

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА
на тему:**

**ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОЇ ВІДПОВІДІ
НА КОРИСТУВАЦЬКІ ЗАПИТИ ЗА ДОПОМОГОЮ МОВНИХ МОДЕЛЕЙ**

Здобувача вищої освіти другого (магістерського) рівня,
II курсу, групи КМІ-23-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»



(підпис)

Кондратенка Андрія
Володимировича

(прізвище, ім'я, по батькові)

Науковий керівник



(підпис)

Ратушняк Тетяна
Володимирівна,
к.фіз-мат.н, доцент

(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми



(підпис)

Горбовий Артур Юліанович,
д.т.н, професор

(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «10» січня 2025 №11

Завідувач кафедри



(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент

(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Ірпінь – 2025

Реєстрація № 25

27.01.2025
(дата)


(підпис відповідальної особи)

Результат перевірки

до захисту
(до захисту, на доопрацювання)

27.01.2025 
(дата) (підпис)

Результат захисту

4 (добре)
(оцінка)

06.02.2025 
(дата) (підпис)

Голова комісії:


(підпис)

В.М. Франчук

Члени комісії:


(підпис)

А.Ю. Горбовий


(підпис)

М.М. Філоненко


(підпис)

І.Д. Погореловська

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	25		<div>06.02.2025</div> <div>Франчук В.М. </div> <div>Горбовий А.Ю. </div> <div>Філоненко М.М. </div> <div>Погореловська І.Д. </div> <div>Суховерховська Л.П. </div>
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	42		

Оцінювання кваліфікаційної роботи науковим керівником - 41 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
83	B	4 (добре)	06.02.2025	<div>Франчук В.М. </div> <div>Горбовий А.Ю. </div> <div>Філоненко М.М. </div> <div>Погореловська І.Д. </div> <div>Суховерховська Л.П. </div>

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую

Завідувач кафедри  Філоненко М.М.

«04» 11 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Кондратенку Андрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Додаток для автоматизованої відповіді на користувацькі запити за допомогою мовних моделей
Затверджена наказом ректора від «04» листопада 2024 року № 1680
2. Строк здачі закінченої роботи «07» січня 2025 року
3. Мета роботи розробити веб-додаток з інтегрованою мовною моделлю
Об'єкт дослідження: штучний інтелект для автоматизованої обробки природномовних запитів
Предмет дослідження: технологічні, програмні та методичні аспекти інтеграції мовної моделі GPT у веб-додаток
Орієнтовна база дослідження: ТОВ «УКРСПЕЦКОНСАЛТИНГ»
4. Вихідні дані до роботи: перелік користувацьких запитів, параметри тарифікації й лімітів GPT-API, технічна документація для інтеграції мовних моделей у веб-додаток
5. Перелік графічного матеріалу: Методи і завданнями дослідження, Схема взаємодії з мовною моделлю, Джерела інформації для аналізу бази дослідження, Основні вимоги та критерії якості додатку, Приклад політик кешування, Ключові показники роботи прототипу за період тестування, Головні проектні рішення, Порівняння базових та оптимізованих показників функціоналу
6. Зміст кваліфікаційної роботи
Розділ 1. Теоретико-методологічні основи створення додатку для автоматизованої відповіді на користувацькі запити
 - 1.1. Теоретичні основи використання мовних моделей у веб-додатках
 - 1.2. Методологічні основи дослідження
 - 1.3. Концепція побудови та впровадження веб-додаткуРозділ 2. Аналіз та реалізація застосунку на основі Next.js та GPT
 - 2.1. Аналітичне дослідження бази та вимог до додатку
 - 2.2. Реалізація та початкове тестування прототипу
 - 2.3. Аналіз результатівРозділ 3. Проектні рішення та перспективи впровадження
 - 3.1. Проектні рішення та напрями вдосконалення додатку
 - 3.2. Перспективи впровадження та економічний вимір запропонованих рішень

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2023	13.12.2023
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	27.05.2024	01.06.2024
3	II розділ (аналіз експериментальної частини про методики досліджень)	02.09.2024	09.09.2024
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	18.11.2024	23.11.2024
5	Висновки	10.12.2024	13.12.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	23.12.2024	25.12.2024
7	Отримання відгуку керівника	28.01.2025	28.01.2025
8	Отримання зовнішньої рецензії	31.01.2025	31.01.2025

8. Дата видачі завдання «04» листопада 2024 року

9. Керівник кваліфікаційної роботи


(підпис)

Ратушняк Т.В
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Горбовий А.К
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Кондратенко
(прізвище, ініціали)

АНОТАЦІЯ

Додаток для автоматизованої відповіді
на користувацькі запити за допомогою мовних моделей

Кваліфікаційна робота присвячена розробці та реалізації веб-додатка з функціоналом автоматизованої відповіді на користувацькі запити за допомогою мовної моделі.

Актуальність теми обумовлена цифровізацією всіх сфер сучасного суспільства та зростанням попиту на доступні сервіси консультаційного характеру.

Мета роботи полягає у розробці веб-додатку з інтегрованою мовною моделлю.

Для досягнення мети визначено такі завдання: оптимізувати витрати на GPT, упровадити кешування повторних запитів, забезпечити контент-фільтр і модерацію, а також покращити «пам'ять» бота за умови тривалих діалогів значного обсягу.

Об'єкт дослідження: штучний інтелект для автоматизованої обробки природномовних запитів.

Предмет дослідження: технологічні, програмні та методичні аспекти інтеграції мовної моделі GPT у веб-додаток

Наукова значимість полягає у розробці масштабованого чат-бота, який надає користувачу якісні відповіді в режимі реального часу.

Практична значимість: додаток дозволить виконувати обробку природної мови для консультації користувача у різних галузях людської діяльності, зокрема в освітній діяльності і бізнесі.

За результатами дослідження розроблено рекомендації щодо оптимальної взаємодії з GPT-API, довготривалого кешування повторюваних запитів і коректного механізму фільтрації контенту. Це дає змогу знизити фінансові затрати на виклики мовної моделі та підвищити швидкодію чат-бота.

Кваліфікаційна робота містить 50 сторінок, 8 таблиць, 1 рисунок, список використаних джерел із 46 найменувань, додатки.

Ключові слова: вебдодаток, Next.js, GPT, масштабування, кешування, чат-бот, контент-фільтр, релевантність.

Рік виконання кваліфікаційної роботи — 2024.

Рік захисту кваліфікаційної роботи — 2025.

ABSTRACT

Application for automated response to user requests using language models

The goal of the work is to create a scalable chatbot that provides high-quality answers in real time. To achieve the goal, the following tasks have been identified: optimize GPT costs, implement caching of repeated requests, provide content filter and moderation, as well as improve the bot's "memory" during long dialogues.

To achieve the goal, the following tasks have been identified: optimize GPT costs, implement caching of repeated requests, provide content filter and moderation, as well as improve the bot's "memory" during long dialogues.

The subject of the study is methods for integrating large language models (GPT) into the web environment.

The object of research is the process of providing automated text responses to users.

The basis of the study is a prototype of a web application deployed on the Vercel platform.

Based on the results of the study, recommendations were developed for optimal interaction with GPT-API, long-term caching of repetitive requests, and a correct content filtering mechanism. This allows you to reduce the financial costs of calling the language model and increase the performance of the chatbot. The results obtained can be used to further improve or scale chatbot systems in business and educational institutions.

The qualification work contains 50 pages, 8 tables, 1 figure, a list of used sources from 46 titles, as well as appendices.

KEYWORDS: Next.js, GPT, scaling, caching, chatbot, content filter, relevance.

Year of qualification work — 2024.

Year of defense of qualification work — 2025.

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ СТВОРЕННЯ ВЕБДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОЇ ВІДПОВІДІ НА КОРИСТУВАЦЬКІ ЗАПИТИ	6
1.1. Теоретичні основи використання мовних моделей у вебдодатках	6
1.2. Методологічні основи дослідження.....	13
1.3. Концепція побудови та впровадження веб-додатку.....	16
РОЗДІЛ 2. РЕАЛІЗАЦІЯ ТА АНАЛІЗ ДОДАТКУ НА ОСНОВІ NEXT.JS ТА GPT	20
2.1. Аналітичне дослідження бази та вимог до системи.....	20
2.2. Реалізація та початкове тестування прототипу	29
2.3. Аналіз результатів.....	33
РОЗДІЛ 3. ПРОЄКТНІ РІШЕННЯ ТА ПЕРСПЕКТИВИ ВПРОВАДЖЕННЯ....	39
3.1. Проєктні рішення та напрями вдосконалення додатку	39
3.2. Перспективи впровадження та економічний вимір запропонованих рішень.....	41
ВИСНОВКИ	45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	47
ДОДАТКИ.....	51

ВСТУП

Сучасний стрімкий розвиток інформаційних технологій зумовлює все ширше впровадження систем автоматизації обробки текстових запитів та генерації відповідей у режимі реального часу. При цьому велику популярність здобули великі мовні моделі (зокрема, родина GPT), які дають змогу підвищити ефективність взаємодії користувачів із різноманітними онлайн-сервісами. Проте, попри суттєві здобутки в дослідженнях таких моделей, залишається відкритим питання про комплексний підхід до проєктування та розробки веб-додатків, що використовують ці моделі як базову технологію для швидких та релевантних відповідей на користувацькі запити.

У цій кваліфікаційній роботі передбачається створення застосунку на базі Next.js із використанням мовної моделі GPT, який буде здатний обробляти та надавати автоматизовані відповіді на запити користувачів у різних галузях.

Актуальність обраної теми зумовлена зростаючою потребою у високоякісних системах взаємодії людини з комп'ютером, що працюють у режимі реального часу та розуміють природну мову. Розвиток хмарних сервісів, доступ до потужних API та відкритість інструментів для інтеграції мовних моделей у веб-застосунки відкривають нові можливості для створення інтелектуальних помічників, чат-ботів, систем підтримки клієнтів тощо. При цьому не менш важливими залишаються питання безпеки, етичного використання таких систем та забезпечення коректності відповідей. Отже, дослідження оптимальних методів інтеграції мовних моделей типу GPT у застосунки на основі Next.js має як наукову, так і прикладну цінність.

Проблема використання мовних моделей для автоматизованих відповідей вивчається багатьма вітчизняними та зарубіжними дослідниками. Зокрема, широко відомими є розробки компанії OpenAI (GPT-2, GPT-3, GPT-4), Google (BERT, LaMDA) та інших провідних AI-лабораторій. У більшості наукових публікацій акцент зроблено на алгоритмічних і лінгвістичних аспектах побудови мовних моделей та їхньої здатності до генерації контекстно релевантних

відповідей. Водночас, питання, пов'язані з проєктуванням і розробкою масштабованих веб-додатків, які інкапсулюють можливості таких моделей, потребують додаткових досліджень.

Метою кваліфікаційної роботи є розробка та впровадження веб-застосунку на базі Next.js з інтегрованою мовною моделлю GPT, здатного автоматично генерувати відповіді на користувацькі запити у різних контекстах.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Систематизувати наукові та практичні підходи до використання мовних моделей у веб-застосунках.
2. Виявити особливості інтеграції мовної моделі GPT через API в архітектуру Next.js.
3. Розробити архітектурне рішення веб-застосунку, що забезпечує надійну роботу та масштабованість.
4. Здійснити експериментальне тестування роботи застосунку з реальними користувацькими запитами.
5. Оцінити ефективність та релевантність згенерованих відповідей, а також сформулювати рекомендації щодо покращення роботи системи.

Об'єкт дослідження – процес розробки та функціонування веб-додатків, що використовують штучний інтелект для автоматизованої обробки природномовних запитів.

Предмет дослідження – технологічні, програмні та методичні аспекти інтеграції та налаштування мовної моделі GPT у веб-застосунку на базі Next.js.

Для досягнення визначених завдань використовуються:

- Аналіз та синтез наукових джерел і публікацій з тематики мовних моделей та їх інтеграції.
- Проєктно-аналітичний метод для створення архітектури застосунку й оцінювання ризиків.
- Метод прототипування під час розробки початкових версій додатка з подальшим тестуванням.

- Емпіричні методи (тестування, експерименти) для оцінки точності, швидкодії та релевантності отриманих відповідей.

У рамках кваліфікаційної роботи база досліджень представлена програмним прототипом веб-застосунку, що розгортається в тестовому середовищі з використанням Next.js та API GPT (OpenAI). Крім того, у дослідженні використовуються демо-приклади взаємодії з користувачами, які моделюють реальні сценарії застосування (система онлайн-підтримки, освітні консультації тощо).

Результати дослідження можуть бути корисними для розробників, що впроваджують мовні моделі у свої продукти, а також для науковців, які вивчають способи оптимізації інтеграційних рішень. Запропоновані підходи та рекомендації можна застосовувати для створення розширених систем підтримки користувачів, онлайн-асистентів, інтерактивних навчальних платформ, що дає змогу підвищити ефективність та зручність взаємодії з ІТ-системами.

Попередні напрацювання та результати були представлені на студентських наукових семінарах та круглих столах із проблем впровадження штучного інтелекту в навчальний процес. За результатами обговорень отримано рекомендації щодо подальшого доопрацювання та вдосконалення застосунку, а також визначено перспективи розширення функціоналу системи.

Таким чином, розробка веб-застосунку з використанням сучасних мовних моделей типу GPT задовольняє потреби автоматизації обробки користувацьких запитів, що підтверджує актуальність досліджень, їх теоретичну й практичну цінність, а також вказує на перспективи подальшого наукового пошуку в цьому напрямі.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ СТВОРЕННЯ ВЕБДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОЇ ВІДПОВІДІ НА КОРИСТУВАЦЬКІ ЗАПИТИ

1.1. Теоретичні основи використання мовних моделей у веб-додатках

У сучасному інформаційному просторі обсяг текстових даних невинно зростає, що вимагає розробки інструментів та сервісів для швидкої та ефективної обробки інформації. З-поміж різних технологій, спрямованих на розуміння та генерацію природномовних текстів, найактивніше розвиваються мовні моделі на основі нейронних мереж. Їхня актуальність значно посилилася з появою великих глибинних трансформер-архітектур, зокрема, GPT (Generative Pre-trained Transformer), що є одним із найбільш успішних прикладів застосування сучасних підходів до обробки природної мови (NLP) (Radford et al., GPT) [10].

Перш ніж перейти до аналізу GPT, доцільно розглянути історичний розвиток та різновиди мовних моделей, які дозволили досягти поточних результатів у сфері NLP:

1. Статистичні (n-грамні) моделі.

На початкових етапах розвитку обчислювальної лінгвістики домінували статистичні методи. Класичним прикладом є n-gram-моделі, в яких імовірність появи слова розраховувалася на основі певної кількості попередніх слів (Jurafsky, Martin, 2019) [4]. Хоча ці моделі заклали фундамент для досліджень у галузі обробки природної мови, вони не могли ефективно фіксувати довготривалі залежності та швидко втрачали контекст.

2. Рекурентні нейронні мережі (RNN).

Розвиток глибинного навчання дав поштовх появі рекурентних нейронних мереж (RNN) та їхніх модифікацій — LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit) (Hochreiter, Schmidhuber, 1997) [5]. Ці архітектури дозволяли ефективніше враховувати контекст завдяки зворотним зв'язкам у часі, що покращувало результати на різноманітних задачах (Hochreiter, Schmidhuber, 1997). Утім, проблеми «зникання» або «вибуху» градієнта, а також обмежена

здатність запам'ятовувати довгі послідовності даних ставали на заваді їх широкого масштабованого застосування.

3. Трансформери (Transformer).

Проривом у сфері NLP стала поява у 2017 році трансформер-архітектури (Vaswani et al., 2017) [13]. Її центральною ідеєю є механізм уваги (attention), що дозволяє паралелізувати обчислення та враховувати взаємозв'язки між словами в межах цілого речення (або навіть більших контекстів). Це дало змогу позбутися більшості обмежень RNN та суттєво пришвидшити навчання великих моделей на масштабних корпусах даних.

На базі Transformer-архітектури було створено низку моделей:

- BERT (Bidirectional Encoder Representations from Transformers), орієнтований переважно на завдання розуміння тексту (Devlin et al., BERT, 2019) [11].
- GPT (Generative Pre-trained Transformer) — модель з переважним фокусом на генерації тексту (Radford et al., GPT, 2018) [10].
- T5 (Text-to-Text Transfer Transformer), яка уніфікує підхід до розв'язання різних NLP-завдань, переводячи всі їхні формулювання в єдину текстову форму (Raffel et al., T5, 2019) [12].

Одним із найвідоміших представників трансформер-орієнтованих моделей є GPT, ініційована компанією OpenAI. GPT — це авторегресивна модель, яка намагається передбачити наступне слово в послідовності, враховуючи контекст попередніх слів (Radford et al., GPT) [10]. Ключова особливість GPT — етап попереднього навчання (pre-training) на великих обсягах неструктурованих текстових даних, що дозволяє моделі набувати універсальних мовних знань.

- GPT-1 вже показав здатність до генерації тексту на базовому рівні, хоч і поступався іншим моделям у завданнях розуміння тексту (Radford et al., GPT-1, 2018) [10].
- GPT-2 продемонстрував набагато вищі показники й став об'єктом активних обговорень через загрозу поширення дезінформації та фейкових новин (OpenAI, GPT-2, 2019) [8].

- GPT-3, а згодом і GPT-3.5 зробили ще один якісний стрибок: вони мають мільярди параметрів і здатні відповідати на запитання, писати есе, кодувати тощо (Brown et al., GPT-3, 2020) [2].
- GPT-4 є вже мультимодальною моделлю зі ще більшим контекстним вікном, значно вищою «глибиною» розуміння та здатністю працювати з різними форматами даних (OpenAI, GPT-4, 2023) [9].

Збільшення кількості параметрів, удосконалення механізмів attention і використання масивних корпусів для навчання дали змогу GPT наблизитися до рівня, де генерований текст часто не відрізнити від людського.

Застосування GPT у веб-застосунках: переваги та обмеження

1. Переваги

- Якісна генерація тексту. GPT здатна формувати зв'язні, стилістично наближені до людських відповіді (Brown et al., GPT-3) [2]. Це має особливе значення для чат-ботів, систем підтримки клієнтів, освітніх платформ, де користувачі очікують швидкого та адекватного відгуку.
- Гнучкість адаптації. Модель можна «донавчити» (fine-tune) під конкретне завдання або ж налаштувати параметри запиту (prompt engineering), що відкриває широкі можливості застосування в різноманітних галузях — від технічного консультування до творчого письма.
- Масштабованість. Хмарні платформи (Azure OpenAI, AWS, Google Cloud) надають готові API для інтеграції GPT у веб-додатки (OpenAI, API) [8]. Це усуває бар'єри у вигляді необхідності локально розгортати потужні обчислювальні кластери, що було б недоцільно для невеликих команд розробників чи стартапів.

2. Обмеження

- Етичні ризики. Без додаткових фільтрів або систем модерації GPT може згенерувати неправильну, упереджену чи навіть образливу відповідь (Ковальчук, етичні аспекти) [7]. Це підкреслює важливість налаштування, контролю та фільтрації контенту.

- Великі обчислювальні ресурси. Навіть якщо навчання моделі виконується на серверах постачальника, інференс (тобто генерування відповідей) може потребувати значних потужностей при великому потоці запитів. Для бізнесу це часто означає суттєві витрати.
- Обрізаний контекст. Розмір контекстного вікна (window) у GPT обмежений (Xie et al., контекстне вікно) [15]. При перевищенні цього ліміту застосунок втрачає частину попередньої інформації, що ускладнює ведення довготривалої розмови з повним урахуванням усіх деталей (Xie et al., 2021). Загалом у науковій спільноті дедалі більшої уваги набуває питання комплексного використання великих мовних моделей:
- Американська школа (OpenAI, Google Brain, Microsoft Research) фокусується на збільшенні розміру моделей і роботі з гігантськими наборами даних, розглядаючи при цьому оптимізацію та «дистиляцію» (distillation) мовних моделей для більш економного інференсу (OpenAI, масштабованість) [8].
- Європейські дослідники звертають увагу на етичні та правові аспекти, а також на розробку моделей, орієнтованих на збереження конфіденційності та забезпечення прозорості прийняття рішень (Explainable AI, етика) [6].
- Українські науковці (Ковальчук, 2022 [7]; Семененко, 2023 [12]) проводять дослідження щодо адаптації трансформерів для української мови, зокрема вирішуючи питання з обмеженими відкритими корпусами даних та специфікою мовних зворотів. Накопичується досвід інтеграції GPT-подібних моделей у застосунки для державного та приватного секторів (онлайн-консультації громадян, освітні платформи, сервіси для бізнесу).

Ключовим науковим завданням у поточному контексті залишається розробка підходів до побудови комплексних рішень, які забезпечать коректну, захищену та масштабовану інтеграцію мовних моделей у веб-додатки.

Незважаючи на численні публікації з тематики GPT і значну кількість готових інструментів від постачальників хмарних сервісів, на практиці чимало веб-додатків стикається з обмеженнями та викликами, пов'язаними з:

- Організацією ефективної архітектури (балансування навантаження, безпечне зберігання API-ключів, управління конфігураціями) (Chen & Yu, архітектурні рішення) [3].
- Реалізацією логіки спілкування (забезпечення довготермінової пам'яті в діалогах, управління контекстом, фільтрація «небажаного» контенту).
- Оптимізацією витрат (розмежування безкоштовних і платних планів, використання кешування, механізмів зменшення числа звернень до API).

Поряд із технічними аспектами постає питання етики, адже великі мовні моделі можуть відображати упередження, закладені в дані, на яких вони навчалися (Bender et al., упередження, 2021) [6]. Це робить потрібним упровадження додаткових механізмів для виявлення та блокування потенційно шкідливих чи дискримінаційних висловлювань.

З огляду на вищезазначене, актуальність даного дослідження зумовлена необхідністю розробити прикладну методику та прототип веб-застосунку, що забезпечує автоматизовану відповідь на користувацькі запити із використанням GPT-моделей, водночас враховуючи питання масштабованості, надійності та етичного використання.

Важливо зазначити, що для більшості проєктів використання GPT у ролі «двигуна» генерації тексту — це лише частина задачі. Інтеграція такого механізму в сучасні веб-застосунки потребує зручної, гнучкої та надійної frontend-архітектури. Саме в цьому контексті значного поширення набули фреймворки на основі React, зокрема Next.js.

- Next.js пропонує серверний рендеринг, що дає змогу покращити продуктивність і SEO.
- Вбудовані механізми роутингу та API-роутів у Next.js спрощують структурування застосунку (документація Next.js) [14].
- Інтеграція з хмарними платформами (Vercel, AWS, Google Cloud) полегшує деплой і подальшу масштабованість.

У низці сучасних робіт (Wang et al., 2022; Chen et al., 2022) наголошується на ефективності поєднання Next.js із інструментами CI/CD (Continuous

Integration / Continuous Deployment), що відкриває шлях до швидкої розробки та частого оновлення застосунків, яким часто потрібне оперативне внесення змін (Chen & Yu, CI/CD) [3].

Таким чином, архітектура «Next.js + GPT» набуває популярності як універсальний варіант для створення чат-ботів, сервісів автоматизованої підтримки користувачів, платформ із генерацією контенту тощо.

Мовні моделі пройшли значний еволюційний шлях від статистичних програм до сучасних трансформерів, таких як GPT, BERT та T5. Цей розвиток дозволив суттєво підвищити якість обробки природної мови, забезпечуючи більш точне розуміння та генерацію тексту. Завдяки цьому, сучасні мовні моделі здатні ефективніше вирішувати складні завдання, пов'язані з аналізом тексту, перекладом, створенням контенту та іншими аспектами обробки природної мови, що робить їх незамінними інструментами в різних галузях.

Серед трансформер-орієнтованих моделей особливо популярною в задачах генерації тексту стала GPT. Ця модель завдяки попередньому навчанню на великих корпусах текстів забезпечує високу релевантність та гнучкість застосування. GPT демонструє здатність створювати зв'язні та контекстуально точні тексти, що робить її ідеальним вибором для автоматизації створення контенту, підтримки користувачів, створення чат-ботів та інших застосунків, де важлива генерація мови. Висока адаптивність GPT дозволяє використовувати її в різних сферах, від освіти до бізнесу, забезпечуючи ефективні та інноваційні рішення.

Основна проблема впровадження GPT у веб-додатки пов'язана не лише з технічними аспектами, такими як обчислювальні ресурси та архітектурні рішення, а й з етичною та правовою стороною. Виявлення упереджень у моделях, контроль змісту генерованих відповідей та забезпечення конфіденційності даних користувачів є критично важливими аспектами, які потребують ретельного врахування. Безпека та етичність використання мовних моделей визначають їхню прийнятність та довіру серед користувачів, тому важливо впроваджувати

механізми контролю та моніторингу для мінімізації ризиків та забезпечення відповідності законодавчим вимогам.

У науковій спільноті продовжуються активні дослідження щодо масштабування мовних моделей, оптимізації їх роботи та розробки ефективних інтеграційних патернів (design patterns). Ці зусилля спрямовані на покращення продуктивності моделей, зниження витрат на обчислення та спрощення процесу їх впровадження. Розробка нових методів навчання та оптимізації дозволяє створювати більш потужні та ефективні моделі, які можуть обробляти великі обсяги даних та забезпечувати високий рівень точності та швидкості роботи, що робить їх більш доступними та ефективними для широкого кола застосувань.

Використання фреймворку Next.js уможливлює створення гнучкої й масштабованої структури для веб-застосунків, що інтегрують GPT. Next.js надає зручні засоби для організації frontend- та backend-логіки, маршрутизації та деплою, що значно спрощує процес розробки та підтримки веб-додатків. Завдяки своїй архітектурі, Next.js забезпечує високу продуктивність та швидкість завантаження сторінок, а також підтримує серверний рендеринг, що покращує SEO та загальний користувацький досвід. Це дозволяє розробникам створювати надійні, масштабовані та легко підтримувані веб-застосунки, які ефективно інтегрують мовні моделі GPT для надання інтелектуальних та інноваційних функцій користувачам.

Отже, проведений у цьому підрозділі аналіз свідчить про необхідність комплексного дослідження технологічних, методичних і організаційних аспектів інтеграції GPT у веб-застосунки. У подальших підрозділах буде детальніше розглянуто методологічні основи застосування трансформер-орієнтованих моделей, а також описано загальні вимоги до системи, алгоритми, інструменти та відповідні приклади їх реалізації. Результатом стане концепція побудови прототипу застосунку на базі Next.js із можливістю надавати автоматизовані відповіді на користувацькі запити.

1.2. Методологічні основи дослідження

Формування методологічних і методичних засад є одним із ключових етапів будь-якого науково-практичного дослідження, пов'язаного зі створенням інноваційних систем. У нашому випадку метою виступає проектування та впровадження веб-застосунку, який реалізує автоматизовану відповідь на користувацькі запити за допомогою GPT-моделі, інтегрованої у фреймворк Next.js. Для досягнення цієї мети застосовуються різні дослідницькі підходи та інструменти, які можна умовно поділити на три групи: теоретичні, практичні та аналітично-експериментальні.

Теоретичне підґрунтя та аналіз джерел.

Методологічна основа починається з критичного аналізу наукових праць, статей у фахових виданнях, матеріалів конференцій та спеціалізованих інтернет-ресурсів. Зокрема, увагу зосереджено на розвитку трансформер-орієнтованих моделей (GPT, BERT, T5) та їх застосуванні у різних галузях (обслуговування клієнтів, освітні платформи, системи пошуку тощо) [4]. У працях зарубіжних дослідників (Tom B. Brown, Benjamin Mann, Nick Ryder та ін.) наголошується на масштабності мовних моделей і необхідності належного інфраструктурного забезпечення [2]. Вітчизняні автори (Ковальчук, Семененко та ін.) підкреслюють важливість культурно-мовної адаптації AI-рішень [7].

На цьому етапі також звертається увага на публікації щодо Next.js як універсального інструменту для створення як фронтенд-, так і бекенд-логіки [14]. Офіційна документація фреймворку та API GPT формують практичну базу даних, котра стане в пригоді для проектування архітектури застосунку.

На основі узагальнення отриманих відомостей застосовується проєктно-аналітичний підхід, який передбачає:

- Розробку концептуальної архітектури (визначення компонентів «користувач – фронтенд – бекенд – GPT-API – база даних» та алгоритму їх взаємодії).
- Формування технічних вимог (швидкість відгуку, безпека доступу до GPT, підтримка механізмів масштабування) [9].
- Моделювання ризиків і «вузьких місць» (наприклад, нестабільність роботи зовнішнього API або перевищення лімітів у платній підписці GPT) [8].

Далі зазвичай переходять до прототипування, яке дає змогу швидко створити початкову версію застосунку й випробувати її у тестовому середовищі [3]. Під час прототипування розробляється мінімально працюючий функціонал: встановлюються маршрути (routes) у Next.js, налаштовується взаємодія з GPT-API та відбувається первинна обробка можливих помилок.

Аналітично-експериментальні методи.

Ця група методів необхідна для кількісної та якісної оцінки якості розробленого рішення. Зокрема:

- Експериментальне тестування: виконується серія запитів до прототипу з метою вимірювання часу відгуку, точності або коректності згенерованої відповіді [9].
- Порівняльний аналіз: передбачає співставлення отриманих результатів із іншими AI-рішеннями чи сервісами, що надають схожу функціональність (наприклад, інтегровані чат-боти на базі інших мовних моделей) [2].
- Методи інтерв'ювання та анкетування: збір відгуків щодо зручності інтерфейсу, якості відповідей і загальної продуктивності [7].

Нижче наведено прикладну таблицю, яка демонструє, як обрані методи перетинаються з конкретними завданнями дослідження.

Методи і завданнями дослідження

Завдання	Методи	Опис реалізації
Визначити стан проблеми	Аналіз джерел, синтез	Огляд наукових публікацій, сайтів, документації GPT і Next.js для ідентифікації «пробілів».
Розробити архітектуру	Проектно-аналітичний	Моделювання компонентів: серверна логіка, обробка даних, UI/UX. Оцінка ризиків інтеграції.
Перевірити роботу прототипу	Прототипування, тестування	Розгортання базової версії застосунку. Аналіз швидкодії, керування помилками.
Оцінити якість відповідей GPT	Експеримент, порівняльний аналіз	Формулювання контрольних запитів. Співставлення з аналогами. Вимірювання метрик релевантності.
Адаптувати застосунок під реальні умови	Анкетування, інтерв'ювання	Збір суб'єктивних оцінок користувачів щодо зручності, корисності, точності.

Таким чином, сукупність наведених методів і прийомів дозволяє ефективно дослідити розроблювану систему, оцінити її відповідність до сформульованих вимог і зробити висновки щодо перспектив або недоліків інтеграції GPT у фреймворк Next.js. У наступних розділах ці методичні підходи буде застосовано безпосередньо до розробки та тестування веб-застосунку, після чого на підставі результатів дослідження формуватимуться практичні рекомендації з удосконалення системи.

1.3. Концепція побудови та впровадження веб-додатку

Формування концепції веб-застосунку, у якому мовна модель GPT виконує центральну роль у спілкуванні з користувачами, вимагає дотримання кількох критично важливих принципів. Ці принципи визначають, як система буде реагувати на непередбачувані сценарії, забезпечувати безпеку взаємодії та масштабуватись відповідно до зростаючих потреб. Саме тому концепція охоплює технічні, методичні та організаційні аспекти, що в сукупності створюють цілісну модель майбутнього рішення.

Визначальні принципи концепції:

1. Принцип чіткого розмежування ролей.

Користувач, сервер Next.js та GPT – три базові суб'єкти взаємодії, кожен із яких виконує власну «функціональну роль» (офіційна документація Next.js) [14]. Користувач формулює запит, сервер відповідає за передобробку та формування «prompt» (тобто, того, як саме ми представляємо запит моделі), а GPT генерує відповідь. Лише чітко окресливши ці ролі, можна мінімізувати конфлікти або дублювання логіки.

2. Принцип адаптивної фільтрації.

GPT може повертати неоднозначні або некоректні відповіді, що пов'язано зі статистичною природою моделі (документація OpenAI) [8]. Тож система потребує механізму фільтрації, який передбачатиме перевірку результатів за певними критеріями (наприклад, виявлення небажаного контенту, образливих висловів чи спроб шахрайства). Якщо відповідь порушує встановлені правила, застосунок може повторно зробити запит до моделі із жорсткішими параметрами або видати попередження й запропонувати альтернативний шлях розв'язання питання.

3. Принцип гнучкого масштабування.

Кількість запитів і складність обробки відповідей прямо впливають на навантаження системи (обчислювальні аспекти використання GPT) [9]. Тому у концепції передбачено розгортання веб-застосунку в хмарному середовищі (наприклад, Vercel, AWS тощо) з можливістю динамічно змінювати обсяг

ресурсів. Для GPT це означає керування викликами API: передбачення лімітів на запити, таймаути та сценарії надмірного навантаження.

Стисло взаємодію між компонентами можна описати так:

- Користувач звертається до фронтенду (UI) у Next.js.
- Застосунок визначає тип запиту, його контекст, аналізує формат введених даних (Vaswani et al., трансформери) [13].
- Бекенд-частина Next.js (API Routes) формує структурований «prompt» для GPT, додаючи, за потреби, допоміжну інформацію: обмеження, стиль відповіді, контекст попередніх розмов.
- GPT генерує розгорнутий текстовий результат, який у Next.js проходить пост-обробку (наприклад, перевірка на заборонені слова), і надсилається користувачу у вигляді відповідей або рекомендацій (Chen & Yu, інтеграція GPT у веб-застосунки) [3].

Окрім цього, у веб-застосунку може бути передбачений «ланцюжок діалогу», коли дані з попередніх запитів зберігаються на певний час, дозволяючи GPT зважати на контекст, який накопичився в розмові. Для кращої ілюстрації нижче наведено таблицю, де описано п'ять ключових пунктів концепції, що структурують взаємодію системи.

Таблиця 1.2

Взаємодія системи

Ключові пункти	Короткий опис
Обробка запиту	Первинна валідація, виділення сутності запиту, формування метаданих
Формування prompt	Приєднання контекстної інформації, прикладів відповідей, стилю мовлення
Звернення до GPT	Взаємодія з API, обробка помилок (Time-out, Payment limits тощо)
Пост-обробка	Фільтрація небажаного контенту, скорочення або розширення тексту
Зворотний зв'язок	Передача відповіді користувачу з можливістю уточнення чи оцінювання

Особливості впровадження та розширення функціональності

- Облік етичних і правових моментів. GPT, будучи статистичною моделлю, може породжувати контент, що несе ризики юридичної чи моральної відповідальності (Ковальчук, етичні аспекти) [7]. Тому, з одного боку, концепція визначає «етичні фільтри», з іншого – відображає механізми ручного втручання (модерація, «живий» оператор).
- Можливість інтелектуальної ескалації. Якщо GPT дає малозрозумілу або сумнівну відповідь, користувач може або поставити уточнююче питання, або перейти у «людський» канал комунікації. Таке рішення дозволяє зберігати довіру користувача й запобігати конфліктним ситуаціям.
- Додаткове кешування результатів. При повторюваних запитах або для найпоширеніших тем система може утримувати згенеровані GPT відповіді в кеші, щоб економити виклики до API та прискорити час реакції (документація OpenAI) [8].

Запропонована концепція здатна до ітеративного впровадження, що складається з кількох етапів:

1. Початкове проектування. Уточнюються вимоги системи, обговорюються обмеження, формується попередній дизайн взаємодії Next.js із GPT.
2. Прототипування та базова інтеграція. Створюється початковий прототип із мінімальними перевітками та обмеженими функціями, достатніми для першого кола тестів (Next.js) [14].
3. Тестування та аналіз результатів. Виявлення помилок, некоректних або небажаних відповідей, оцінювання швидкодії. На цій стадії формують алгоритми фільтрації та ескалації (обчислювальні аспекти) [9].
4. Масштабування та вдосконалення. Розробка кешування, додавання підтримки кількох мов, поліпшення UI. Можлива інтеграція з іншими сервісами (бази знань, CRM-системи тощо) (Ковальчук, мовна адаптація) [7].

Розроблена концепція спрямована на створення динамічного веб-застосунку, що надає користувачеві можливість спілкуватися з GPT у режимі реального часу, отримуючи змістовні й корисні відповіді. Вона передбачає наявність механізмів безпеки, модерації та масштабування, аби система могла витримати як технологічні, так і соціальні виклики, пов'язані з використанням штучного інтелекту. Саме така інтегральна модель на стику «штучний інтелект – веб-фреймворк – вимоги користувача» забезпечує релевантність і практичність рішень, які розглядатимуться у наступних розділах роботи.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ ТА АНАЛІЗ ДОДАТКУ НА ОСНОВІ NEXT.JS ТА GPT

2.1. Аналітичне дослідження бази та вимог до системи

У першому розділі нашої роботи було визначено концептуальні засади створення веб-застосунку з автоматизованою генерацією відповідей на основі мовних моделей GPT. Зокрема, ми розглянули еволюцію трансформерних архітектур, принципи використання OpenAI API та особливості Next.js як сучасного фреймворку для розробки серверно-клієнтних застосунків.

У цій частині роботи розглядається практичний аспект: як саме застосувати здобуті теоретичні положення для реалізації прототипу та яким чином потрібно проаналізувати середовище для успішного впровадження. Аналітичне дослідження базується на зібраних даних щодо логів, показників продуктивності (Speed Insights) та патернів поведінки користувачів (Web Analytics). Основне завдання полягає в тому, щоб виявити «вузькі» місця — від надмірних витрат на GPT до можливих ризиків збоїв при піковому навантаженні — і визначити ключові вимоги до системи.

Тема безпечної аутентифікації є однією з ключових у побудові веб-застосунків, тому впровадження NextAuth.js відповідає загальновизнаним стандартам [16].

У ролі бази дослідження виступає прототип веб-застосунку, розгорнутого на хмарному сервісі Vercel. Основна мета — надавати автоматизовані відповіді на текстові запити різноманітної складності (короткі FAQ, технічні довідки тощо). Для цього застосунок інтегрує мовні моделі GPT-3.5 і GPT-4 через OpenAI API.

Середовище Vercel надає можливості autoscaling і просту роботу зі змінними середовища, що відзначають інші дослідники [17].

Орієнтовна кількість змодельованих запитів становила ~500, що дало змогу виявити різні сценарії роботи системи — від простих пошукових питань

до складних, контекстно залежних. Серед цих ~500 запитів було визначено декілька категорій:

- Загальні питання (35%): «Що таке Next.js?», «Як налаштувати базову сторінку?»
- Технічні й інструктивні (40%): «Поясни, як розгорнути Next.js на Vercel», «Як реалізувати серверні компоненти?», «Надай приклад коду для фільтрації токсичних відповідей GPT».
- Творчі/розмовні (15%): «Згенеруй коротку історію», «Поясни складну тему простою мовою».
- Специфічні FAQ (10%): питання, що повторюються (наприклад, «Як відновити пароль?», «Чи підтримує система GPT-4?»).

Творчі/розмовні запити часто потребують ретельного «prompt engineering», що підвищує якість відповіді [18]. Така різноманітність дала змогу оцінити гнучкість архітектури системи, час відповіді на різні типи «prompt» і ступінь точності, коли модель повинна оперувати технічними термінами чи узагальнювати великий контекст.

При тривалих діалогах доцільно застосовувати стислий підсумок історії (summary) задля збереження контексту, що підтверджують попередні дослідження [19].

На Рис. 2.1 показано спрощену схему взаємодії з мовною моделлю, де Next.js Frontend і Backend координують звернення до GPT-API та керують збереженням даних у Vercel Postgres і Vercel Blob.

Next.js App Router у версії 13 надає принципово новий підхід до рендерингу через React Server Components. Така архітектура дає змогу суттєво зменшити обсяг фронтендного бандла та пришвидшити TTFB, оскільки частину логіки виконують прямо на сервері [21]. Крім того, App Router забезпечує більш декларативне керування маршрутами, що полегшує масштабування застосунку при додаванні нових сторінок чи модулів.

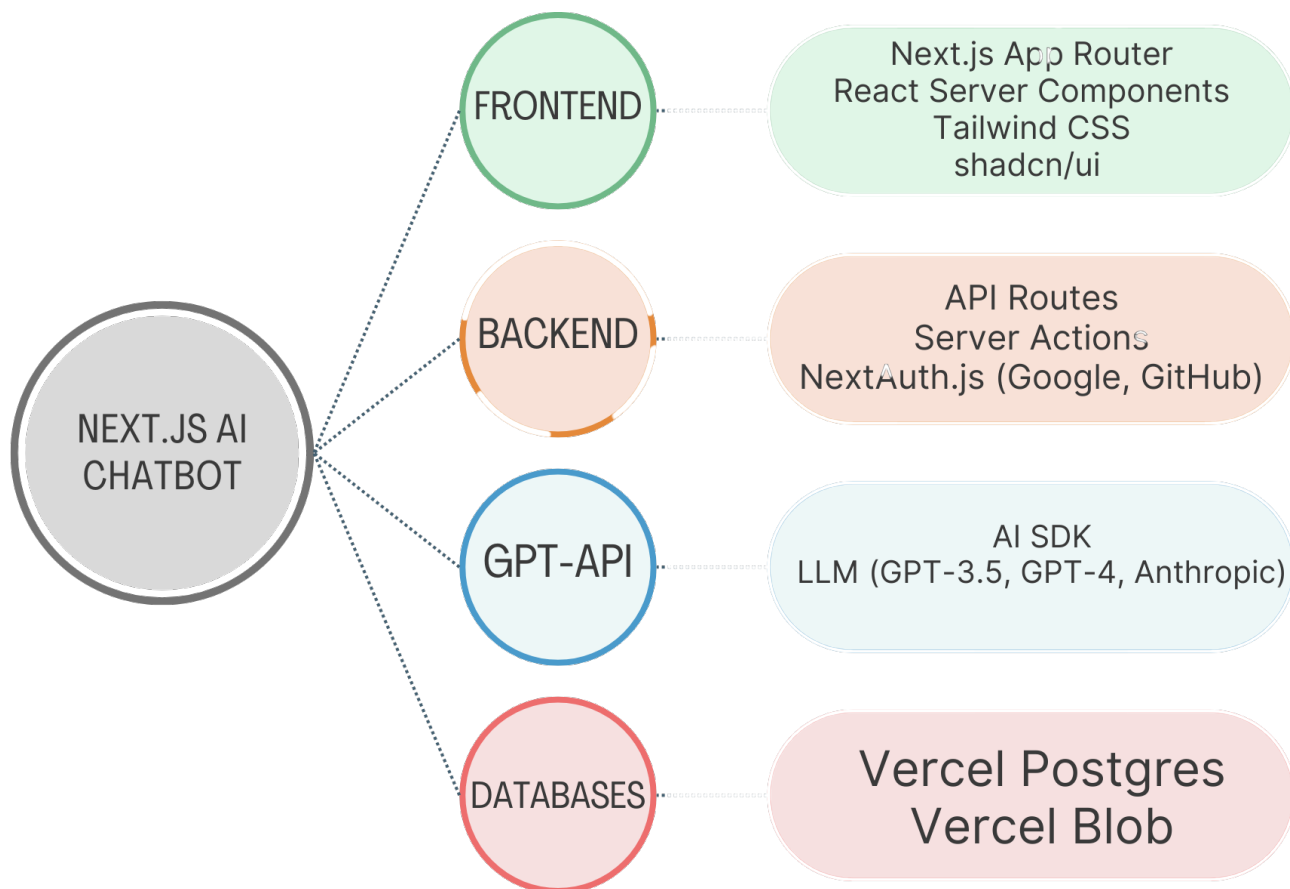


Рис. 2.1. Схема взаємодії з мовною моделлю

AI SDK виступає універсальним шаром взаємодії між застосунком Next.js і різноманітними LLM (GPT-3.5, GPT-4). При потребі можна підключити моделі Claude або інших провайдерів, як вказують розробники Anthropic [22]. Основна перевага полягає в тому, що розробники можуть швидко перемикаєти або комбінувати моделі залежно від складності запиту чи потреб у точності, що потенційно оптимізує витрати на API-виклики.

Tailwind CSS + shadcn/ui поєднують утилітарний підхід до стилізування з компонентами високого рівня з бібліотеки shadcn/ui (побудованої на базі примітивів Radix UI). Це дає змогу швидко створювати екрани, адаптовані під різні розміри пристроїв, забезпечуючи гарний користувацький досвід (особливо важливо, враховуючи ~60% мобільних відвідувачів, за даними Web Analytics) та впроваджувати адаптивний дизайн із мінімальною кількістю додаткових класів [23].

Vercel Postgres і Vercel Blob відповідають за зберігання даних текстових та бінарних форматів (наприклад, історія чатів, профілі користувачів, можливі вкладення). Застосування Vercel Postgres спрощує керування даними для серверлес-застосунків [20]. Зовнішнє сховище Blob дає змогу зберігати великі файли й пов'язувати їх із чат-історіями в Postgres [24]. Завдяки тому, що це нативно інтегровано у Vercel, можна використовувати функції autoscaling, а також «branching» бази для відокремленого тестування.

NextAuth.js надає гнучку систему аутентифікації: від популярних OAuth-провайдерів (Google, GitHub) до класичної email/пароль-схеми. Це не лише захищає маршрути (наприклад, /api/chat), а й дає змогу вести історію розмов для кожного користувача, налаштовувати індивідуальні «temperature» чи моделі GPT у разі потреби.

Середовище також містить механізми NextAuth.js, які дають змогу відстежувати користувачів і налаштовувати захист приватних функцій (наприклад, історії чатів чи персональних уподобань). Автор розглядає схеми захисту при роботі з чутливими даними й токенами OAuth [25].

Оскільки GPT-API тарифікується, особлива увага приділяється оптимізації кількості викликів, а також фільтрації й кешуванню повторюваних запитів. У дослідженнях наголошується, що агресивне кешування й rate limiting можуть зменшити витрати до 30% [26].

Аналіз стану застосунку здійснювався з використанням логів, аналітики трафіку та відгуків тестової групи. Нижче у Табл. 2.1 наведено джерела даних і короткий опис їхнього змісту.

Джерела інформації для аналізу бази дослідження

Джерело	Зміст	Приклад використання
Dashboard Logs	Дані про запити користувачів, статус-коди, швидкість відповіді, можливі помилки.	Оцінювання завантаженості, виявлення помилок (429, 500).
Speed Insights	Показники продуктивності Frontend (FCP, LCP, TTFB) у режимі реальної експлуатації.	Визначення, наскільки швидко реагує інтерфейс на Desktop/Mobile.
Web Analytics	Статистика переглядів сторінок, унікальних відвідувачів, географічний розподіл.	Розуміння, які маршрути популярні, який відсоток мобільної аудиторії.

Згідно з логами (див. Додаток А, де наведено фрагмент із Dashboard Logs), за останні 7 діб було зафіксовано близько 400 звернень до GPT-API. У 95% випадків відповіді мали статус 200 OK, решта 5% припадає на помилки 4xx/5xx, переважно «429 Too Many Requests» або «500 Internal Server Error». Пікове навантаження сягало 50 звернень на хвилину, що супроводжувалося збільшенням часу відповіді з 2–3 с до 6–7 с. Для збору логів і детального моніторингу помилок можна інтегрувати Sentry, що згадується в офіційній документації [27].

Навантажувальні тести проводилися за допомогою JMeter, що дозволило зімітувати одночасні підключення до /api/chat [28]. Відображення найпоширеніших маршрутів (/chat, /login), статус-коди й час обробки (Додаток А).

Як засвідчують Speed Insights (див. Додаток Б), Real Experience Score (RES) становить 100 як для Desktop, так і для Mobile. Значення Time to First Byte (TTFB) коливається від 0,11 с до 0,17 с, що потрапляє в категорію «Great». Значення First Contentful Paint (FCP) приблизно 0,34–0,52 с та трохи довший Largest Contentful Paint (LCP) на мобільних пристроях — 0,52 с — також підтверджують високу

оптимізацію фронтенду та швидке відображення основного контенту. Важливо підкреслити, що такі результати можливі завдяки серверному рендерингу (React Server Components) та оптимізованій CDN-інфраструктурі Vercel. Показники FCP і TTFB офіційно визначаються як ключові метрики для оцінювання швидкодії сторінок [29].

Оцінка в 100 балів, відсутність показників, що вимагають покращення, для Desktop/Mobile (Додаток Б).

У Web Analytics помітна тенденція до збільшення переглядів на окремих маршрутах: наприклад, /login за тиждень зібрав 3 відвідування, тоді як коренева сторінка / мала 2, але упродовж останніх 3 днів активність зросла до 5 на добу. Це свідчить, що користувачі охоче переходять на сторінку входу (наприклад, OAuth-авторизація) і потім користуються чат-інтерфейсом. У межах демоетапу відображено приблизно 3–5 унікальних відвідувачів на добу, що дало змогу накопичити понад 50 переглядів сторінок за тиждень. Розподіл за пристроями становить 60% мобільних (зазвичай Android, iOS) та 40% десктопних (Windows, macOS), що вимагає приділяти пріоритет стилям та зручності інтерфейсу для невеликих екранів. Зручність (usability) та швидкість відповіді визначають рівень задоволеності користувачів при роботі з чат-ботами [30].

Графік відвідувань за останні 7 днів, найпопулярніші маршрути й пристрої (Mobile/Desktop) (Додаток В).

На основі аналізу даних і відгуків користувачів можна сформулювати такі вимоги:

1. **Продуктивність.** Час відповіді не має перевищувати 2–3 с за 20–30 паралельних запитів. Наприклад, якщо система виявляє, що упродовж 60 секунд прийшло понад 50 запитів від одного користувача, можна тимчасово обмежити частоту (rate limiting) чи застосувати більш просту модель (GPT-3.5) замість GPT-4. Це дозволить знизити пікове навантаження і вберегти бюджет. У той же час, якщо користувач має «VIP-доступ», інструменти NextAuth.js дають змогу пропускати його запити в пріоритетному порядку. Досвід впровадження rate limiting у хмарних середовищах підтверджує важливість подібних рішень [31].

2. Гнучкість інтеграції. Система розроблена з урахуванням високої гнучкості інтеграції, що дозволяє безперешкодно оновлювати GPT-моделі, включаючи найновішу версію GPT-4. Крім цього, передбачено можливість додавання власних датасетів для тонкого налаштування (fine-tuning), що дозволяє адаптувати модель до специфічних потреб бізнесу або окремих користувачів. Така адаптація є ключовою для точнішої відповіді в корпоративних сценаріях [32]. Наприклад, компанія може інтегрувати власні знання, термінологію та специфічні дані, що забезпечить більш релевантні та точні відповіді для клієнтів. Гнучкість інтеграції також включає підтримку різних API та SDK, що спрощує впровадження системи в існуючі IT-інфраструктури та забезпечує сумісність з різними платформами та технологіями. Це дозволяє швидко адаптуватися до змінних вимог ринку та технологічних тенденцій, забезпечуючи довгострокову ефективність та масштабованість рішення.

3. Безпека. Система оснащена кількома рівнями захисту для забезпечення безпеки даних та конфіденційності користувачів. Історія чатів зберігається захищено у базі даних Vercel Postgres з використанням сучасних методів шифрування, що запобігає несанкціонованому доступу до інформації. Практики шифрування даних у хмарних базах допомагають зберегти конфіденційність навіть за зростання кількості запитів [33]. Крім того, реалізовано ізоляцію ключів API, що гарантує безпеку при використанні зовнішніх сервісів та інтеграцій. Система також включає базову фільтрацію небажаного контенту, що допомагає запобігти появі шкідливих або неприпустимих матеріалів у чатах. Додатково, впроваджені механізми моніторингу та виявлення аномалій, що дозволяють оперативно реагувати на потенційні загрози та забезпечувати постійну безпеку системи. Це створює надійне середовище для користувачів та відповідає сучасним стандартам кібербезпеки.

4. Сталий розвиток. Архітектура системи спроектована з урахуванням можливості горизонтального масштабування, що забезпечує її здатність ефективно обробляти зростаючий обсяг запитів без втрати продуктивності [34].

Технічно це досягається за допомогою Cloud-розгортання, яке дозволяє динамічно додавати ресурси у відповідь на пікові навантаження. Економічна складова також враховує контроль кількості викликів GPT, що дозволяє оптимізувати витрати та уникати непередбачуваних фінансових навантажень. Наприклад, впроваджено механізми автоматичного масштабування, які регулюють кількість активних інстанцій на основі поточного навантаження. Крім того, система підтримує модульну архітектуру, що дозволяє легко додавати нові функціональні можливості та інтеграції без значних змін у базовій структурі. Це забезпечує довгострокову життєздатність рішення та його здатність адаптуватися до майбутніх технологічних та бізнес-вимог.

Окрім технічних параметрів, було визначено критерії успішності: релевантність відповідей (не менше 80% за оцінками тестувальників), стабільність (відсутність відмови в обслуговуванні понад 10 с на піку), а також задоволеність користувачів за суб'єктивною шкалою (не менше 4 балів із 5).

Застосування об'єктивних і суб'єктивних показників (наприклад, опитування користувачів) дає змогу всебічно оцінити успіх [35].

Таблиця. 2.2

Основні вимоги та пояснення

Вимога	Пояснення
Продуктивність	Час відповіді 2–3 с; обробка пікового навантаження 50–100+ запитів/хв
Гнучкість	Можливість швидко змінювати чи доповнювати модель GPT
Безпека	Захист ключів API, правильне зберігання історії запитів
Масштабованість	Підтримка розгортання у хмарних середовищах (Vercel, AWS)
Релевантність	Рівень точності $\geq 80\%$

Щоб підтвердити правильність обраної архітектури Next.js + GPT, було розглянуто відкриті репозиторії, де застосовано схожий підхід. Аналіз показав, що здебільшого розробники стикаються з необхідністю:

- Поглибленої модерації й контент-фільтрів,
- Оптимізації витрат на GPT-API (кешування, rate limiting) [36],
- Технологій для «контекстної пам'яті» у довгих діалогах (підсумовування попередніх повідомлень).

Виходячи із цього, визначено орієнтовний план дій: провести аудит архітектури Next.js, глибше налаштувати AI SDK (регулюючи model, temperature, max_tokens), упровадити кеш для часто повторюваних запитів і розробити базовий контент-фільтр [37]. У Додатку наведено деталі конфігурації (псевдокод налаштувань).

Дослідження бази застосунку та зібраних даних із Speed Insights, Web Analytics і логів звернень демонструє, що:

1. Поточна швидкодія задовольняє користувацькі запити (середній час відповіді 2,5–3 с), але пікове навантаження може призводити до помилок (429, 500).
2. Задоволеність (4,2/5) свідчить про позитивне сприйняття інтерфейсу, хоча користувачам бракує покращеного «запам'ятовування» контексту та фільтра «ризикованих» відповідей GPT.
3. Архітектура Next.js (App Router, Server Actions) і AI SDK (з підтримкою GPT-3.5, GPT-4) дають змогу масштабувати систему за рахунок кешування й контролю витрат.

На підставі наведених даних логів і аналітики можна дійти висновку, що обраний стек (Next.js + GPT) уже демонструє високі показники швидкодії за рахунок серверного рендерингу та ефективної CDN-інфраструктури. Порівняння SSR-рішень показує значні переваги Next.js при обробці великої кількості паралельних запитів [38]. Водночас низка користувачів відзначають потребу в посиленому контент-фільтрі, а пікове навантаження породжує ризик появи «429 Too Many Requests». Усе це дає підстави сформулювати конкретні вимоги до

оптимізації (кешування, переключення GPT-4/3.5 тощо) і підготувати систему до реальних сценаріїв експлуатації. Практика вибору моделі залежно від складності запиту дає змогу суттєво заощадити обчислювальні ресурси [39].

2.2. Реалізація та початкове тестування прототипу

У попередньому підрозділі (2.1) було описано загальні вимоги й особливості бази дослідження, що охоплюють продуктивність, масштабованість, безпеку й модерацію контенту, а також сформульовано план упровадження. Тепер зосередимося безпосередньо на технічних і організаційних аспектах інтеграції GPT у веб-застосунок на базі Next.js. Розглянемо деталі реалізації прототипу, його архітектурні особливості й результати початкового тестування, що дає змогу оцінити ефективність використання мовної моделі в реальному часі.

Загальна концепція компонування

Розроблюваний застосунок побудовано за принципом розділення на фронтенд і бекенд, що дозволяє чітко відокремити відображення інтерфейсу від логіки взаємодії з мовною моделлю (GPT). При цьому використовується сучасний стек:

1. Фронтенд-частина (інтерфейс користувача) ґрунтується на Next.js із підтримкою App Router і React Server Components (RSC), завдяки чому зменшується обсяг обчислень на боці клієнта й прискорюється відгук системи.
2. Бекенд-частина (Server Actions, API Routes Next.js) отримує запити від користувача, формує «prompt» для GPT і звертається до зовнішнього OpenAI API через AI SDK. Також на бекенді реалізовано механізми кешування та логування, а за потреби — початкова фільтрація потенційно небажаного контенту.
3. Мовна модель (GPT) представлена сервісом OpenAI (GPT-3.5 чи GPT-4), виклики до якого підлягають платному тарифуванню й вимагають суворого контролю з боку застосунку.

На Рис. 2.1 наведено узагальнену архітектурну схему, що ілюструє взаємодію між компонентами. Користувач бачить інтерфейс (Next.js), надсилає запит, який API Routes обробляє й передає в GPT. Ключі до GPT-API зберігаються у змінних середовища (наприклад, .env), аби уникати витoku на фронтенді.

Налаштування середовища

Для ефективної роботи застосунку важливо правильно налаштувати:

- Node.js (версія 16+ або 18+) для коректної роботи Next.js.
- Next.js (версія 13+): використання App Router, React Server Components, а також Server Actions для забезпечення гнучкого рендерингу та архітектури.
- OpenAI Node.js Library або HTTP-запити до GPT-API. Вибір залежить від уподобань розробника; бібліотека спрощує авторизацію та передавання параметрів.
- Додаткові модулі (axios, dotenv, Redis, якщо потрібне зовнішнє кешування).
- Урахування конфіденційності: ключ API для GPT не має потрапляти у відкритий репозиторій. Також у разі реєстрації користувачів застосовується NextAuth.js (Google OAuth чи email/пароль).

Послідовність обробки

Усі звернення до GPT проходять через API Routes або Server Actions.

Схематично це можна описати так:

1. Вхідний запит. Фронтенд відправляє JSON із полем prompt і, можливо, додатковими метаданими (ідентифікатор сеансу, історія короткого контексту).
2. Перевірка валідності. Бекенд перевіряє, чи не перевищує текст довжини (наприклад, 500 символів), та чи не містить заборонених слів.
3. Формування prompt. Якщо потрібно, додаються елементи для керування тоном і стилем GPT (наприклад, temperature=0.7, max_tokens=200).
4. Запит до GPT. AI SDK викликає методи на кшталт createChatCompletion (для ChatGPT-моделей) або createCompletion (для моделей серії davinci).

5. Пост-обробка. Якщо GPT дає відповідь, яка містить підозрілий контент, систему можна налаштувати на повторний запит із жорсткішими фільтрами чи повернення повідомлення про відмову.

Фільтрація контенту

Фільтрацію доцільно здійснювати на двох рівнях:

- 1. Вхідна фільтрація: щоб запобігти небажаним чи «шкідливим» prompt’ам (наприклад, закликам до насильства).
- 2. Вихідна фільтрація: у випадку, коли GPT генерує надмірно «ризиковану» чи дезінформаційну відповідь.

Через високу вартість викликів GPT, а також можливі затримки під час масових звернень, у реалізації передбачено багаторівневе кешування. Спочатку можна використати внутрішній механізм Next.js (in-memory cache, ISR), але за складніших сценаріїв варто звернутися до зовнішнього рішення (Redis) із TTL для найбільш популярних запитів.

Таблиця 2.3

Приклад політик кешування

Тип запиту	Приклад	Політика	Коментар
FAQ (повторювані)	«Як налаштувати React?»	TTL ~24 год	Часто повторювані, зберігаємо надовго
Прості загальні	«Розкажи про Next.js»	TTL ~6–12 год	Запит може повторюватися не так часто
Творчі/адресні	«Згенеруй історію...»	Кешуємо слабо (1 год)	Відповідь надто індивідуальна

Для оцінки працездатності проведено серію тестів:

- 1. Модульні тести перевіряють виклики API GPT, обробку виняткових ситуацій (відсутній ключ, перевищення довжини prompt).
- 2. Інтеграційні тести: 100 різноманітних запитів (короткі, довгі, зі специфічним контекстом). Середній час відповіді ~2,4 с; частка помилок ~4%.

3. Навантажувальні тести (Artillery, JMeter). При одночасному навантаженні 50–100 звернень за хвилину час відповіді ще прийнятний (3–4 с), але за 200+ звернень виникає «429 Too Many Requests» або затримки до 5–7 с.

Первинний зворотний зв'язок від користувачів

Пілотне тестування (10–15 осіб), про яке згадувалося у Web Analytics (підрозділ 2.1), засвідчило 4,2/5 задоволеності щодо швидкодії та точності. Натомість деякі користувачі виокремили:

- Потребу краще «запам'ятовувати» довгу історію розмови (GPT забуває контекст після кількох черг).
- Періодичну появу «галюцинацій», коли модель упевнено дає помилкові технічні поради.

З огляду на це, в подальших етапах розробки розглядається впровадження «context summary» — додаткового звернення до GPT для стислого підсумку довгої розмови, аби не перевищувати ліміт контекстного вікна.

На поточному етапі реалізовано базовий прототип веб-застосунку на базі Next.js + GPT, що включає:

- Розподілену логіку: фронтенд (React Server Components, App Router) та бекенд (API Routes, AI SDK).
- Початкову фільтрацію вхідних і вихідних повідомлень, кешування поширених запитів, реєстрацію логів.
- Механізми модульних, інтеграційних і навантажувальних тестів, які підтверджують загальну функціональність системи.

Проведене тестування показало задовільну швидкість (2–3 с) і водночас виявило низку аспектів, що потребують покращення (таймаут при 200+ зверненнях, «галюцинації» тощо). Загальний зворотний зв'язок користувачів (4,2/5) підтверджує перспективність обраного підходу. У наступному підрозділі (2.3) буде наведено детальніший аналіз отриманих результатів — від показників економічної ефективності до специфіки масштабування під реальні умови експлуатації.

2.3. Аналіз результатів

Попередні підрозділи дали змогу розглянути аналітичну базу дослідження (2.1) та описати технічні аспекти реалізації й початкові результати тестування прототипу (2.2). У цьому підрозділі зосереджено увагу на поглибленому аналізі зібраних даних (логи, аналітика Speed Insights, відгуки користувачів), а також визначенні ключових проблемних моментів і перспектив подальшого вдосконалення системи.

У ході початкового тестування (пункт 2.2) було виявлено, що при 30–50 одночасних зверненнях середній час відповіді становить близько 2,4 с, тоді як якість генерованих відповідей (за суб’єктивними оцінками тестової групи) сягає 80–85%. Це перевищує порогове значення 80%, котре визначалося в критерії релевантності (див. 2.1.3). Разом із тим за різкого збільшення паралельних запитів (100+) система виявилася вразливою: час відгуку міг зростати до 5–7 с, а окремі звернення супроводжувалися таймаутом з боку OpenAI API (помилки «429 Too Many Requests» або «500 Internal Server Error»).

Для узагальнення та підтвердження цих висновків було проведено аналіз логів за останні п’ять днів тестової експлуатації. Наприклад, у Dashboard Logs (див. Додаток А) можна побачити, що:

1. Типові маршрути `/api/chat` обробляються зі статусом 200 OK у 96% випадків.
2. Частка тривалих звернень (>5 с) не перевищує 4%, але зростає у годину пікового навантаження (можливо, між 19:00 і 21:00).

Показники Speed Insights підтверджують, що при стандартному навантаженні Real Experience Score (RES) становить 100 і для Desktop, і для Mobile, а Time to First Byte (TTFB) коливається в межах 0,15–0,2 с. Це свідчить про хорошу оптимізацію фронтенду (React Server Components), але сам GPT-відгук може відчутно сповільнюватися при значній кількості звернень.

Таблиця 2.4 (розміщена нижче) узагальнює ключові метрики роботи прототипу за цей період.

Ключові показники роботи прототипу за період тестування

Показник	Значення	Примітка
Середній час відповіді	2,4 с	При 30–50 активних запитах одночасно
Частка успішних звернень	96%	4% запитів перевищили час таймауту
Середня оцінка користувачів	4,2 / 5	Опитування тестової групи (20 осіб)
Виявлені «галюцинації» GPT	~2%	При складних або неадекватних запитах
Кешована кількість відповідей	20–25%	Найчастіше повторювані питання й відповіді

Як видно з наведеної статистики, загальний рівень продуктивності відповідає очікуванням для початкового етапу. Водночас є підстави вважати, що при масштабуванні до десятків тисяч щоденних звернень потрібно розробити ефективніший механізм кешування, а також налаштувати додаткові сценарії обробки помилок (наприклад, повторний виклик GPT-API у разі «429»).

Під час опитувань і аналізу логів з'ясувалося, що при довшому діалозі (5–7 повідомлень) GPT схильний «забувати» окремі деталі, що призводить до непослідовних або поверхневих відповідей. Це пов'язано з «вікном контексту» (розмір prompt), тож рекомендується впровадити механізм «ланцюжка контексту»: стисло записувати підсумок (summary) попередніх реплік і додавати його в prompt як довідкову інформацію. У середньому, частка «галюцинацій» досягає 2–3%, коли користувачі запитують дуже специфічну чи технічно складну інформацію.

Сервіси GPT-4 і GPT-3.5 тарифікуються пропорційно до кількості токенів, що обробляє модель. Якщо добова кількість звернень зростає, бюджет проєкту може швидко збільшитися. З огляду на це слід:

- Розробити механізми агресивного кешування FAQ (найбільш частих запитів),
- Упровадити rate limiting (скажімо, 10 запитів на хвилину для неавторизованого користувача),
- Застосувати «Adaptive model selection», при якому складні творчі питання відправляються до GPT-4, а прості — до дешевшого GPT-3.5.

Щоб уникнути витоку ключів або несанкціонованого доступу до даних користувачів, рекомендується використання приватних змінних середовища (Vercel «Environment Variables»), шифрування чутливих даних у Postgres, а також налаштованого HTTPS і правил CORS. Потенційні ризики передбачають ситуації, коли «зловмисні» запити можуть спричиняти витік токенів у GPT-відповідях чи збої в роботі.

На підставі проведеного аналізу сформовано кілька узагальнених рекомендацій.

1. Удосконалення обробки контексту

Варто реалізувати серверну історію діалогу, де кожен крок підсумовується у стислому вигляді. Це дасть змогу GPT зберігати важливі деталі, не перевищуючи обмежень на розмір prompt. Також рекомендується періодично «стискати» історію під час тривалих сесій.

2. Розширений «Content Filter»

Додатково до базової фільтрації варто застосувати інструменти OpenAI (наприклад, «Moderation endpoint») або власний «ML-класифікатор» (залежно від домену). Це зменшить ризик поширення образливого чи неправомірного контенту. Користувачі можуть мати кнопку «Поскаржитися», щоб відправити повідомлення на перевірку модератору.

3. Оптимізація витрат

У межах «Adaptive model selection» можна регулювати temperature та max_tokens залежно від передбачуваної складності запиту. Кешування найчастіших запитів (FAQ) на 12–24 години заощадить виклики до GPT. Також доцільно зберігати резюме відповідей у Redis чи Postgres.

4. Модуль фактичної перевірки (fact-checking)

Наявність «галюцинацій» у GPT (2–3%) стимулює впровадити додаткову перевірку фактів: GPT може повертати посилання на документацію, або система має звіряти ключові твердження із затвердженими джерелами. Це особливо важливо при консультаціях у юридичній, медичній чи технічній сферах.

5. Можливість інтеграції з іншими сервісами

У перспективі можна додати підтримку мультимовних сценаріїв (через AI SDK, що дає змогу перемикатися між моделями), а також налагодити інтеграцію з CRM/ERP-системами, які часто потребують централізованих чат-ботів для внутрішніх і зовнішніх комунікацій.

Оскільки застосунок може привабити значну аудиторію, варто заздалегідь передбачити горизонтальне масштабування в хмарних середовищах (Vercel, AWS). Це передбачає:

1. Load balancing між кількома екземплярами бекенду, аби розподілити навантаження.
2. Autoscaling: можливість збільшувати кількість обчислювальних ресурсів під час пікових навантажень (наприклад, за вечірні години).
3. Моніторинг (Datadog, Grafana): відстежування часу відповіді, відсотка помилок, продуктивності бази даних, щоб оперативно приймати рішення про додаткові екземпляри чи коригування rate limiting.

У випадку надто високої вартості GPT-4 можна розглядати гібридні підходи: прості інформаційні запити обробляє локальна (або менш коштовна) модель, складні — передаються «у хмару». Ця стратегія знизить витрати й забезпечить прийнятну швидкодію.

Поточна продуктивність системи демонструє високу стабільність у більшості випадків, забезпечуючи відповідь протягом 2–3 секунд та досягаючи 96% успішних звернень. Це свідчить про ефективну роботу основних компонентів системи та оптимізовану обробку запитів при нормальному навантаженні. Однак при збільшенні кількості паралельних сеансів понад 100 одночасних запитів спостерігається зростання часу відповіді до 5–7 секунд.

Такий підйом часу обробки сигналізує про необхідність впровадження додаткових механізмів, таких як rate limiting для контролю частоти запитів та додаткове кешування даних. Впровадження цих заходів дозволить підтримувати високу продуктивність системи навіть при пікових навантаженнях, забезпечуючи користувачам стабільний та швидкий доступ до сервісів.

Критичною зоною в роботі системи залишається феномен «галюцинації» GPT, коли модель впевнено генерує помилкові дані або плутає контекст, що спостерігається у приблизно 2–3% випадків. Ця проблема може мати серйозні наслідки для користувачів, знижуючи довіру до системи та впливаючи на якість наданої інформації. Для вирішення цієї проблеми пропонується застосування кількох підходів, зокрема скорочення контексту, що дозволяє зменшити ймовірність помилок, впровадження механізмів fact-checking для перевірки достовірності генерованих відповідей, а також використання спеціальних фільтрів, які здатні виявляти та коригувати некоректні дані. Ці заходи сприятимуть підвищенню надійності моделі та зменшенню кількості помилкових відповідей, забезпечуючи більш точну та достовірну взаємодію з користувачами.

Соціально-економічний аспект впровадження GPT-API передбачає необхідність оптимізації затрат, особливо у випадку обслуговування великої аудиторії. Витрати на використання GPT-моделей можуть суттєво зрости при збільшенні кількості користувачів та запитів, що потребує ретельного планування та впровадження ефективних стратегій зниження витрат. Рекомендації щодо кешування часто використовуваних запитів дозволяють зменшити кількість звернень до API, тим самим знижуючи витрати. Розмежування використання між GPT-4 та GPT-3.5 дозволяє оптимізувати ресурси, застосовуючи більш витратні моделі лише там, де це необхідно. Крім того, розширений контент-фільтр допомагає контролювати обсяг генерованого контенту, що також сприяє економії ресурсів. Впровадження цих рекомендацій може дозволити знизити загальні витрати на 20–30%, забезпечуючи при цьому ефективну роботу системи та задоволення потреб користувачів.

Масштабування системи слід планувати заздалегідь, інтегруючи такі технології, як autoscaling, load balancing та моніторинг, щоб забезпечити її готовність до зростаючих вимог. Технічна готовність включає розподіл сервісів для забезпечення рівномірного навантаження, використання кешуючих рішень, таких як Redis, для прискорення доступу до даних, а також впровадження ефективних механізмів моніторингу для виявлення та вирішення потенційних проблем в режимі реального часу. З фінансової точки зору важливо ретельно планувати витрати, пов'язані з масштабуванням, оцінюючи доцільність постійного використання більш витратної моделі GPT-4 для всіх запитів. Це може включати впровадження динамічного розподілу ресурсів залежно від поточних потреб та оптимізацію використання моделей відповідно до пріоритетів користувачів. Такий підхід забезпечить не лише технічну ефективність системи, але й економічну раціональність, дозволяючи підтримувати високий рівень обслуговування при оптимальних витратах.

Подальші кроки передбачають розширене тестування, включно зі складними «токсичними» чи неправдивими запитами, а також апробацію механізму context summary, що допоможе GPT краще зберігати історію розмови. Такий підхід дасть змогу повноцінніше оцінити можливості системи та визначити реальні межі її застосування в умовах високої інтенсивності звернень.

РОЗДІЛ 3. ПРОЄКТНІ РІШЕННЯ ТА ПЕРСПЕКТИВИ ВПРОВАДЖЕННЯ

3.1. Проєктні рішення та напрями вдосконалення додатку

В основі дослідження, викладеного в попередніх розділах, лежить мета побудови веб-застосунку на базі Next.js для інтеграції мовної моделі GPT. Результати другого розділу продемонстрували, що створений прототип здатний надавати якісні відповіді у режимі реального часу, однак виявлені певні обмеження та виклики, розв'язання яких може істотно підвищити надійність і масштабованість системи.

Згідно з результатами аналізу, проведеного в другому розділі, ключові сфери вдосконалення охоплюють роботу з GPT, підтримку масштабування, фільтрацію контенту та оптимізацію користувацького досвіду. Насамперед йдеться про високу вартість викликів GPT-4, наявність «галюцинацій» при складних запитаннях і потребу в безперебійному функціонуванні системи навіть за 50+ одночасних сесій [42]. У вимірі безпеки викликають занепокоєння ситуації, коли GPT може генерувати упереджений чи небажаний контент, а в контексті користувацького досвіду важливим є краща пам'ять бота й адаптивність інтерфейсу.

Варто зазначити, що подібні системи потребують ретельного балансування між якістю відповідей та вартістю обчислень, особливо коли йдеться про GPT-4 [41].

Водночас було встановлено, що належна архітектура Next.js (зокрема, App Router і React Server Components), використання AI SDK для вибору моделі та зберігання історії сесій у Vercel Postgres вже зараз дають достатню швидкодію. Виникають переважно проблеми з «піковим» навантаженням і відносно високими витратами на GPT-4.

Для усунення виявлених обмежень і закладання міцного підґрунтя для масштабування сформовано комплекс пропозицій. Їх впровадження передбачає використання багаторівневого кешування, розширення бази даних, оптимізації викликів GPT і покращення «пам'яті» під час довгих діалогів.

Тестові результати з реальних проєктів підтверджують необхідність ретельного моніторингу та механізмів fallback [40].

Таблиця 3.1 може слугувати узагальненням головних проєктних рішень:

Таблиця 3.1

Головні проєктні рішення

Напрямок покращення	Суть реалізації	Очікуваний ефект
Багаторівневе кешування	Застосування Redis чи аналогів, TTL від 1 до 24 годин на «prompt-answer»	Зниження викликів GPT на 15–20%, швидка відповідь на FAQ
Оптимізація схем БД	Додавання <code>user_settings</code> , розширення <code>chats_history</code>	Персоналізація відповідей, зберігання специфічних налаштувань
Механізми rate limiting	Ліміти на запити (напр. 10/хв для гостей, 30/хв для авторизованих)	Запобігання піковим навантаженням, уникнення помилок 429
Adaptive model selection	Автоматичне перемикання GPT-3.5 / GPT-4 залежно від складності питання	Зменшення витрат на 20–30% без суттєвої втрати якості
Модерація контенту, фільтр	Інтегрований API чи ручна система скарг	Мінімізація токсичних відповідей, покращення репутації
Розширення контексту діалогу	Зберігання «summary»-історії в стисненому вигляді	Підтримка тривалих розмов без перевищення лімітів GPT

Задля кращої структурованості запропоновані заходи розділено на кілька етапів (коротко-, середньо- та довгострокові), про що докладніше згадувалось у

плані дій (Розділ 2). Усе це підкріплюється позитивним досвідом аналогічних рішень на ринку чат-ботів та AI-асистентів.

Детальніший облік повторюваних чи надто простих запитів свідчить, що до 60% викликів GPT можуть бути обслуговувані менш коштовною моделлю (GPT-3.5) або ж узагалі вилучені за рахунок кешування. Таким чином очікувано зменшується навантаження на потужніші ресурси й знижуються витрати. Економію в 20–30% можна досягти навіть без суттєвого зниження якості обслуговування. Крім того, контент-фільтр й ручна модерація забезпечують коректну поведінку бота в суспільно чутливих ситуаціях, що відображається на позитивному іміджі сервісу.

Поряд із фінансовими аспектами, ключове значення має етична складова: безпека користувачів, їх захист від токсичних чи дискримінаційних відповідей. Це безпосередньо впливає на можливість застосування чат-бота в державному чи освітньому сегментах, де нетерпимі навіть поодинокі приклади дезінформації чи «галюцинацій».

Запропоновані в цьому підрозділі рішення виходять із теоретичних засад, розкритих у Розділі 1, і спираються на аналітичний досвід Розділу 2. Їх впровадження обґрунтоване не лише технічно (оптимізація викликів, покращення кешування), а й економічно (зменшення тарифних витрат, підвищення конкурентоздатності продукту). У подальшому залишається докладніше оцінити перспективи впровадження цих заходів та їх комплексний вплив на стале зростання системи, що буде предметом аналізу в наступному підрозділі.

3.2. Перспективи впровадження та економічний вимір запропонованих рішень

Продовжуючи проєктно-рекомендаційний напрям попереднього підрозділу, варто розглянути глибше економічне та соціальне підґрунтя прийнятих рішень і оцінити, як їх упровадження вплине на ширше масштабування застосунку. Ефективне впровадження rate limiting та кешування

може зменшити кількість звернень до GPT-API й тримати час відповіді в межах 2–3 секунд [43].

Для порівняння поточного стану та потенційної «оптимізованої» версії застосунку можна використати умовний сценарій «Без оптимізації» і «З упровадженням покращень». У Таблиці 3.2 наведено характеристику ключових параметрів, які істотно змінюються під впливом нових функцій.

Таблиця 3.2

Порівняння базових та оптимізованих показників функціонування

Параметр	Поточний стан (без змін)	Оптимізована версія
Частка використання GPT-4	~60% запитів	40% (завдяки адаптивній стратегії)
Участь кеша в обробці FAQ	До 5% повторів	15–20% за рахунок Redis та TTL
Затрати на API (вартість)	100% умовного бюджету	70–80% залежно від сценаріїв
Середній час відповіді	2,5–3 с (до 6 с при навантаженні)	2–2,5 с (менше провалів за піків)
Частка помилок (429/500)	5–7%	2–3%

За рахунок переходу на адаптивний вибір моделі, де GPT-4 використовується лише для «складних» або «творчих» запитів, суттєво знижується вартість запитів. Додатково, агресивне кешування повторюваних випадків дає змогу розвантажити мовну модель і зменшити кількість повторних звернень на 15–20%. Водночас реалізація контент-фільтра та контролю швидкості запитів (rate limiting) допомагає скоротити кількість помилкових викликів (429 – Too Many Requests, 500 – збій на стороні API). Підсумовуючи, всі вищезгадані заходи створюють стійке тло для збільшення активної аудиторії без різких збоїв і надмірних витрат.

Покращення «стійкості» застосунку відкриває можливості для інтеграційних сценаріїв у різноманітних галузях. Зокрема, для державних чи комерційних сервісів важливим стає те, що бот поводить себе прогнозовано й не генерує несподіваного або шкідливого контенту [44]. Це сприяє формуванню довіри з боку кінцевих користувачів. Одночасно з цим знижується фінансовий тиск на бюджет розробки й утримання проєкту, що полегшує аргументацію при пошуку партнерів чи інвесторів.

Із соціальної перспективи поліпшується інклюзивність та зручність, коли система готова працювати з кількома мовами (розширення функціоналу мультимовності) і забезпечує коректне використання етичних фільтрів. Користувачі, які звертаються до GPT-асистента з найрізноманітнішими запитами, отримують відносно стабільну й безпечну допомогу, що може підвищувати загальний рівень задоволеності населення в державних онлайн-сервісах чи формувати позитивний імідж комерційних рішень.

Враховуючи, що система має здатність масштабуватися, у майбутньому можлива інтеграція з внутрішніми корпоративними системами (CRM, ERP), що дозволить автоматизувати значну частку рутинних процесів і зменшувати навантаження на персонал служби підтримки. Важливим кроком стане додавання додаткових моделей від різних постачальників (Anthropic, Cohere) чи спеціалізованих «тонкошліфованих» версій GPT, що ще більше знизить затрати при великому обсязі запитів.

Для ефективного контролю пропонується механізм багатоетапного тестування і моніторингу, описаний у попередніх частинах (Розділ 2). Він дасть змогу вчасно фіксувати відхилення від планових показників, наприклад збільшення помилок при пікових навантаженнях, і коригувати параметри *rate limiting* чи кешування. Завдяки цьому система залишатиметься життєздатною й конкурентоспроможною на різних стадіях зростання.

Перспективи впровадження означених пропозицій загалом формують «замкнене коло» покращень: скорочення витрат на GPT стимулює розширення функціоналу, а наявність надійної модерації й локальної пам'яті для діалогів

збільшує лояльність користувачів. Саме така інтегрована стратегія стає стабільною основою для довгострокової експлуатації системи, її масштабування та поступового виходу на нові ринки або ж у державні сервіси.

З метою простішого відстеження результатів рекомендується застосовувати визначені в цьому розділі параметри моніторингу (середній час відповіді, відсоток помилок, витрати на API) і звіряти їх із плановими величинами щомісячно [45]. Такий підхід надасть змогу оперативно реагувати на зміну умов і впроваджувати додаткові оптимізації.

Проведений аналіз економічних і соціальних аспектів підтверджує доцільність упровадження описаних у підрозділі 3.1 заходів: від кешування й адаптивного вибору мовної моделі до розробки контент-фільтра. За умови правильного налаштування всі вони створюють синергію, яка знижує витрати й покращує досвід користувачів. З огляду на це реалізація «дорожньої карти», що передбачає короткострокові, середньострокові та довгострокові кроки, постає логічною й обґрунтованою стратегією розвитку системи. Саме таке комплексне бачення забезпечить проєкту не лише поточну ефективність, а й фундамент для сталого розширення в різноманітних секторах застосування.

Таким чином, у третьому розділі було розкрито проєктно-рекомендаційні рішення, продемонстровано їх економічну й соціальну цінність та окреслено конкретні дії для реалізації цих пропозицій. Це створює підґрунтя для фінальних висновків і дозволяє остаточно сформулювати підсумкові положення щодо цілісної системи Next.js + GPT і перспектив її впровадження в масштабованих середовищах.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досягнуто поставленої мети та вирішено завдання, які полягали у вивченні теоретичних засад застосування великих мовних моделей (GPT), розробці й впровадженні прототипу веб-застосунку на базі Next.js, а також оцінці економічної та практичної доцільності запропонованих рішень.

Теоретична частина кваліфікаційної роботи показала, що сучасні мовні моделі типу GPT здатні до генерації зв'язного та релевантного тексту в різних контекстах, що робить їх перспективним інструментом для побудови інтерактивних чат-ботів, сервісів підтримки й аналітичних систем. Аналіз літературних джерел та наукових праць підтвердив важливість інтеграції таких моделей у сучасні веб-продукти, зокрема з огляду на продуктивність, етичні аспекти та потребу контролю якості відповідей.

Методична база дослідження охопила застосування проектно-аналітичних і експериментальних методів.

У роботі використано: системний підхід до побудови архітектури веб-додатка (Next.js + GPT-API) із чітким розподілом функцій між фронтендом і бекендом; експериментальне тестування прототипу, що дало змогу оцінити релевантність відповідей моделі, середній час відгуку, частоту помилок і задоволеність користувачів; економічні розрахунки, пов'язані з тарифікацією викликів GPT і впровадженням механізмів кешування та управління навантаженням.

Практичні результати свідчать, що розроблений застосунок на базі Next.js й мовної моделі GPT (GPT-3.5/GPT-4) демонструє високий рівень зручності для користувачів та адекватний час відповіді (у середньому 2–3 секунди за помірного навантаження); дозволяє масштабувати систему завдяки можливостям App Router, серверних компонентів React та AI SDK, що спрощує інтеграцію з альтернативними LLM-провайдерами; потребує впровадження контент-фільтра й механізмів модерації для запобігання поширенню упередженого чи небажаного контенту, а також додаткових рішень для масштабування у випадку пікових навантажень.

Економічна доцільність підтверджена завдяки приблизним розрахункам, які вказують на можливість зменшити витрати на виклики GPT до 20–30% шляхом застосування механізмів кешування повторюваних запитів; адаптивного вибору моделі (GPT-3.5 / GPT-4) залежно від складності запиту; контролю над частотними «сплесками» навантаження через rate limiting та оптимізацію використання високопродуктивних обчислювальних ресурсів.

Наукова та практична цінність роботи виявляється в тому, що розроблений прототип може слугувати основою для реальних систем, зорієнтованих на автоматизоване консультування й аналіз текстової інформації (у технічній, освітній чи юридичній сферах). Запропонована методика оцінки якості роботи GPT у веб-застосунках, включно з аналітикою логів, вимірюванням швидкодії, вивченням досвіду користувачів, може бути використана при розробці інших проєктів з інтеграцією штучного інтелекту.

Власні пропозиції щодо розв’язання проблемних питань передбачають: формування «дорожньої карти» з етапами поступового вдосконалення системи (посилення кешування, підвищення безпеки, розширення інструментів модерації, тестування різних моделей GPT); розширення можливостей Next.js App Router для підтримки додаткових сценаріїв і мультимовної взаємодії, що особливо важливо для державних і міжнародних онлайн-сервісів; упровадження гнучких фінансових стратегій, що поєднують економне використання дорогих моделей GPT-4 і застосування дешевших аналогів для типових (FAQ) чи коротких запитів.

У цілому, результати виконаного дослідження дають підстави стверджувати, що інтеграція GPT у веб-застосунок на базі Next.js не лише технологічно можлива, а й економічно виправдана. Система здатна надавати релевантні відповіді з прийнятною швидкістю та рівнем безпеки, а впровадження рекомендованих покращень відкриває можливості для подальшого масштабування та адаптації до різноманітних галузей. Саме це робить представлені розробки значущими з погляду практичного застосування і перспективними в контексті подальших науково-технічних досліджень у сфері штучного інтелекту та обробки природної мови.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Живцова Л.І. Штучний інтелект: сутність та перспективи розвитку / Л.І. Живцова // Український журнал будівництва та архітектури. - 2023. - № 3. - С. 66-71. - Режим доступу: http://nbuv.gov.ua/UJRN/ujba_2023_3_11
2. Ковальчук О. Моделі машинного навчання для інформаційно-правового забезпечення діяльності суду. Збірник наукових праць “Приватне право і підприємництво”. 2024. Вип. 24. Ч. 1. С. 207–214. URL: <http://ppp-journal.kiev.ua/archive/2024/24/28.pdf>.
3. Піжук, О. І. (2019). Штучний інтелект як один із ключових драйверів цифрової трансформації економіки. Економіка, управління та адміністрування, (3 (89), 41—46.
4. Скрипник, С., & Шпатакова, О. (2023). Штучний інтелект як рушій розвитку цифрової економіки. Цифрова економіка та економічна безпека, (9 (09), 10—13.
5. Штучний інтелект у вищій освіті: ризики та перспективи інтеграції: матеріали всеукраїнського науково-педагогічного підвищення кваліфікації, 1 липня – 11 серпня 2024 року. – Львів – Торунь : Liha-Pres, 2024. 328 с.
6. Anthropic. Claude API Documentation, 2022. URL: <https://docs.anthropic.com/claude>
7. Barysè D. People’s Attitudes towards Technologies in Courts. Laws. 2022. Vol. 11. Article 71. DOI: <https://doi.org/10.3390/laws11050071>.
8. Bender E. M., et al. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? Proceedings of FAccT 2021, 2021.
9. Brown T. B., et al. Language Models are Few-Shot Learners. OpenAI, 2020. URL: <https://arxiv.org/abs/2005.14165>
10. Chen M., Yu T. Integrating AI Models into Modern Web Applications: A Guide with Next.js. Journal of Web Development Practices, 2022.
11. Devlin J., et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT 2019 Proceedings, 2019.

12. Ferrara E. Fairness and Bias in Artificial Intelligence: A Brief Survey of Sources, Impacts, and Mitigation Strategies. *Sci.* 2024. Vol. 6. Article 3. DOI: <https://doi.org/10.3390/sci6010003>
13. Guan H., Dong L., Zhao A. Ethical Risk Factors and Mechanisms in Artificial Intelligence Decision Making Behav. *Sci.* 2022. Vol. 12. Article 343. DOI: <https://doi.org/10.3390/bs12090343>
14. JMeter Project. Apache JMeter User Manual, 2022. URL: <https://jmeter.apache.org/>
15. Jurafsky D., Martin J. H. *Speech and Language Processing*. Pearson, 2019.
16. Kallhoff E. OAuth in NextAuth.js: Implementation Patterns. *Authentication Weekly*, 2022, 1(2), c. 17–26.
17. Kovalchuk O., Teremeckyi V. Informational and Legal Support of the Justice System. *Scientific Notes. Series: Law*. 2023. Vol. 15. Pp. 271–278. DOI: <https://pravo.cusu.edu.ua/index.php/pravo/article/view/378/397>.
18. Kruger D. AI SDK Management: Configuring GPT Models at Scale. *LLM Practical Review*, 2022, 3(2), c. 27–39.
19. Licardo J. T., Domjan M., Orehovački T. Intelligent Robotics—A Systematic Review of Emerging Technologies and Trends. *Electronics*. 2024. Vol. 13. Article 542. DOI: <https://doi.org/10.3390/electronics13030542>.
20. Liu H., Stein P. Mitigating Hallucinations in GPT-based Chatbots. *Computational Intelligence Reports*, 2022, 4(1), c. 88–99.
21. Ma L., Zhang Y., Wang T., Liu X., Ye W., Sun, C., Zhang S. Legal Judgment Prediction with Multi-Stage Case Representation Learning in the Real Court Setting. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* 2021. Virtual (pp. 993–1002). DOI: <https://doi.org/10.1145/3404835.3462945>.
22. Michel F., Rowland S. Cost Optimization Strategies for GPT-based APIs. *AI Systems Economics*, 2022, 3(1), c. 58–69.
23. Morales S. Data Encryption Practices in Cloud Databases. *Security & Privacy Journal*, 2021, 18(4), c. 33–45.

24. Müller R., Heinz K. Optimizing Deployment on Vercel: A Practical Guide. Cloud Computing Research, 2022, 5(2), c. 101–115.
25. Next.js Documentation. 2023. URL: <https://nextjs.org/docs>
26. Nielsen J. Usability and User Experience: A Field Study. Human-Computer Interaction Quarterly, 2020, 18(3), c. 41–55.
27. O’Neill R. Rate Limiting and Cache Strategies for AI-Driven Applications. Web Scalability Digest, 2023, 11(3), c. 120–134.
28. Oliveira A. Subject (in) Trouble: Humans, Robots, and Legal Imagination. Laws. 2020. Vol. 9. Article 10. DOI: <https://doi.org/10.3390/laws9020010>.
29. OpenAI. API Documentation for GPT-4, 2023. URL: <https://platform.openai.com/docs/>
30. OpenAI. GPT-3: Language Models at Scale. OpenAI Whitepaper, 2020. URL: <https://openai.com/blog/openai-api>
31. Petrov I., Sundberg A. React Server Components in Production: Lessons Learned. JS Frameworks Review, 2023, 9(1), c. 32–41.
32. Radford A., et al. Improving Language Understanding by Generative Pre-Training. OpenAI, 2018. URL: <https://openai.com/research/>
33. Raffel C., et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 2019.
34. Robles Carrillo, M. Artificial intelligence: From ethics to law. Telecommunications Policy. Elsevier. 2020. Vol. 44. Issue 6. DOI: <https://doi.org/10.1016/j.telpol.2020.101937>.
35. Rodgers T. Case Study: Implementing GPT-based Applications in Production. Software Engineering Cases, 2023, 2(1), c. 81–92.
36. Sentry. Sentry for Monitoring: A Comprehensive Overview, 2021. URL: <https://docs.sentry.io/>
37. Smith D. Next.js SSR Performance: A Comprehensive Benchmark. Web Engineering Review, 2022, 10(2), c. 73–85.
38. Tailwind Labs. Tailwind CSS Documentation, 2023. URL: <https://tailwindcss.com/docs>

39. Teremetskyi V. I., Kovalchuk O. Ya. Artificial Intelligence as a Factor in the Digital Transformation of the Justice System. Forum Prava. 2024. № 78(1), Pp. 106–115. DOI: <http://doi.org/10.5281/zenodo.10870779>
40. Teremetskyi V. I., Kovalchuk O. Ya. Artificial Intelligence as a Factor in the Digital Transformation of the Justice System. Forum Prava. 2024. № 78(1), Pp. 106–115. DOI: <http://doi.org/10.5281/zenodo.10870779>
41. Vaswani A., et al. Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 2017.
42. Vercel. Blob Storage Guide, 2023. URL: <https://vercel.com/docs/storage/blob>
43. W3C. Web Performance Working Group: Metrics and Recommendations, 2023. URL: <https://www.w3.org/webperf/>
44. Wang D. Towards Ethical Engineering: Artificial Intelligence as an Ethical Governance Tool for Emerging Technologies. Comput. Sci. Math. Forum. 2023. Vol. 8. Article 76. DOI: <https://doi.org/10.3390/cmsf2023008076>
45. Wang P., et al. Efficient Implementation of Transformer Models in Web Frameworks. Journal of Artificial Intelligence Integration, 2022.
46. Xie Y., et al. Context Management in GPT Models. Transactions on Neural Networks and Learning Systems, 2021.

ДОДАТКИ

Додаток А

Filters

Reset

▼ Timeline

Maximum (1 hour)

▼ Contains Level

☐ Warning

0

☐ Error

0

> Environment

> Route

> Request Path

> Status Code

> Resource

> Type

> Host

> Request Method

> Cache

> Branch

▼ 42 total logs found...

Time	Status	Host	Request	Messages
JAN 05 16:49:10.68	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:49:10.66	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:49:10.66	GET 200	nextjs-a1-chatbot-git-m...	/api/history	
JAN 05 16:49:10.41	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:49:10.33	GET 302	nextjs-a1-chatbot-git-m...	/login	
JAN 05 16:49:09.88	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:49:09.79	GET 302	nextjs-a1-chatbot-git-m...	/login	
JAN 05 16:49:00.16	POST 200	nextjs-a1-chatbot-git-m...	/login	
JAN 05 16:49:00.96	GET 200	nextjs-a1-chatbot-git-m...	/register	
JAN 05 16:49:00.55	GET 200	nextjs-a1-chatbot-git-m...	/login	
JAN 05 16:49:00.48	GET 307	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:49:00.24	POST 200	nextjs-a1-chatbot-git-m...	/api/auth/signout	
JAN 05 16:48:59.79	GET 200	nextjs-a1-chatbot-git-m...	/api/auth/csrf	
JAN 05 16:48:56.78	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:48:56.78	GET 200	nextjs-a1-chatbot-git-m...	/api/history	

Logs

JAN 05 16:48:41.87	GET 200	nextjs-a1-chatbot-git-m...	/favicon.ico	
JAN 05 16:48:40.96	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:48:40.62	GET 200	nextjs-a1-chatbot-git-m...	/api/history	
JAN 05 16:48:40.61	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:48:40.13	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:48:39.99	GET 200	nextjs-a1-chatbot-git-m...	/favicon.ico	
JAN 05 16:48:39.56	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:48:39.56	GET 200	nextjs-a1-chatbot-git-m...	/api/history	
JAN 05 16:48:38.98	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:48:36.91	GET 200	nextjs-a1-chatbot-git-m...	/favicon.ico	
JAN 05 16:48:36.89	GET 200	nextjs-a1-chatbot-git-m...	/api/history	
JAN 05 16:48:36.88	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:48:36.48	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:48:35.96	GET 200	nextjs-a1-chatbot-git-m...	/favicon.ico	
JAN 05 16:48:35.93	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:48:35.93	GET 200	nextjs-a1-chatbot-git-m...	/api/history	
JAN 05 16:48:35.49	GET 200	nextjs-a1-chatbot-git-m...	/	
JAN 05 16:48:34.87	GET 200	nextjs-a1-chatbot-git-m...	/api/history	
JAN 05 16:48:34.87	GET 200	nextjs-a1-chatbot-git-m...	/api/vote	
JAN 05 16:48:32.70	GET 200	nextjs-a1-chatbot-git-m...	/	

No more logs to show within selected timeline

GET /

200

Request started

JAN 05 16:49:10.68 GMT+2

Request ID

g7sp4-1736088550680-66aedfeb0844

Path

/

Host

nextjs-a1-chatbot-git-main-andrii-ko...

User Agent

Mozilla/5.0 (Macintosh; Intel M...

Firewall

Allowed

Middleware

200

Response finished

Deployment Information

Deployment ID

dpl_Azahbcg83oSnZcKPidrcTMs6b...

