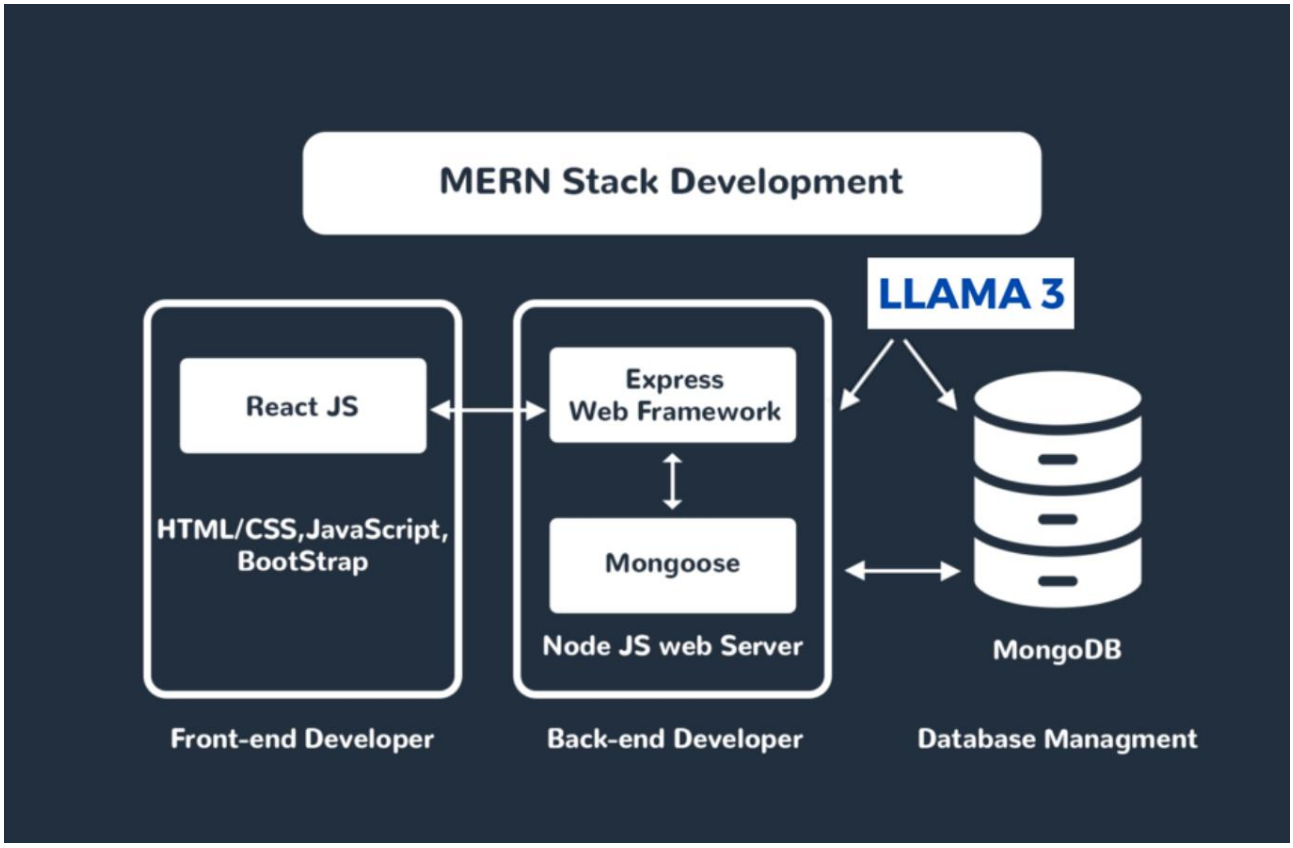

Рисунок 2.9 – Архітектура

Особливу увагу було приділено продуктивності та масштабованості. Хоча модель LLaMA відома своєю ефективністю, її інтеграція у реальне середовище вимагала додаткової оптимізації. Для цього було реалізовано кілька стратегій. По-перше, застосовувалося паралельне виконання запитів, що дозволило одночасно обробляти кілька користувацьких запитів без зниження швидкості відповіді. По-друге, модель була налаштована на виконання найпоширеніших запитів із попередньо згенерованими шаблонами відповідей, що зменшило час обробки. Крім того, була передбачена можливість горизонтального масштабування системи шляхом додавання нових серверів для обробки запитів у разі збільшення навантаження. Це забезпечило стабільну роботу навіть під час пікових навантажень, таких як одночасне використання платформи великою кількістю користувачів.

Контроль якості інтеграції був одним із ключових аспектів продуктивного впровадження моделі. Для цього було створено серію тестів, які перевіряли відповідність результатів роботи моделі очікуваним стандартам. Наприклад, запити на створення рецептів тестувалися на повноту та точність, а відповіді

моделі на питання щодо алергенів та дієтичних обмежень оцінювалися на коректність і зрозумілість. Крім того, проводилися навантажувальні тести, які моделювали сценарії з високою інтенсивністю запитів. Результати цих тестів дозволили виявити потенційні вузькі місця в системі та усунути їх ще до запуску платформи.

Важливим етапом інтеграції стало забезпечення безпеки. Оскільки модель працює з чутливою інформацією, такою як дієтичні уподобання та алергії користувачів, були впроваджені спеціальні механізми захисту даних. Це включало шифрування даних під час передачі між платформою та моделлю, обмеження доступу до API через автентифікацію, а також фільтрацію запитів для виявлення потенційно небажаного контенту. Окрім цього, був реалізований захист від атак, таких як спроби перевантаження системи або внесення шкідливої інформації.

Завершальним етапом інтеграції стало впровадження багаторівневого доступу, що дозволило створити різні рівні функціональності для різних категорій користувачів. Наприклад, кінцеві користувачі могли отримувати доступ до функцій створення рецептів та аналізу алергенів, тоді як адміністративний рівень включав розширені інструменти для моніторингу роботи платформи, налаштування моделі та аналізу статистики використання. Це забезпечило гнучкість системи та відповідність її функціоналу потребам різних груп користувачів.

Таким чином, процес інтеграції моделі LLaMA у кулінарну платформу включав низку етапів, спрямованих на забезпечення стабільної, безпечної та зручної роботи системи. Налаштування API, адаптація до архітектури платформи, оптимізація продуктивності, контроль якості та впровадження багаторівневого доступу забезпечили високу ефективність моделі та її здатність відповідати потребам користувачів у реальних умовах. Це дозволило створити інноваційне рішення, яке поєднує можливості штучного інтелекту з практичністю використання в кулінарній галузі.

3 ОПТИМІЗАЦІЯ ЯКОСТІ КУЛІНАРНОЇ ПРОДУКЦІЇ ТА СИСТЕМНИЙ АНАЛІЗ

3.1 Підвищення якості та безпеки страв: моніторинг продуктів, оцінка свіжості інгредієнтів, підтримка дотримання стандартів

Тестування мовної моделі є критично важливим етапом у процесі її впровадження, оскільки воно дозволяє оцінити якість роботи, виявити недоліки та визначити ступінь готовності системи до реального використання. У контексті інтеграції моделі LLaMA в кулінарну платформу тестування було спрямоване на перевірку її функціональності, точності, надійності, продуктивності та відповідності вимогам безпеки. Цей процес складався з кількох етапів, кожен з яких забезпечував комплексну перевірку різних аспектів роботи моделі, а результати дозволили доопрацювати та оптимізувати її перед запуском.

Перший етап тестування передбачав оцінку функціональності моделі. Було перевірено, наскільки добре модель справляється з основними завданнями, для яких вона була створена. Наприклад, моделі ставилися задачі генерації рецептів на основі заданих інгредієнтів, аналізу алергенів у рецептах та підбору страв відповідно до дієтичних обмежень. У процесі тестування проводилося порівняння отриманих результатів із контрольними даними, підготовленими заздалегідь. Цей етап дозволив виявити, чи може модель адекватно обробляти специфічні кулінарні терміни, а також наскільки точно вона враховує контекст запитів користувачів. Одним із ключових аспектів було оцінити здатність моделі працювати з багатозначними запитами, де одна й та ж інформація може бути інтерпретована по-різному залежно від контексту. У разі виявлення неточностей у відповідях модель доопрацьовувалася, що включало внесення змін до алгоритмів обробки даних та покращення навчального корпусу.

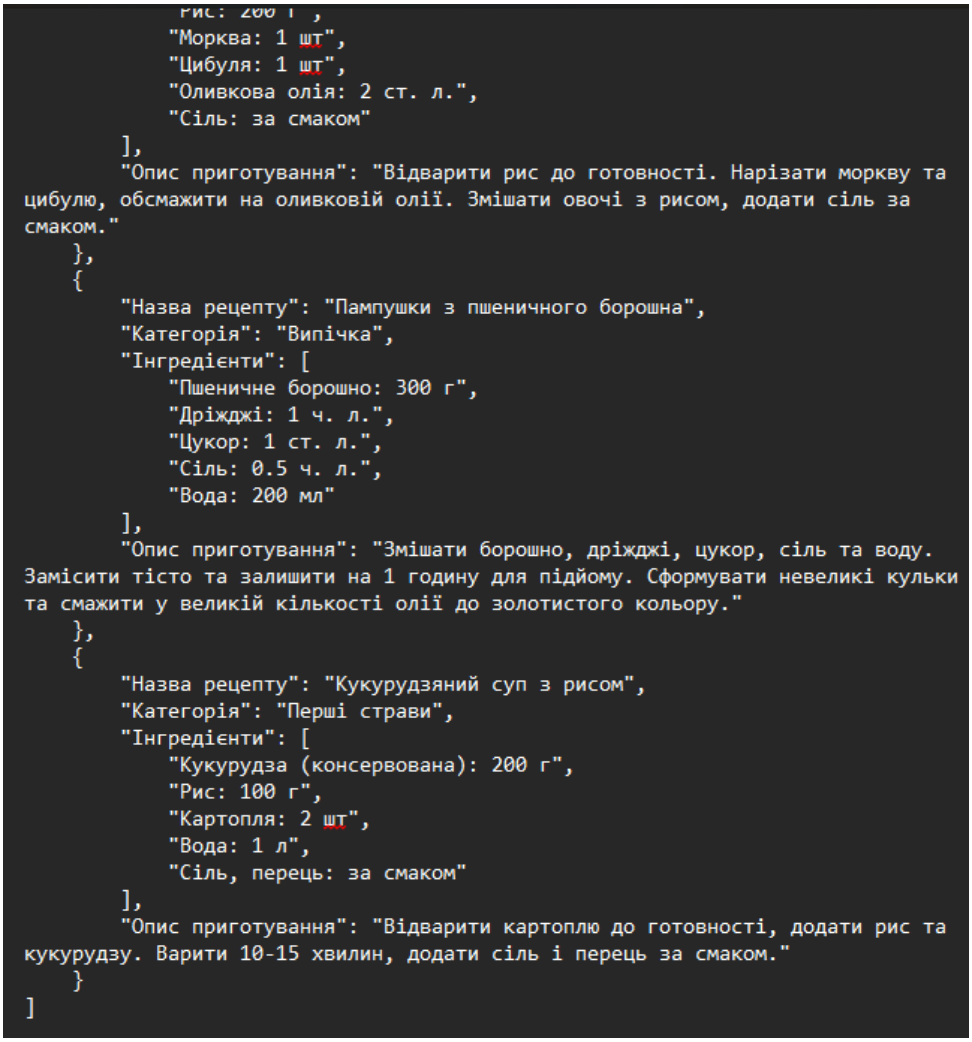

Рисунок 3.1 – Вихідні дані перед оптимізацією

Наступним етапом стало тестування точності моделі. Для цього було створено спеціалізований тестовий набір даних, який включав складні сценарії взаємодії користувачів із системою. Наприклад, модель перевірялася на здатність створювати рецепти із заданими обмеженнями, такими як виключення певних алергенів або обмеження калорійності. Ці тести дозволили визначити, наскільки точно модель розуміє специфічні запити користувачів та адаптує свої відповіді відповідно до їхніх потреб. Для кількісної оцінки точності використовувалися метрики, такі як BLEU (Bilingual Evaluation Understudy) та ROUGE (Recall-Oriented Understudy for Gisting Evaluation), які дозволяють оцінювати відповідність результатів моделі заданим стандартам. Крім того, проводилися вибірковий аналіз, в яких вручну оцінювали результати роботи моделі.

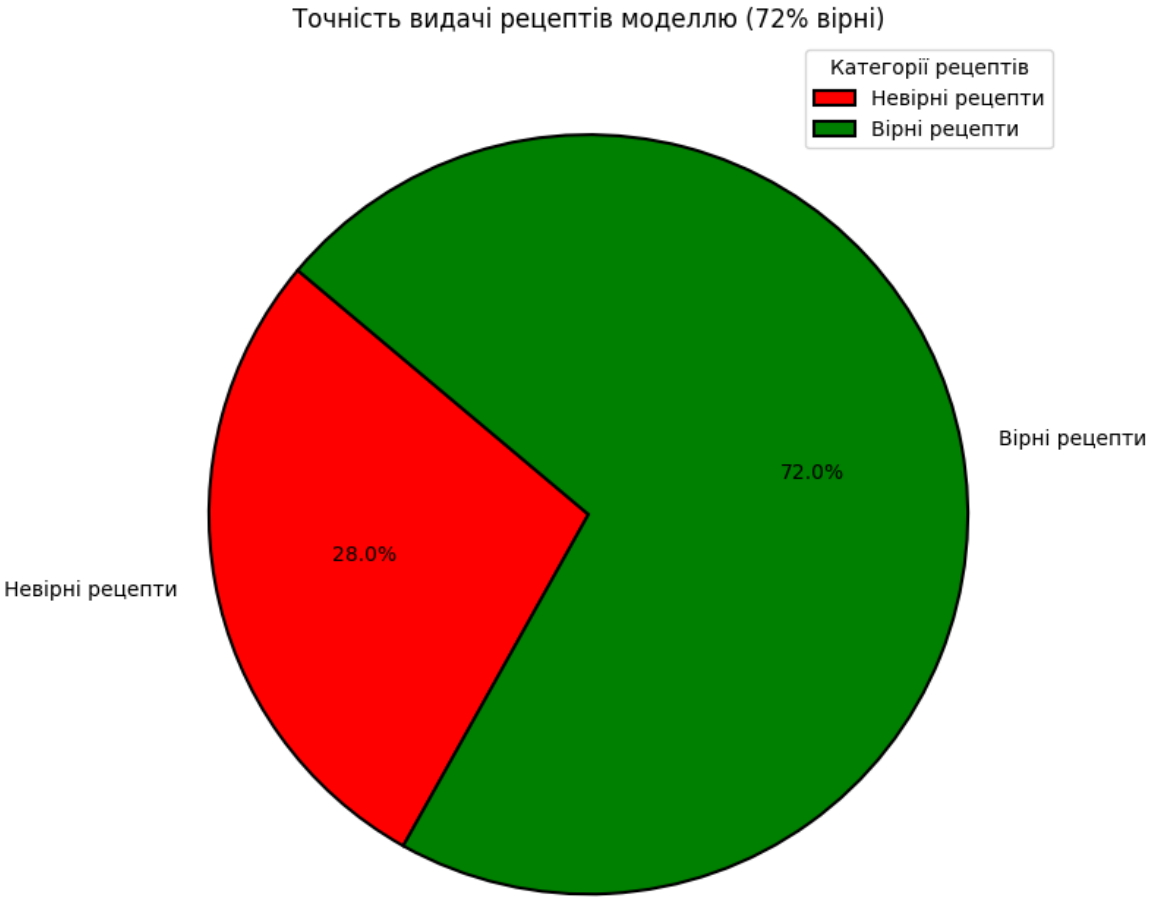

Рисунок 3.2 – точність на 100 генерацій

Особливу увагу було приділено перевірці надійності моделі, тобто її здатності працювати коректно за різних умов. Цей етап включав тестування моделі на обробку великого обсягу запитів, що моделювало ситуації з високим навантаженням на платформу. Наприклад, одночасна генерація кількох рецептів із різними параметрами або аналіз великої кількості страв на відповідність дієтичним обмеженням. Також модель тестувалася на стійкість до некоректних запитів, таких як введення помилкових даних або відсутність деяких параметрів у запиті. У таких випадках модель мала генерувати відповідь, яка чітко пояснює користувачеві, які дані необхідно уточнити або виправити. Висока надійність моделі була досягнута завдяки впровадженню додаткових алгоритмів перевірки вхідних даних та обробки виняткових ситуацій.

Продуктивність моделі була протестована в реальних умовах, які передбачали інтеграцію її у платформу та роботу через API. Було проведено вимірювання часу відповіді моделі на різні типи запитів, щоб оцінити, наскільки швидко система може обробляти дані. Зокрема, увагу приділено запитам, які

вимагають значних обчислювальних ресурсів, таких як створення складних рецептів із багатьма інгредієнтами та умовами.

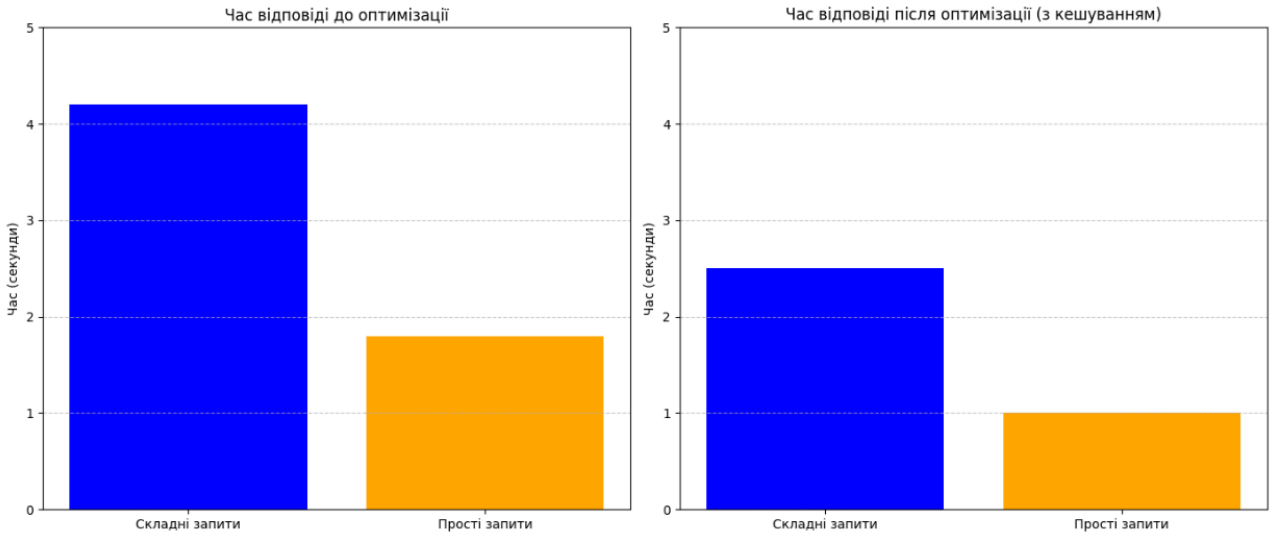

Рисунок 3.3 – Середній час відповіді після оптимізації

Результати цього тестування дозволили оптимізувати алгоритми моделі, а також внести зміни до архітектури платформи для зниження затримок у відповідях. Наприклад, були реалізовані механізми кешування для часто повторюваних запитів, що значно прискорило роботу системи.

Окремий етапом тестування було присвячено перевірці відповідності моделі вимогам безпеки. Було проведено серію тестів, які моделювали потенційні ризики, такі як введення шкідливих даних, спроби отримати небажану інформацію або генерування некоректного контенту. Для цього використовувалися спеціальні фільтри, які аналізували відповіді моделі на наявність небезпечного чи небажаного контенту. Крім того, тестувалися механізми автентифікації та шифрування, які забезпечують захист даних користувачів під час їхньої взаємодії з системою. Цей етап дозволив переконатися, що модель працює безпечно, дотримуючись високих стандартів конфіденційності та захисту інформації.

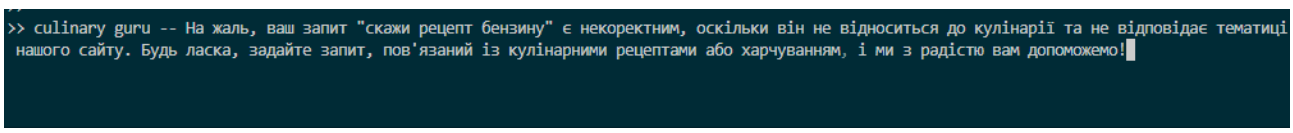

Рисунок 3.4 – Результат відповіді на не коректне запитання

Усі результати тестування були проаналізовані, і на їхній основі було проведено оптимізацію моделі та платформи. Це включало внесення змін до алгоритмів обробки запитів, покращення якості навчального корпусу, а також адаптацію моделі до нових сценаріїв використання. Завдяки цьому модель LLaMA успішно пройшла всі етапи тестування та була готова до продуктивного впровадження, забезпечуючи високу точність, надійність та безпеку роботи.

3.2 Розробка адаптаційних алгоритмів

Розробка адаптаційних алгоритмів є невід'ємною частиною вдосконалення інтегрованих систем на базі мовних моделей, оскільки вони дозволяють моделі реагувати на зміни у вхідних даних і відповідати новим вимогам користувачів. У контексті кулінарної платформи адаптація є критично важливою для підтримання актуальності й точності результатів, враховуючи постійні оновлення в кулінарній індустрії, зміну харчових трендів і появу нових дієтичних рекомендацій. На цьому етапі було розроблено алгоритми, які дозволяють системі ефективно обробляти нову інформацію, автоматично оновлювати дані і вдосконалювати функціональність без необхідності масштабного втручання в проект. Цей розділ детально описує ключові етапи створення адаптаційних алгоритмів, їхню архітектуру, а також результати, які вони дозволили досягти.

Процес розробки адаптаційних алгоритмів розпочався з визначення основних вимог, що висуваються до системи. Однією з основних вимог було забезпечення динамічної адаптації моделі до нових даних. У контексті кулінарної платформи це включало можливість швидкого додавання нових рецептів, інгредієнтів, алергенів та дієтичних рекомендацій до бази даних платформи. Важливо було розробити механізм, який дозволяв би моделі автоматично інтегрувати цю нову інформацію у свої відповіді. Наприклад, якщо в базі даних з'являється новий продукт із певними алергенними властивостями, система повинна одразу враховувати ці дані під час аналізу страв або створення нових рецептів. Для цього було створено алгоритми, що забезпечують автоматичне оновлення моделей даних і їхню інтеграцію у робочий процес

мовної моделі.

Особливу увагу приділили алгоритмам самонавчання, які дозволяють моделі оновлювати свої параметри на основі нових вхідних даних. Такі алгоритми використовують підхід донавчання, що дозволяє моделі вдосконалювати свої знання без необхідності повного повторного навчання на всьому корпусі даних. Це стало особливо важливим у контексті використання моделі на платформі з обмеженими ресурсами, де повне перенавчання може бути надто затратним як за часом, так і за обчислювальними потужностями. Було реалізовано процес, у якому модель періодично оновлюється на основі нових даних, які надходять до бази платформи, при цьому зберігаючи попередньо здобуті знання. Цей підхід дозволив значно підвищити ефективність роботи системи, забезпечуючи її адаптацію до змін у реальному часі.

Другим важливим аспектом розробки адаптаційних алгоритмів стала автоматизація оновлення даних про алергени та дієтичні рекомендації. У сучасній кулінарії ці дані постійно змінюються, оскільки з'являються нові дослідження про властивості інгредієнтів, а також змінюються стандарти харчової безпеки. Алгоритми, які були розроблені, забезпечують автоматичне завантаження нових даних із зовнішніх джерел, таких як офіційні бази даних харчових продуктів чи медичних рекомендацій. Наприклад, якщо у списку алергенів з'являється новий продукт, система завантажує ці дані, аналізує їх і додає до відповідних записів у базі. Цей процес включає перевірку достовірності джерел, обробку даних для їхньої сумісності з існуючою базою та інтеграцію оновлень у модель. Завдяки цьому користувачі завжди мають доступ до актуальної інформації, яка відповідає найновішим стандартам.

Адаптаційні алгоритми також забезпечують врахування змін у вподобаннях користувачів. У процесі роботи платформи накопичуються дані про взаємодію користувачів із системою, такі як пошукові запити, обрані рецепти та введені параметри. Ця інформація використовується для створення моделей поведінки, які дозволяють системі прогнозувати потреби користувачів і вдосконалювати свої відповіді. Наприклад, якщо користувач регулярно шукає рецепти з низьким вмістом калорій, система може автоматично адаптувати

результати пошуку, надаючи перевагу відповідним стравам. Для реалізації цього підходу були розроблені алгоритми аналізу користувацьких даних, які використовують методи машинного навчання для виявлення закономірностей і прогнозування майбутніх запитів.

Окремим напрямом адаптації стало вдосконалення алгоритмів обробки природної мови, які забезпечують інтерпретацію складних і багатозначних запитів користувачів. У кулінарній сфері користувачі часто формулюють запити, які можуть містити неточності або двозначності. Наприклад, запит "рецепт для вечері з куркою без глютену" може мати різні інтерпретації залежно від додаткових умов, які не були зазначені явно. Розроблені алгоритми дозволяють моделі уточнювати такі запити через взаємодію з користувачем, ставлячи уточнювальні питання або надаючи варіанти вибору. Це забезпечує більш точний аналіз і підвищує якість відповідей.

Для забезпечення надійності адаптаційних алгоритмів проводилося їхнє багатоетапне тестування. Було створено тестові сценарії, які моделювали різні типи змін у даних, включаючи додавання нових продуктів, зміну властивостей існуючих інгредієнтів та оновлення дієтичних стандартів. У кожному випадку аналізувалися результати роботи моделі, щоб переконатися у правильності обробки змін і їхній інтеграції у відповіді системи. Крім того, оцінювалася швидкість адаптації, що дозволило визначити, наскільки оперативно система може реагувати на оновлення даних.

Результатом розробки адаптаційних алгоритмів стало створення динамічної системи, яка здатна ефективно працювати в умовах постійних змін. Модель LLaMA, інтегрована в кулінарну платформу, тепер має можливість автоматично оновлювати свої знання, враховуючи нові дані та змінюючи свої відповіді відповідно до потреб користувачів. Ця функціональність не лише підвищила точність і актуальність результатів, але й забезпечила гнучкість системи, що дозволяє їй швидко адаптуватися до змін у кулінарній індустрії. Адаптаційні алгоритми стали важливою частиною загальної архітектури платформи, підвищуючи її конкурентоспроможність і забезпечуючи високий рівень задоволеності користувачів.

3.3 Системний аналіз

Системний аналіз є комплексним процесом, спрямованим на оцінку ефективності роботи мовної моделі в умовах реального використання. Цей етап передбачає всебічне дослідження різних аспектів функціонування моделі, включаючи точність, швидкість відповіді, масштабованість, зручність для користувачів і загальну відповідність очікуванням. Для досягнення об’єктивних результатів використовуються спеціалізовані метрики, що дозволяють оцінити продуктивність системи з кількісної та якісної точок зору. У процесі системного аналізу модель LLaMA була детально протестована, щоб визначити її сильні сторони, можливі недоліки та шляхи подальшого вдосконалення.

Перший важливий аспект цього аналізу — визначення точності моделі, що показує, наскільки коректно модель відповідає на запити користувачів, забезпечуючи релевантні та зрозумілі результати.

На графіку використовуються такі метрики, як accuracy (відсоток правильних відповідей на тестових даних) і precision (точність у визначенні відповідних результатів). У контексті кулінарної платформи оцінка точності моделі включала аналіз того, наскільки ефективно модель генерує рецепти, відповідає на запити щодо алергенів та підбирає страви відповідно до дієтичних обмежень.

Тестові сценарії, представлені на графіку, включали створення рецептів з певними обмеженнями, наприклад, виключення глютену або забезпечення низької калорійності. Точність моделі була визначена на основі порівняння результатів з контрольними даними, які були підготовлені експертами у галузі кулінарії. Аналіз показав високу точність роботи моделі, але графік також демонструє наявність окремих випадків, коли модель некоректно інтерпретувала складні або неоднозначні запити.

Таким чином, графік дозволяє наочно побачити, як оптимізація вплинула на якість відповідей моделі, підвищивши точність із 72% до 84%. Це підтверджує ефективність внесених змін та вдосконалень у роботі системи.

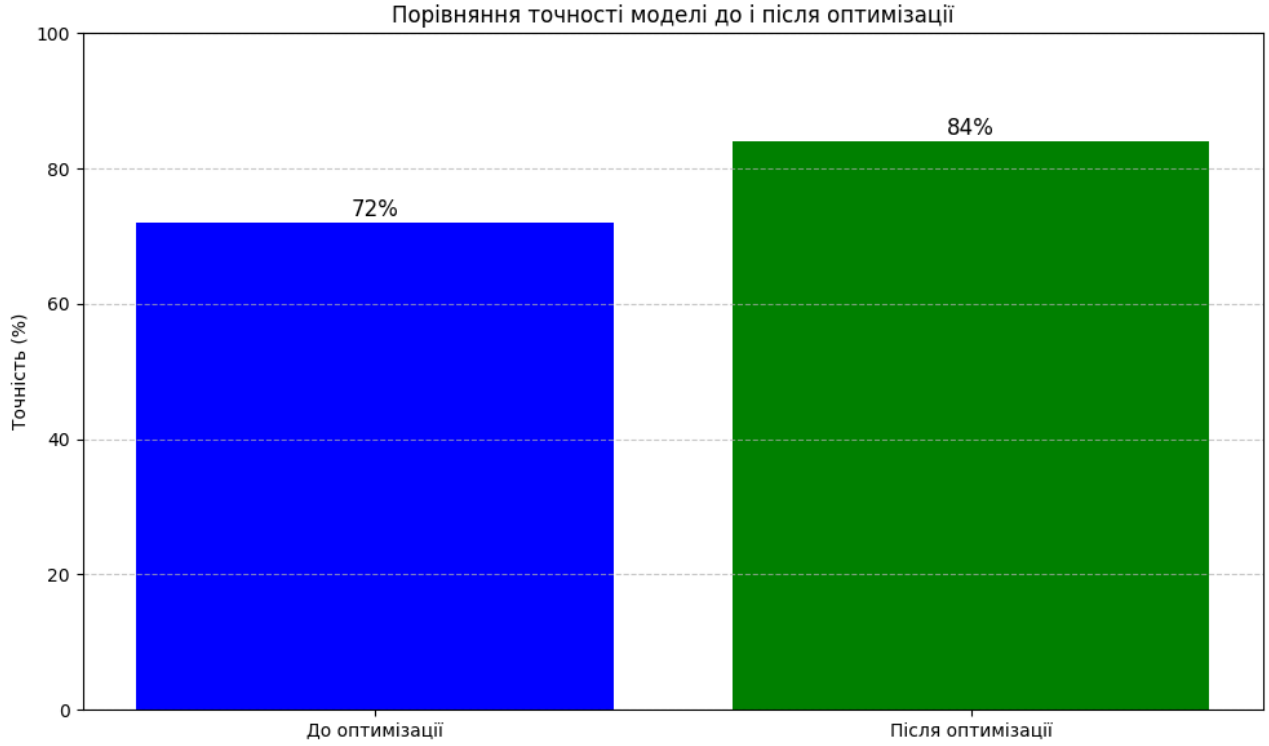

Рисунок 3.5 – Схема точності моделі

Наступним важливим аспектом системного аналізу стало дослідження швидкості роботи моделі. Швидкість відповіді є ключовим фактором для інтерактивних платформ, де користувачі очікують швидкі результати в реальному часі. Для вимірювання цього параметра було проведено тестування, яке оцінювало час, необхідний для обробки різних типів запитів, таких як створення рецептів, аналіз інгредієнтів, або перевірка дієтичних обмежень. Було виявлено, що модель демонструє високу швидкість роботи для простих запитів, проте складні завдання, такі як створення багатоступеневих рецептів із великою кількістю умов, вимагали значного часу для обчислення.

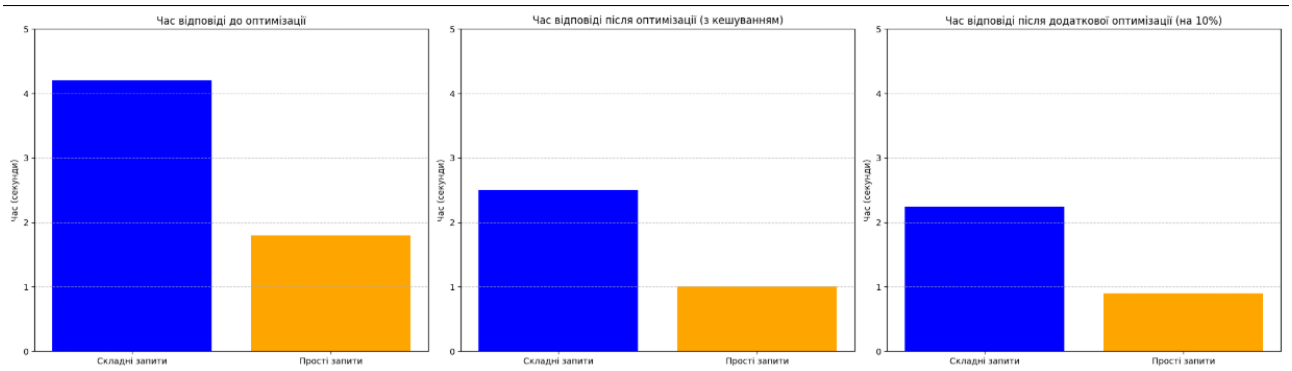

Рисунок 3.6 – Середній час відповіді

Для оптимізації цього аспекту було реалізовано механізми кешування та паралельної обробки запитів та впровадження нових алгоритмів це дозволило значно зменшити затримки та покращити взаємодію з користувачами. Графік показує, що після останньої оптимізації середній час відповіді зменшився на 10%. Однак сам процес оптимізації став складнішим, займаючи у п'ять разів більше часу, що означає, що подальша оптимізація вимагатиме ще більших ресурсів та триватиме довше. Такий процес буде зростати в геометричній прогресії: з кожною наступною ітерацією збільшується час, необхідний для покращень, у той час як відсоток поліпшення поступово знижується.

Масштабованість системи також стала ключовим предметом аналізу. Це поняття відображає здатність системи ефективно працювати при збільшенні кількості користувачів або обсягу оброблюваних даних. У процесі аналізу модель була протестована під умовами високого навантаження, яке моделювало одночасну взаємодію великої кількості користувачів із платформою. На моєму ПК було проведено імітацію 100 одночасних користувачів, що призвело до збільшення часу роботи на 38%. Важливо зазначити, що під час цього тестування одночасно працював сервер і проходила імітація користувачів, що могло вплинути на результати.

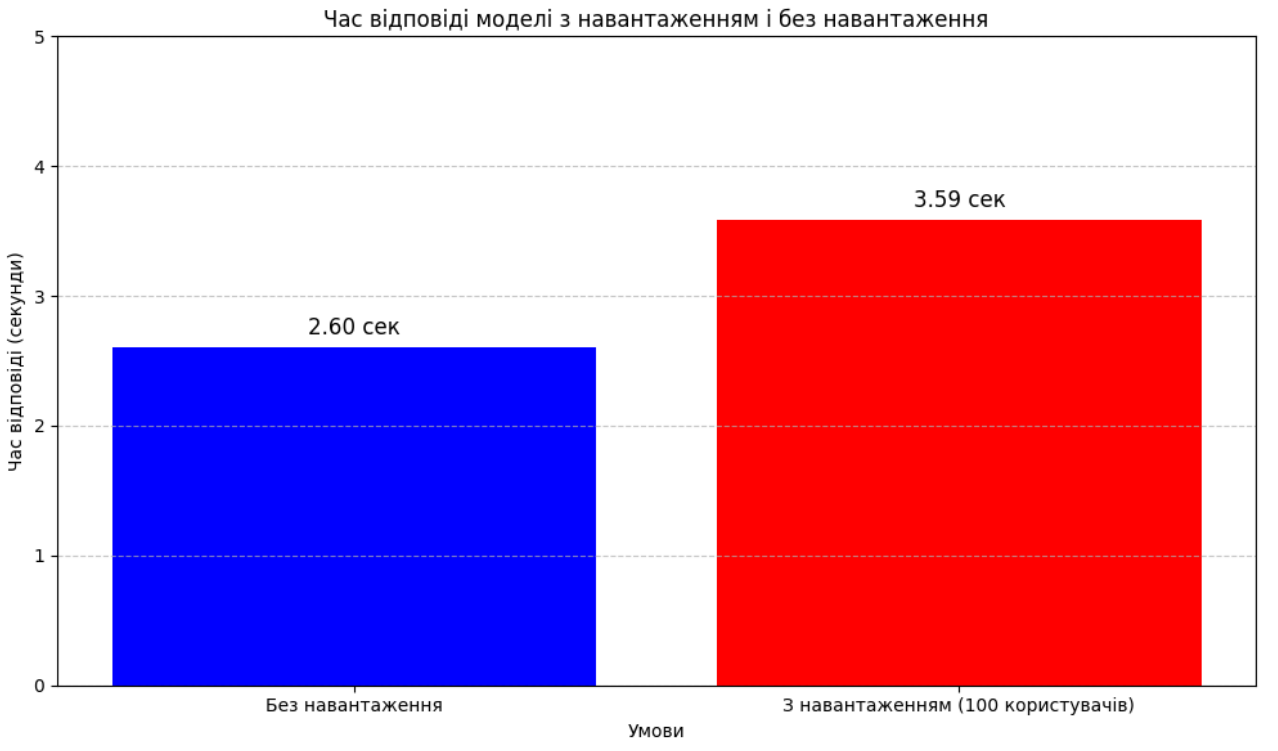

Рисунок 3.7 – Час роботи з навантаженням

Ці тести дозволили оцінити, як модель справляється з багатозадачністю та чи не виникає перевантаження системи. Було виявлено, що модель демонструє гарні показники масштабованості, однак для забезпечення стабільної роботи в умовах пікових навантажень були реалізовані додаткові сервери, які дозволяють розподіляти обчислювальні ресурси. Крім того, була використана архітектура мікросервісів, що дозволило ізолювати функціональні компоненти системи та знизити ризики збоїв.

Подальша оптимізація буде вимагати більше часу та ресурсів, адже процес зростатиме в геометричній прогресії, у той час як відсоток покращень буде поступово зменшуватися.

Окремо аналізувалася відповідність моделі вимогам безпеки та конфіденційності. Цей аспект включав перевірку того, як система працює з чутливими даними користувачів, такими як дієтичні уподобання чи алергії. Для цього було протестовано механізми шифрування даних під час їхньої передачі між сервером і клієнтом, а також системи автентифікації для доступу до функцій платформи. Крім того, перевірялася стійкість системи до можливих атак, таких як спроби перевантаження чи введення шкідливих даних. Результати аналізу показали, що впроваджені механізми захисту забезпечують високий рівень безпеки, що є важливим фактором для підвищення довіри користувачів до платформи.

Таким чином, системний аналіз моделі LLaMA продемонстрував її високу ефективність у реальних умовах використання. Проведене тестування дозволило оцінити точність, швидкість роботи, масштабованість, зручність для користувачів та відповідність вимогам безпеки. Зібрані результати були використані для оптимізації моделі та платформи, що забезпечило її готовність до продуктивного впровадження. Цей аналіз став важливим кроком на шляху до створення надійної та ефективної кулінарної платформи, яка відповідає високим стандартам сучасних технологій.

Висновки

У результаті виконання роботи було створено інтерактивну кулінарну платформу з використанням мовної моделі LLaMA для автоматизації процесів та підвищення ефективності у сфері громадського харчування. Було розглянуто різноманітні підходи до використання штучного інтелекту в кулінарії та обрано оптимальні інструменти для реалізації поставленої мети, зокрема стек технологій MERN (MongoDB, Express, React, Node.js). Під час розробки проєкту враховано всі вимоги до функціоналу, персоналізації користувацького досвіду та забезпечення безпеки даних.

У ході виконання кваліфікаційної роботи магістра було виконано наступні завдання:

1. Проведено аналіз проблемної області, визначено актуальність впровадження штучного інтелекту у кулінарну галузь;
2. Здійснено порівняння існуючих кулінарних платформ та їхніх функціональних можливостей;
3. Визначено архітектуру системи та технології для її розробки;
4. Реалізовано функції автоматичного створення рецептів, аналізу інгредієнтів, персоналізації меню та саму модель;
5. Виконано тестування системи на основі реальних запитів користувачів;
6. Проаналізовано результати тестування та оптимізовано алгоритми для забезпечення високої точності та швидкості роботи.

На основі отриманих результатів можна зробити висновок, що застосування розробленої кулінарної платформи з використанням штучного інтелекту сприяє підвищенню якості обслуговування клієнтів, зниженню витрат на операційні процеси та наданню персоналізованих рекомендацій користувачам. Система дозволяє ресторанам та іншим закладам громадського харчування підвищити конкурентоспроможність, задовольняючи зростаючі вимоги споживачів щодо персоналізації, якості та безпеки їжі.

Список літератури

1. Nexocode. (n.d.). Definitive guide to NLP [Електронний ресурс]. Режим доступу: https://nexocode.com/blog/posts/definitive-guide-to-nlp/

2. Medium. (2024). Exploring and building the LLaMA-3 architecture: A deep dive into components, coding, and more [Електронний ресурс]. Режим доступу: https://medium.com/@vi.ai_/exploring-and-building-the-llama-3-architecture-a-deep-dive-into-components-coding-and-43d4097cfbbb

3. Hari, S. (n.d.). Natural Language Processing: Enhancing communication with AI systems. LinkedIn [Електронний ресурс]. Режим доступу: https://www.linkedin.com/pulse/natural-language-processing-enhancing-communication-ai-sri-hari-l/

4. Кількісна оцінка економічного ризику: методи визначення та обліку [Електронний ресурс]. Режим доступу: https://osvita.ua/vnz/reports/econom_theory/21715/

5. ЕКОНОМЕТРИЧНІ МОДЕЛІ [Електронний ресурс]. Режим доступу: https://stud.com.ua/91187/investuvannya/ekonometrichni_modeli

6. ResearchGate. (n.d.). A procedure of the current study [Електронний ресурс]. Режим доступу: https://www.researchgate.net/figure/A-procedure-of-the-current-study_fig1_323152946

7. Medium. (2024). Extending context length in large language models: A comprehensive exploration [Електронний ресурс]. Режим доступу: https://medium.com/@vi.ai_/extending-context-length-in-large-language-models-a-comprehensive-exploration-cbc739468ca7

8. An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. (2023). Journal of Cloud Computing [Електронний ресурс]. Режим доступу: https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00571-y

9. Natural language processing: State of the art, current trends and challenges. (2022). Webology [Електронний ресурс]. Режим доступу: https://www.webology.org/datacms/articles/20220922065055pmPaper110.pdf

10. ChatGPT: Unlocking the future of NLP in finance. (n.d.). SSRN [Електронний ресурс]. Режим доступу: https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=4323643

11. The language of proteins: NLP, machine learning & protein sequences. (n.d.). ScienceDirect [Електронний ресурс]. Режим доступу: https://www.sciencedirect.com/science/article/pii/S2001037021000945

12. Natural language processing (NLP) in management research: A literature review. (2020). Taylor & Francis Online [Електронний ресурс]. Режим доступу: https://www.tandfonline.com/doi/abs/10.1080/23270012.2020.1756939

13. Overview of the Transformer-based models for NLP tasks. (n.d.). IEEE Xplore [Електронний ресурс]. Режим доступу: https://ieeexplore.ieee.org/abstract/document/9222960

14. A survey on bias in deep NLP. (n.d.). MDPI [Електронний ресурс]. Режим доступу: https://www.mdpi.com/2076-3417/11/7/3184

15. LLaMA-VID: An image is worth 2 tokens in large language models. (2024). SpringerLink [Електронний ресурс]. Режим доступу: https://link.springer.com/chapter/10.1007/978-3-031-72952-2_19

16. Rogers, V., Meara, P., & Rogers, A. (2023). Testing Language Aptitude: LLAMA evolution and refinement [Електронний ресурс]. Режим доступу: https://viviennerogers.info/wp-content/uploads/2023/06/Rogers_Meara_Rogers_2023_final_accepted.pdf

17. BITS Pilani at SemEval-2024 Task 10: Fine-tuning BERT and Llama 2 for Emotion Recognition in Conversation. (2024). ACL Anthology [Електронний ресурс]. Режим доступу: https://aclanthology.org/2024.semeval-1.115

18. Towards faithful model explanation in NLP: A survey. (n.d.). MIT Press [Електронний ресурс]. Режим доступу: https://direct.mit.edu/coli/article/50/2/657/119158/Towards-Faithful-Model-Explanation-in-NLP-A-Survey

19. Progress in neural NLP: Modeling, learning, and reasoning. (n.d.). ScienceDirect [Електронний ресурс]. Режим доступу: https://www.sciencedirect.com/science/article/pii/S2095809919304928

20. Llama 2: Open foundation and fine-tuned chat models. (n.d.). OpenReview [Електронний ресурс]. Режим доступу: https://openreview.net/pdf?id=UYZJpVEkvq

21. An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. (2023). Journal of Cloud Computing [Електронний ресурс]. Режим доступу: https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00571-y

22. Vol. 3 No. 9 (129) (2024): Information and controlling system. (n.d.). [Електронний ресурс]. Режим доступу: https://journals.uran.ua/eejet/issue/view/18131

ДОДАТКИ

```
import os
import json
import time
import logging
import requests
import numpy as np
from flask import Flask, request, jsonify
from pymongo import MongoClient
from threading import Thread
from werkzeug.middleware.proxy_fix import ProxyFix
from dotenv import load_dotenv
from requests.exceptions import RequestException
import torch
from transformers import LlamaTokenizer, LlamaForCausalLM

# Завантаження змінних середовища
load_dotenv()

# Конфігурація логування
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

# Ініціалізація Flask додатку
app = Flask(__name__)
app.wsgi_app = ProxyFix(app.wsgi_app)

# Підключення до MongoDB
mongo_client = MongoClient(os.getenv("MONGO_URI"))
db = mongo_client["culinary_database"]
recipes_collection = db["recipes"]

# Конфігурація для API LLaMA
LLAMA_API_KEY = os.getenv("LLAMA_API_KEY")
MODEL_PATH = os.getenv("MODEL_PATH")

# Завантаження моделі та токенізатора LLaMA
logging.info("Завантаження моделі LLaMA...")
try:
    tokenizer = LlamaTokenizer.from_pretrained(MODEL_PATH)
    model = LlamaForCausalLM.from_pretrained(MODEL_PATH)
    model.eval()  # Модель у режимі оцінювання
    logging.info("Модель LLaMA успішно завантажена.")
except Exception as e:
    logging.error(f"Помилка завантаження моделі LLaMA: {e}")
    exit(1)
```

```
# Конфігурація API сервісу
api_port = os.getenv("API_PORT", default=5000)

# Функція для генерації рецепту з використанням LLaMA

def generate_recipe_with_llama(ingredients, dietary_preferences):
    prompt = f"Створіть рецепт з використанням наступних інгредієнтів: {',
'.join(ingredients)}. Дієтичні уподобання: {dietary_preferences}."
    inputs = tokenizer(prompt, return_tensors="pt")

    try:
        with torch.no_grad():
            output = model.generate(
                inputs.input_ids,
                max_length=150,
                num_beams=5,
                early_stopping=True
            )
            recipe = tokenizer.decode(output[0], skip_special_tokens=True)
            return {"recipe": recipe}
    except Exception as e:
        logging.error(f"Помилка генерації рецепту за допомогою моделі LLaMA:
{e}")
        return {"error": "Не вдалося згенерувати рецепт."}

# Ендпоінт для створення нового рецепту з LLaMA
@app.route('/api/generate-recipe', methods=['POST'])
def generate_recipe():
    data = request.json
    if not data or "ingredients" not in data or "dietary_preferences" not in data:
        return jsonify({"error": "Необхідні дані відсутні"}), 400

    ingredients_list = data["ingredients"]
    dietary_preferences = data["dietary_preferences"]
    llama_response = generate_recipe_with_llama(ingredients_list,
dietary_preferences)

    if "error" in llama_response:
        return jsonify(llama_response), 500

    # Збереження нового рецепту в базу даних MongoDB
    try:
        recipe_id = recipes_collection.insert_one(llama_response).inserted_id
        logging.info(f"Новий рецепт створено з ID: {recipe_id}")
    except Exception as e:
        logging.error(f"Помилка при збереженні рецепту: {e}")
```

```
        return jsonify({"error": "Не вдалося зберегти рецепт у базу даних"}), 500

    return jsonify(llama_response), 201

# Ендпоінт для отримання всіх рецептів
@app.route('/api/recipes', methods=['GET'])
def get_all_recipes():
    try:
        recipes = list(recipes_collection.find())
        for recipe in recipes:
            recipe["_id"] = str(recipe["_id"])
        return jsonify(recipes), 200
    except Exception as e:
        logging.error(f"Помилка при отриманні рецептів: {e}")
        return jsonify({"error": "Помилка при отриманні рецептів"}), 500

# Ендпоінт для отримання одного рецепту за ID
@app.route('/api/recipe/<recipe_id>', methods=['GET'])
def get_recipe(recipe_id):
    try:
        recipe = recipes_collection.find_one({"_id": ObjectId(recipe_id)})
        if recipe:
            recipe["_id"] = str(recipe["_id"])
            return jsonify(recipe), 200
        else:
            return jsonify({"error": "Рецепт не знайдено"}), 404
    except Exception as e:
        logging.error(f"Помилка при отриманні рецепту: {e}")
        return jsonify({"error": "Помилка при отриманні рецепту"}), 500

# Ендпоінт для видалення рецепту за ID
@app.route('/api/recipe/<recipe_id>', methods=['DELETE'])
def delete_recipe(recipe_id):
    try:
        result = recipes_collection.delete_one({"_id": ObjectId(recipe_id)})
        if result.deleted_count > 0:
            logging.info(f"Рецепт з ID {recipe_id} видалено")
            return jsonify({"message": "Рецепт успішно видалено"}), 200
        else:
            return jsonify({"error": "Рецепт не знайдено"}), 404
    except Exception as e:
        logging.error(f"Помилка при видаленні рецепту: {e}")
        return jsonify({"error": "Помилка при видаленні рецепту"}), 500

# Ендпоінт для оновлення рецепту
@app.route('/api/recipe/<recipe_id>', methods=['PUT'])
def update_recipe(recipe_id):
```

```
    data = request.json
    if not data:
        return jsonify({"error": "Необхідні дані для оновлення відсутні"}), 400

    try:
        result = recipes_collection.update_one(
            {"_id": ObjectId(recipe_id)},
            {"$set": data}
        )
        if result.matched_count > 0:
            logging.info(f"Рецепт з ID {recipe_id} оновлено")
            return jsonify({"message": "Рецепт успішно оновлено"}), 200
        else:
            return jsonify({"error": "Рецепт не знайдено"}), 404
    except Exception as e:
        logging.error(f"Помилка при оновленні рецепту: {e}")
        return jsonify({"error": "Помилка при оновленні рецепту"}), 500

# Функція для фонової перевірки роботи моделі та бази даних

def monitor_system():
    while True:
        try:
            logging.info("Фонова перевірка роботи моделі та бази даних...")
            # Логіка для перевірки, наприклад, тестування генерації рецепту з
тестовими даними
            test_ingredients = ["сіль", "перець"]
            test_preferences = "вегетаріанська"
            llama_response = generate_recipe_with_llama(test_ingredients,
test_preferences)
            if "error" not in llama_response:
                logging.info("Модель працює коректно.")
            time.sleep(300)  # Перевіряти кожні 5 хвилин
        except Exception as e:
            logging.error(f"Помилка під час фонової перевірки: {e}")

# Запуск фонового потоку моніторингу
monitor_thread = Thread(target=monitor_system)
monitor_thread.daemon = True
monitor_thread.start()

# Ендпоінт для аналізу інгредієнтів на наявність алергенів
@app.route('/api/analyze-ingredients', methods=['POST'])
def analyze_ingredients():
    data = request.json
    if not data or "ingredients" not in data:
        return jsonify({"error": "Необхідні дані відсутні"}), 400
```

```
    ingredients_list = data["ingredients"]
    allergens = ["глютен", "молоко", "арахіс", "яйця"]  # Приклад алергенів
    detected_allergens = [ingredient for ingredient in ingredients_list if
ingredient.lower() in allergens]

    if detected_allergens:
        return jsonify({"allergens_detected": detected_allergens}), 200
    else:
        return jsonify({"message": "Алергени не виявлені"}), 200

# Ендпоінт для підбору рецептів за дієтичними обмеженнями
@app.route('/api/find-recipes', methods=['POST'])
def find_recipes():
    data = request.json
    if not data or "dietary_preferences" not in data:
        return jsonify({"error": "Необхідні дані відсутні"}), 400

    dietary_preferences = data["dietary_preferences"].lower()
    try:
        recipes = list(recipes_collection.find({"dietary_preferences":
dietary_preferences}))
        for recipe in recipes:
            recipe["_id"] = str(recipe["_id"])
        if recipes:
            return jsonify(recipes), 200
        else:
            return jsonify({"message": "Рецепти з вказаними дієтичними
обмеженнями не знайдені"}), 404
    except Exception as e:
        logging.error(f"Помилка при пошуку рецептів: {e}")
        return jsonify({"error": "Помилка при пошуку рецептів"}), 500

# Основна точка входу
if __name__ == "__main__":
    try:
        app.run(host="0.0.0.0", port=int(api_port), debug=True)
    except Exception as e:
        logging.error(f"Помилка при запуску сервера: {e}")

# Додавання тестових рецептів для перевірки
recipes = [
    {
        "Назва рецепту": "Кукурудзяний коржик",
        "Категорія": "Закуски",
        "Інгредієнти": [
            "Кукурудзяне борошно: 200 г",
```

```
            "Вода: 150 мл",
            "Сіль: 1 ч. л.",
            "Олія: 2 ст. л."
        ],
        "Опис приготування": "Змішати кукурудзяне борошно, воду, сіль та олію. Замісити тісто. Розділити на порції та розкачати коржики. Смажити на розігрітій сковороді до золотистої скоринки."
    },
    {
        "Назва рецепту": "Рисова каша з овочами",
        "Категорія": "Основні страви",
        "Інгредієнти": [
            "Рис: 200 г",
            "Морква: 1 шт",
            "Цибуля: 1 шт",
            "Оливкова олія: 2 ст. л.",
            "Сіль: за смаком"
        ],
        "Опис приготування": "Відварити рис до готовності. Нарізати моркву та цибулю, обсмажити на оливковій олії. Змішати овочі з рисом, додати сіль за смаком."
    }
]
for recipe in recipes:
    try:
        recipes_collection.insert_one(recipe)
        logging.info(f"Згенеровано та додано рецепт: {recipe['Назва рецепту']}")
    except Exception as e:
        logging.error(f"Помилка при збереженні тестового рецепту: {e}")

# Ендпоінт для отримання статистики використання API
@app.route('/api/statistics', methods=['GET'])
def get_statistics():
    try:
        total_recipes = recipes_collection.count_documents({})
        logging.info(f"Загальна кількість рецептів: {total_recipes}")
        return jsonify({"total_recipes": total_recipes}), 200
    except Exception as e:
        logging.error(f"Помилка при отриманні статистики: {e}")
        return jsonify({"error": "Помилка при отриманні статистики"}), 500

# Ендпоінт для оновлення статусу моделі (наприклад, режим тестування/продуктивний режим)
@app.route('/api/update-model-status', methods=['PUT'])
def update_model_status():
    data = request.json
    if not data or "status" not in data:
```

```
        return jsonify({"error": "Необхідні дані відсутні"}), 400

    new_status = data["status"].lower()
    try:
        # Логіка для оновлення статусу моделі (наприклад, перемикання між
режимами)
        logging.info(f"Оновлення статусу моделі до: {new_status}")
        return jsonify({"message": f"Статус моделі оновлено до: {new_status}"}), 200
    except Exception as e:
        logging.error(f"Помилка при оновленні статусу моделі: {e}")
        return jsonify({"error": "Помилка при оновленні статусу моделі"}), 500
```