Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Інтеграція LLM у веб додаток для вирішення задачі генерації відповідей на запити користувачів

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Антонюк Д.Т.
(підпис) (прізвище та ініціали)

Керівник _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Никитюк. В.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«13» квітня 2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Антонюку Дмитру Тарасовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Інтеграція LLM у веб додаток для вирішення задачі генерації відповідей на запити користувачів

Керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від « 10 » березня 2026 року № 4/9-150
2. Термін подання студентом завершеної роботи 14 травня 2026 р.
3. Вихідні дані до роботи Науково-технічні публікації (друковані та розміщені в Інтернеті)
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області. 1.1 Огляд сучасних мовних моделей та їх архітектури 1.2 Методи інтеграції LLM у веб-додатки (API, мікросервіси). 1.3 Переваги та обмеження використання LLM у веб середовищі. 1.4 Висновок до першого розділу. 2 Підходи та методи проєктування систем генерації відповідей у вебдодатках. 2.1 Постановка задачі та вибір технологій для реалізації. 2.2Обгрунтування вибору архітектури системи генерації відповідей 2.3 Загальна архітектура системи з інтегрованим LLM. 2.4 Реалізація механізмів генерації відповідей на запити користувачів. 2.5 Висновки до другого розділу. 3 Реалізація вебдодатка та інтеграція модуля автоматизації відповідей. 3.1 Експериментальні дослідження та методика тестування якості відповідей LLM. 3.2 Порівняння роботи системи з традиційними методами пошуку. 3.3 програмна реалізація та опис інструкції користувача. 3.4 Оптимізація алгоритмів пошуку інформації на вебсайті за допомогою машинного навчання. 3.5 Забезпечення продуктивності та безпеки вебдодатку. 3.6 Висновки до третього розділу. 4 Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Обов'язкові медичні огляди працівників певних категорій. 4.2 Способи і засоби пожежогасіння. Висновки. Перелік джерел. Додатки
5. Перелік графічного матеріалу
1. Титульна сторінка. 2. Актуальність роботи. 3. Мета, обєкт, предмет досідження та наукова новизна.4. Аналіз предметної області та теоретичні основи інтеграції LLM. 5. Основні етапи процесу промпт-інжинірингу. 6. Вибір сервісу інтеграції LLM. 7. Постановка задачі. 8. Архітектура системи. 9. Механізм обробки запиту. 10. Тестування системи. 1.1 Реалізація вебдодатка Urban Wear. 12. Продуктивність та безпека. 13. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 13 квітня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	13.04.2026	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір	13.04.2026-20.04.2026	*Виконано*
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	21.04.2026-03.05.2026	*Виконано*
	Роботи		
4.	Оформлення розділу	04.05.2026-10.05.2026	*Виконано*
	«Аналіз предметної області та теоретичні основи»		
5.	Оформлення розділу « Проєктування та реалізації	04.05.2026-10.05.2026	*Виконано*
	Веб-додатку»		
	»		
6.	Оформлення розділу « Експериментальна оцінка та	04.05.2026-10.05.2026	*Виконано*
	Аналіз результатів»		
7.	Виконання завдання до підрозділу «Охорона праці»	27.04.2026-10.05.20	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	27.04.2026-10.05.20	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	11.05.2026-13.05.2026	*Виконано*
10.	Нормоконтроль	14.05.2026	*Виконано*
11.	Перевірка на плагіат	15.05.2026	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.05.2026	*Виконано*
13.	Захист кваліфікаційної роботи	26.05.2026	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Інтеграція LLM у веб додаток для вирішення задачі генерації відповідей на запити користувачів// Кваліфікаційна робота освітнього рівня «Магістр» // Антонюк Дмитро Тарасович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2026 // С. 78, рис. – 13, табл. – 3, додат. – 2, бібліогр. – 54.

Ключові слова: великі мовні моделі, Azure OpenAI, вебдодаток, генерація відповідей, API, штучний інтелект, клієнт-серверна архітектура, автоматизація обробки запитів.

Кваліфікаційна робота присвячена дослідженню та розробленню вебдодатка з інтеграцією великих мовних моделей для автоматизації генерації відповідей на запити користувачів.

У першому розділі кваліфікаційної роботи здійснено огляд сучасних великих мовних моделей та їх застосування у вебсередовищі. Розглянуто методи інтеграції LLM через API та мікросервісні архітектури, проаналізовано можливості Azure OpenAI та переваги використання мовних моделей для автоматизації взаємодії з користувачем. На основі цього обґрунтовано вибір технологій і визначено вимоги до системи.

Другий розділ присвячено архітектурі та функціональній структурі вебдодатка. Описано механізми обробки й передачі запитів користувачів, спроєктовано взаємодію між клієнтською частиною, сервером та AI-сервісом.

У третьому розділі представлено реалізацію вебдодатка з чат-інтерфейсом, каталогом товарів, кошиком та особистим кабінетом. Здійснено інтеграцію із сервісом Azure OpenAI для генерації відповідей у режимі реального часу, розроблено механізми асинхронної обробки запитів. Проведено тестування працездатності системи та оцінено якість відповідей, що підтвердило ефективність запропонованих рішень.

ANNOTATION

Integration of LLM into a Web Application for Solving the Problem of Generating Responses to User Requests // Qualification Thesis for the Master's Degree // Antoniuk Dmytro Tarasovych // Ivan Puluj Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group СНнм-61 // Ternopil, 2026 // P. 78, fig. – 13, tabl. – 3, append. – 2, bibliography – 59.

Keywords: large language models, Azure OpenAI, web application, response generation, API, artificial intelligence, client-server architecture, automation of request processing.

The qualification work is devoted to the research and development of a web application with the integration of large language models to automate the generation of responses to user requests.

The first section of the qualification work reviews modern large language models and their application in the web environment. Methods for integrating LLM via API and microservice architectures are considered, the capabilities of Azure OpenAI and the advantages of using language models to automate user interaction are analyzed. Based on this, the choice of technologies is justified and the requirements for the system are determined.

The second section is devoted to the architecture and functional structure of the web application. The mechanisms for processing and transmitting user requests are described, the interaction between the client part, the server and the AI-service is designed.

The third section presents the implementation of a web application with a chat interface, product catalog, shopping cart and personal account. Integration with the Azure OpenAI service is carried out to generate responses in real time, and mechanisms for asynchronous request processing are developed. The system's performance was tested and the quality of the responses was assessed, which confirmed the effectiveness of the proposed solutions.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект.

AI (англ. Artificial Intelligence) – технології штучного інтелекту для автоматизованої обробки інформації.

AJAX (англ. Asynchronous JavaScript and XML) – асинхронна технологія обміну даними між клієнтом і сервером без перезавантаження сторінки.

API (англ. Application Programming Interface) – програмний інтерфейс для взаємодії між різними програмними компонентами.

Azure OpenAI – хмарний сервіс Microsoft для роботи з великими мовними моделями.

CRUD (англ. Create, Read, Update, Delete) – базові операції над даними в інформаційних системах.\

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів для оформлення вебсторінок.

DOM (англ. Document Object Model) – об’єктна модель документа у вебпрограмуванні.

FAQ (англ. Frequently Asked Questions) – розділ поширених запитань.

GPT (англ. Generative Pre-trained Transformer) – тип великої мовної моделі для генерації тексту.

HTML (англ. HyperText Markup Language) – мова розмітки гіпертексту.

HTTP (англ. HyperText Transfer Protocol) – протокол передачі гіпертексту.

HTTPS (англ. HyperText Transfer Protocol Secure) – безпечний протокол передачі гіпертексту.

ID (англ. Identifier) – унікальний ідентифікатор елемента або запису.

JSON (англ. JavaScript Object Notation) – формат обміну структурованими даними.

JS (англ. JavaScript) – мова програмування для веброзробки.

LLM (англ. Large Language Model) – велика мовна модель для обробки та генерації природної мови.

MVC (англ. Model-View-Controller) – шаблон архітектури програмного забезпечення.

PHP (англ. PHP: Hypertext Preprocessor) – мова серверного програмування для веброзробки.

Prompt Engineering – процес формування та оптимізації запитів до мовної моделі.

REST (англ. Representational State Transfer) – архітектурний стиль для побудови API.

SEO (англ. Search Engine Optimization) – пошукова оптимізація вебресурсів.

SPA (англ. Single Page Application) – односторінковий вебдодаток із динамічним оновленням контенту.

SQL (англ. Structured Query Language) – мова структурованих запитів до баз даних.

Tailwind CSS – CSS-фреймворк для створення адаптивного користувацького інтерфейсу.

UI (англ. User Interface) – інтерфейс користувача.

URL (англ. Uniform Resource Locator) – уніфікований локатор ресурсу в інтернеті.

UX (англ. User Experience) – досвід взаємодії користувача з системою.

WebSocket – технологія двостороннього обміну даними між клієнтом і сервером у режимі реального часу.

XML (англ. eXtensible Markup Language) – розширювана мова розмітки.

ЗМІСТ

ВСТУП

Актуальність теми. З розвитком штучного інтелекту та стрімким поширенням великих мовних моделей (LLM) змінюються підходи до взаємодії користувачів із вебдодатками. Сучасні сервіси дедалі частіше потребують автоматизованої обробки запитів користувачів, швидкого формування відповідей та підтримки природної мови спілкування. Інтеграція LLM, зокрема через хмарні платформи на кшталт Azure OpenAI, дозволяє значно підвищити якість користувацького досвіду, зменшити навантаження на підтримку та забезпечити персоналізовану взаємодію. Водночас постають задачі ефективної інтеграції моделей у вебархітектуру, оптимізації запитів та контролю якості відповідей, що робить дану тему актуальною для сучасної інженерії програмного забезпечення.

Мета і задачі дослідження. Метою кваліфікаційної роботи є розробка та інтеграція LLM у вебдодаток для автоматизації генерації відповідей на запити користувачів із використанням сервісу Azure OpenAI.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасні підходи до використання великих мовних моделей у вебдодатках;

–дослідити можливості платформи Azure OpenAI та її API;

–розробити архітектуру вебдодатку з інтеграцією LLM;

– реалізувати механізм обробки користувацьких запитів і генерації відповідей;

– оцінити ефективність та якість отриманих результатів роботи системи.

Об’єкт дослідження: процеси автоматизованої обробки природномовних запитів користувачів у вебдодатках.

Предмет дослідження: методи та засоби інтеграції великих мовних моделей (LLM) у вебдодатки з використанням Azure OpenAI API.

Наукова новизна одержаних в кваліфікаційній роботі результатів полягає у розробці підходу до інтеграції LLM у вебдодаток із застосуванням

Azure OpenAI для автоматизованої генерації контекстно-залежних відповідей користувачам та оптимізації взаємодії “користувач–система”.

Практичне значення одержаних результатів. Розроблено прототип вебдодатку з інтегрованою LLM-системою, що дозволяє автоматично формувати відповіді на запити користувачів у режимі реального часу. Результати можуть бути використані для створення чат-ботів, систем підтримки користувачів та інтелектуальних вебсервісів.

Апробація результатів магістерської роботи. Основні результати проведених досліджень апробовано на XIV науково-технічній конференції «Актуальні задачі сучасних технологій» та XIII міжнародній студентській науково-практичній конференції молодих учених і студентів «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2025 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (додатки А1 та А2).

1 АНАЛІЗ ПРЕДМЕТНОЇ ТА ТЕОРЕТИЧНІ ОСНОВИ ІНТЕГРАЦІЇ LLM

1.1 Огляд сучасних мовних моделей та їх архітектури

Сучасне вебсередовище стрімко трансформується під впливом розвитку цифрових технологій та зростання обсягів інформації, з якою взаємодіють користувачі. Вебдодатки дедалі частіше виконують не лише функцію відображення даних, але й забезпечують інтелектуальну обробку запитів, надаючи користувачам змістовні та контекстно релевантні відповіді. У зв’язку з цим особливого значення набувають технології обробки природної мови, що дозволяють системам ефективно взаємодіяти з людиною у звичній для неї формі.

Традиційні підходи до обробки текстових даних, засновані на використанні ключових слів або простих лінгвістичних правил, поступово втрачають свою ефективність через складність сучасних запитів та необхідність врахування контексту. Користувачі все частіше формулюють запити у вигляді повноцінних речень або навіть діалогів, очікуючи отримати не просто набір результатів, а чітку, зрозумілу та логічно побудовану відповідь. Це створює потребу у впровадженні більш складних інтелектуальних моделей, здатних аналізувати семантику тексту та враховувати контекст взаємодії.

Одним із найперспективніших напрямів у цій сфері є використання великих мовних моделей. Такі моделі являють собою складні нейронні мережі, навчені на великих масивах текстових даних, що дозволяє їм розуміти структуру мови, виявляти закономірності та генерувати зв’язні текстові відповіді. На відміну від класичних алгоритмів, великі мовні моделі здатні працювати з контекстом, враховувати попередні повідомлення у діалозі та формувати відповіді, максимально наближені до природної людської мови [10].

Важливою особливістю великих мовних моделей є їх універсальність. Вони можуть використовуватися для широкого спектра задач, зокрема генерації тексту, автоматичного перекладу, узагальнення інформації, відповіді на запитання та підтримки діалогових систем. У контексті вебдодатків це відкриває нові можливості для автоматизації обробки запитів користувачів, створення інтелектуальних чат-ботів та підвищення загального рівня взаємодії між системою та користувачем [11].

Застосування великих мовних моделей дозволяє значно підвищити якість обслуговування користувачів за рахунок швидкого формування відповідей, адаптації до різних стилів запитів та здатності працювати з неструктурованими даними. Водночас їх використання потребує врахування певних обмежень, зокрема можливості генерації неточних або некоректних відповідей, що вимагає додаткового контролю якості та налаштування процесів взаємодії [11].

В основі архітектури сучасних великих мовних моделей лежить нейромережева архітектура Transformer (Трансформер), яка базується на механізмі самоуваги (Self-Attention). Цей механізм дозволяє моделі аналізувати зв'язки між усіма токенами в тексті одночасно, незалежно від їхнього розташування, що забезпечує глибоке розуміння контексту. Архітектурно моделі поділяються на енкодери (для аналізу та класифікації), декодери (для генерації тексту, такі як серія GPT чи Llama) та комбіновані системи.

Великі мовні моделі мають важливе значення у трансформації сучасних вебсервісів, оскільки дозволяють перейти від статичних способів подання інформації до інтерактивних і динамічних форм взаємодії. Завдяки здатності аналізувати текстові запити та генерувати змістовні відповіді, такі моделі використовуються для створення чат-ботів, віртуальних асистентів та систем підтримки користувачів. Це дозволяє значно зменшити навантаження на персонал та підвищити швидкість обслуговування. Користувач може задавати питання у довільній формі, використовуючи розмовний стиль, а система, у свою чергу, здатна інтерпретувати зміст запиту та надати релевантну відповідь. Це

суттєво покращує користувацький досвід та робить взаємодію з вебресурсом більш інтуїтивною [12].

Використання LLM дозволяє підвищити рівень персоналізації сервісів, оскільки система може враховувати контекст запиту та адаптувати відповіді під потреби користувача. У порівнянні з традиційними підходами до обробки запитів, великі мовні моделі забезпечують більш гнучку та ефективну взаємодію, що узагальнено представлено в таблиці 1.1.

Таблиця 1.1 – Порівняння традиційних систем та систем на основі LLM

Характеристика	**Традиційні системи**	**Системи на основі LLM**
Тип відповідей	Шаблонні	Генеровані, адаптивні
Розуміння запиту	За ключовими словами	З урахуванням контексту
Персоналізація	Обмежена	Висока
Гнучкість взаємодії	Низька	Висока
Обробка складних запитів	Ускладнена	Ефективна

Сучасні великі мовні моделі інтегруються у різноманітні сфери життєдіяльності завдяки своїй унікальній здатності до глибокої обробки природної мови та генерації логічно вибудованих, контекстуальних відповідей. Їхня технологічна універсальність дозволяє використовувати цей інструментарій для побудови нового покоління інтелектуальних систем взаємодії.

В електронній комерції у межах сучасних інтернет-магазинів LLM виступають не просто як чат-боти, а як повноцінні віртуальні консультанти, здатні аналізувати складні запити, порівнювати характеристики товарів та надавати персоналізовані рекомендації на основі вподобань користувача [13].

Це сприяє індивідуалізації процесу обслуговування, дозволяючи клієнтам отримувати додаткові роз’яснення без затримок [14].

У напрямі технічної та клієнтської підтримки впровадження LLM дозволяє радикально оптимізувати бізнес-процеси. Віртуальні асистенти, інтегровані у вебсервіси, забезпечують безперервну підтримку користувачів у режимі 24/7, ефективно опрацьовуючи типові звернення та вирішуючи багатоетапні запити без залучення операторів. Це не лише мінімізує час очікування для клієнта, а й дозволяє компаніям зосередити людські ресурси на вирішенні критичних або нестандартних завдань, що потребують глибокої експертизи [15].

Крім того, великі мовні моделі стали незамінним інструментом у сфері контент-менеджменту вебресурсів. Вони демонструють високу ефективність при генерації описів товарів, створенні інформаційних статей та розробці технічних інструкцій. Автоматизація рутинних завдань із написання текстів дозволяє значно прискорити наповнення вебресурсу та забезпечити високу стилістичну однорідність контенту при великих обсягах публікацій. Велика аналітика відповідей стає джерелом цінних інсайтів для бізнесу, дозволяючи відстежувати ринкові тенденції та приймати обґрунтовані рішення на основі об’єктивних даних.

1.2 Методи інтеграції LLM у веб-додатки (API, мікросервіси)

Інтеграція великих мовних моделей у вебдодатки передбачає використання комплексу технологій та підходів, що охоплюють як серверну, так і клієнтську частину системи, а також засоби взаємодії з зовнішніми сервісами обробки природної мови. Вибір підходу до використання LLM залежить від вимог до продуктивності, масштабованості, рівня контролю над даними та необхідності врахування контексту користувацьких запитів. Основною метою є забезпечення ефективної генерації відповідей у межах

вебінтерфейсу з урахуванням сучасних вимог до швидкості та якості обслуговування.

Одним із найбільш поширених підходів є використання API для взаємодії з великими мовними моделями. У цьому випадку вебдодаток надсилає запити до зовнішніх сервісів, які виконують обробку тексту та повертають згенеровану відповідь. Такий підхід реалізується за допомогою REST API і підтримується хмарними платформами, такими як Azure OpenAI Service або OpenAI. Для передачі запитів використовуються стандартні інструменти, зокрема Fetch API або бібліотека Axios на клієнтській стороні, а також серверні HTTP-клієнти. Перевагою цього підходу є простота інтеграції та доступ до потужних моделей, однак він передбачає залежність від зовнішніх сервісів і мережевих затримок [18].

Іншим підходом є використання чат-орієнтованих інтерфейсів, які дозволяють реалізувати взаємодію з користувачем у форматі діалогу. Такий підхід активно застосовується у сучасних вебдодатках, де користувач взаємодіє із системою через текстові повідомлення. Для реалізації інтерфейсу використовуються фронтенд-технології, такі як React або Vue.js, що дозволяють створювати динамічні та інтерактивні компоненти. Обмін даними між клієнтом і сервером здійснюється асинхронно, що забезпечує швидке оновлення інформації без перезавантаження сторінки [18].

Також поширеним є підхід із реалізацією обробки запитів на серверній стороні вебдодатку. У цьому випадку сервер виступає посередником між користувачем та мовною моделлю, виконуючи попередню обробку запитів, формування структури звернення до моделі та подальшу обробку отриманої відповіді. Для цього використовуються серверні фреймворки, такі як Django, Flask або Laravel. Такий підхід дозволяє реалізувати додаткові механізми перевірки, фільтрації та логування запитів, що підвищує контроль над системою та її безпеку [18].

Окремо варто виділити підхід, при якому велика мовна модель інтегрується як окремий компонент або мікросервіс у структурі вебдодатку. У

цьому випадку LLM розглядається як незалежний модуль чи сервіс, що розгортається локально або в ізольованому контейнері й взаємодіє з іншими частинами системи через API або внутрішні сервіси. Такий підхід часто використовується в мікросервісній архітектурі та дозволяє забезпечити гнучкість, масштабованість і можливість незалежного оновлення компонентів системи [18].

Варіативність методологій інтеграції великих мовних моделей у структуру вебдодатків зумовлена необхідністю адаптації системи до конкретних технічних викликів та функціональних вимог. Вибір тієї чи іншої архітектурної схеми базується на детальному аналізі складності поставлених завдань, очікуваних показників продуктивності та пріоритетів щодо забезпечення якісної та безпечної взаємодії з кінцевим користувачем. Використання конкретного інструментарію в межах обраного підходу дозволяє реалізувати гнучкі рішення, що відповідають специфіці розроблюваного програмного продукту.

Ефективність роботи великих мовних моделей (LLM) значною мірою залежить від якості сформованих запитів, які передаються до системи. Процес створення таких запитів отримав назву prompt engineering і є критично важливим етапом інтеграції нейромереж у сучасні вебдодатки [19]. Цей процес не є одноразовою дією, а являє собою послідовність логічних кроків: від постановки завдання до фінального коригування результату, як це схематично зображено на рисунку 1.1.

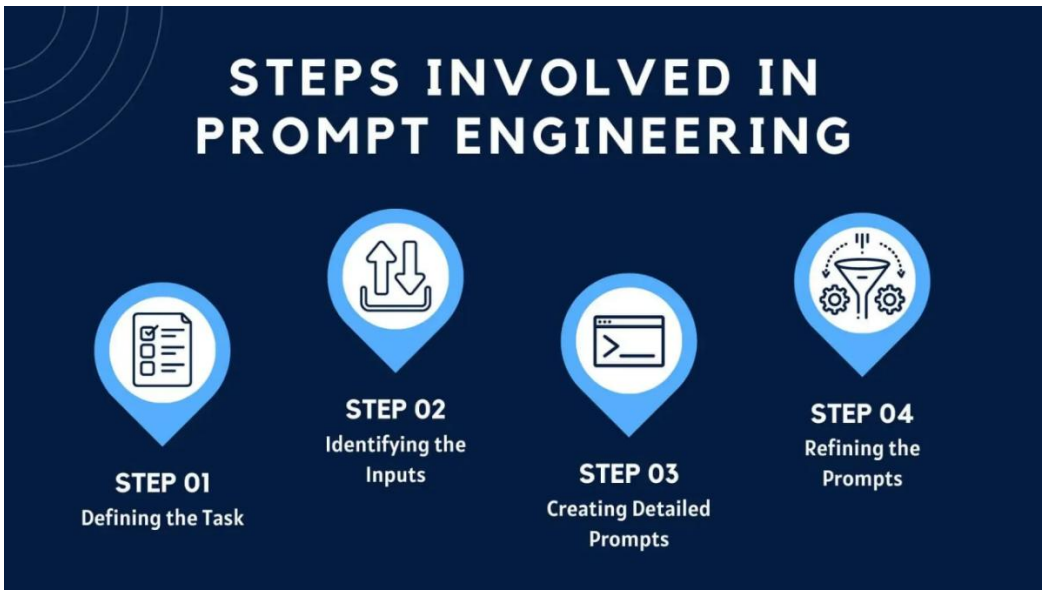

Рисунок 1.1 – Основні етапи процесу промпт-інжинірингу (Steps involved in Prompt Engineering)

Згідно з наведеною схемою, першим етапом є чітке визначення завдання. Запит повинен містити конкретну інструкцію для моделі, без двозначностей або зайвої інформації. Наприклад, замість загального «Розкажи про товар», варто використовувати конкретний варіант: «Опиши основні характеристики смартфона для покупця», що дозволяє отримати структуровану відповідь.

Наступним кроком є ідентифікація вхідних даних та врахування контексту. У вебдодатках запити користувачів часто є частиною діалогу, тому система повинна передавати історію повідомлень разом із поточним запитом. Це забезпечує логічну послідовність взаємодії, що підтримується більшістю сучасних API, зокрема Azure OpenAI Service.

Для створення деталізованих запитів застосовуються системні інструкції, які визначають роль, стиль та поведінку моделі. Наприклад, можна задати формальний тон мовлення або надання інформації виключно у вигляді списку. Також ефективним є підхід структурованих запитів, де інформація передається чіткими блоками: окремо інструкція, окремо контекст та безпосередньо питання [20].

Завершальним етапом є вдосконалення промпту. Порівнюючи різні варіанти, можна побачити прогрес: від простого «Поясни, що таке вебдодаток» до професійного запиту з контекстом «Поясни, що таке вебдодаток для студента-початківця, наведи приклад і опиши його функції». Деталізація та постійна ітерація запиту безпосередньо впливають на точність, релевантність та повноту згенерованої відповіді [20]. Системний підхід до формування та обробки запитів є ключовим фактором ефективного використання LLM, що дозволяє забезпечити якісну та зрозумілу взаємодію з користувачем у вебдодатку.

Сучасні великі мовні моделі зазвичай не розгортаються безпосередньо у вебдодатках, а використовуються через спеціалізовані хмарні сервіси, які надають доступ до них через API. Такий підхід дозволяє значно спростити процес інтеграції, зменшити вимоги до обчислювальних ресурсів та забезпечити доступ до актуальних версій моделей без необхідності їх самостійного оновлення.

Одним із найбільш поширених рішень є Azure OpenAI Service, який надає можливість інтеграції великих мовних моделей у вебдодатки з використанням інфраструктури Microsoft Azure. Даний сервіс забезпечує високу масштабованість, безпеку та контроль доступу, що є важливим для корпоративних застосунків. Він підтримує генерацію тексту, роботу з діалогами, а також можливість налаштування поведінки моделей під конкретні задачі [24].

Іншим популярним рішенням є API від OpenAI, яке дозволяє інтегрувати мовні моделі у вебдодатки через стандартні HTTP-запити [25].

Серед альтернативних інструментів варто відзначити платформу Google Cloud, яка пропонує власні рішення для обробки природної мови та інтеграції моделей у вебсервіси [26]. Також існують відкриті моделі, які можуть бути розгорнуті локально або у хмарі як мікросервіси, однак такий підхід потребує значних ресурсів і складнішого налаштування.

Для узагальнення характеристик основних сервісів доцільно представити їх у вигляді таблиці.

Таблиця 1.2 – Порівняння сервісів для інтеграції LLM у вебдодатки

Сервіс	Основні можливості	Особливості
Azure OpenAI Service	Генерація тексту, чат-інтерфейси, інтеграція з хмарою	Висока безпека, масштабованість, інтеграція з Azure
OpenAZURE OPENAI	Генерація тексту, обробка запитів, швидка інтеграція	Простота використання, широка документація
Google Cloud	NLP-сервіси, обробка тексту	Інтеграція з екосистемою Google
Локальні моделі	Повний контроль над системою	Високі вимоги до ресурсів, складність налаштування

Окрім безпосередніх сервісів доступу до мовних моделей, важливу роль відіграють допоміжні інструменти, які спрощують інтеграцію та використання LLM у вебдодатках. До них належать бібліотеки для роботи з API, засоби обробки запитів, а також фреймворки для побудови інтерфейсу користувача. Наприклад, для реалізації клієнтської частини можуть використовуватися React або Vue.js, а для серверної логіки – Django чи Laravel.

Застосування сучасних сервісів та інструментарію дозволяє здійснювати ефективну інтеграцію великих мовних моделей у структуру вебдодатків, уникаючи при цьому необхідності складного технічного налаштування внутрішньої інфраструктури. Саме поєднання потужних хмарних платформ із актуальними вебтехнологіями виступає основою для створення гнучких та масштабованих систем, що здатні забезпечити високу якість і швидкість роботи модулів автоматичної генерації відповідей у режимі реального часу.

1.3 Переваги та обмеження використання LLM у веб-середовищі

Інтеграція великих мовних моделей (Large Language Models, LLM) у сучасне веб-середовище стала одним із ключових драйверів трансформації клієнт-серверної архітектури та користувацького досвіду. Використання LLM у веб-додатках дозволяє автоматизувати генерацію контенту, створювати адаптивні інтерфейси та забезпечувати персоналізовану взаємодію в режимі реального часу. Однією з головних переваг інтеграції LLM є кардинальне покращення процесу генерації програмного коду на стороні сервера та автоматизації розробки. Завдяки здатності моделей інтерпретувати природну мову, розробники веб-систем отримали можливість впроваджувати фреймворки, що уточнюють специфікації вимог безпосередньо під час діалогу з користувачем. Це мінімізує синтаксичні та логічні помилки у генерованих скриптах, оптимізуючи загальну архітектуру програмних рішень [16].

Поряд із покращенням бекенд-процесів, використання штучного інтелекту у веб-просторі супроводжується серйозними викликами у сфері кібербезпеки, які виступають головним обмеженням технології. Найбільш критичною загрозою для промислових веб-систем є атаки типу Prompt Injection (ін'єкції операторів), коли зловмисник через поле введення даних маніпулює контекстом запиту, змушуючи модель ігнорувати системні інструкції. Це може призвести до витоку конфіденційних даних користувачів, несанкціонованого доступу до баз SQL або виконання шкідливого коду. Розробка надійних механізмів фільтрації, ізоляції контексту та дворівневої верифікації запитів є обов'язковою умовою для безпечного розгортання LLM у веб-інфраструктурі [17].

Для досягнення максимальної ефективності веб-орієнтованих систем стандартні базові моделі часто виявляються недостатньо точними у вузьких предметних областях. Перевагою сучасного інжинірингу є наявність гнучких підходів до адаптації штучного інтелекту під конкретні бізнес-завдання. Існує

чотири базові методології створення та налаштування спеціалізованих LLM для веб-додатків: оперативне проектування (prompt engineering), пошукова оптимізація (Retrieval-Augmented Generation, RAG), тонке налаштування (fine-tuning) та навчання моделі з нуля. Вибір конкретного підходу дозволяє розробникам балансувати між фінансовими витратами та глибиною експертних знань, які система демонструє під час взаємодії з кінцевим користувачем сайту [18].

Найбільш доступним та економічно вигідним інструментом керування поведінкою моделей у веб-середовищі є промпт-інжиніринг, перевага якого полягає у відсутності необхідності перенавчання нейромережі. Чітко сформульовані текстові інструкції, контекстні обмеження та приклади очікуваних відповідей дозволяють веб-додатку гнучко підлаштовувати логіку відповідей під індивідуальні запити користувачів, виступаючи сполучною ланкою між динамічним інтерфейсом та ядром штучного інтелекту [19]. Грамотне проектування запитів безпосередньо підвищує продуктивність AI-моделей, мінімізуючи час відповіді сервера (latency) та зменшуючи кількість токенів, що знижує витрати на використання хмарних API, таких як Azure OpenAI [20].

Незважаючи на потужні інтелектуальні можливості LLM, вони функціонують виключно як логічне ядро або інтелектуальний сервіс, що накладає обмеження на їхнє безпосереднє сприйняття користувачем. Для забезпечення повноцінного функціонування системи результати роботи моделі мають бути візуалізовані та інтегровані у клієнтську частину. Веб-інтерфейс (фронтенд) виступає необхідним посередником, який трансформує текстові відповіді штучного інтелекту у структуровані елементи сторінки, форми, графіки або інтерактивні тренажери, забезпечуючи зручність та доступність взаємодії (UX/UI) [21].

Важливим обмеженням та водночас предметом наукових дискусій є проблема оцінювання релевантності та точності відповідей LLM у веб-системах. Оскільки моделі генерують умовно-випадковий текст, традиційні

детерміновані методи тестування програмного забезпечення тут неефективні. Для валідації відповідей впроваджують інноваційні підходи, зокрема системи автоматичного оцінювання (autoraters), де одна LLM виступає експертом для аналізу та валідації вихідних даних іншої моделі на основі заданих критеріїв безпеки, тональності та відповідності контексту [22]. Комплексна оцінка ефективності великих мовних моделей у веб-середовищі потребує поєднання автоматизованих метрик, моніторингу користувацьких сесій та аналізу бізнес-показників, що дозволяє своєчасно виявляти галюцинації моделі та оптимізувати алгоритми обробки даних перед їх фінальним виведенням на екран користувача [23].

1.4 Висновок до першого розділу

У першому розділі було проведено аналіз підходів до інтеграції великих мовних моделей у вебдодатки для автоматизації обробки та генерації відповідей на запити користувачів. Розглянуто сутність великих мовних моделей як інструменту обробки природної мови, їх архітектуру, значення у сучасних вебсервісах, а також основні галузі застосування.

Проаналізовано підходи до використання LLM у вебдодатках, зокрема інтеграцію через API, мікросервісну та серверну обробку запитів, а також використання чат-інтерфейсів. Окрему увагу приділено методам формування запитів до моделей, що включають принципи prompt engineering та врахування контексту для підвищення якості відповідей.

Розглянуто особливості взаємодії користувача з мовними моделями у вебсередовищі, що характеризуються діалоговим форматом, різноманітністю запитів та залежністю результату від способу формулювання запиту. Також визначено переваги, обмеження, основні критерії та методи оцінювання якості відповідей, що дозволяють комплексно аналізувати ефективність роботи системи.

Крім того, здійснено огляд сучасних сервісів та інструментів для інтеграції LLM, зокрема Azure OpenAI Service та OpenAI, що надають зручні засоби для реалізації функцій генерації тексту у вебдодатках.

Отримані результати аналізу створюють теоретичну основу для подальшого проєктування системи генерації відповідей на основі великих мовних моделей, що буде розглянуто у наступному розділі.

2 ПІДХОДИ ТА МЕТОДИ ПРОЄКТУВАННЯ СИСТЕМ ГЕНЕРАЦІЇ ВІДПОВІДЕЙ У ВЕБДОДАТКАХ

2.1 Постановка задачі та вибір технологій для реалізації

У межах виконання цієї роботи постає завдання розробити сучасний вебдодаток інтернет-магазину, ключовою особливістю якого є наявність інтелектуального помічника для взаємодії з клієнтами. Система має забезпечувати стандартний комерційний функціонал – перегляд каталогу товарів, авторизацію користувачів та оформлення замовлень, але водночас пропонувати якісно новий рівень клієнтської підтримки через інтерактивний чат. Основна технічна проблема полягає в тому, щоб поєднати звичні статичні модулі сайту з динамічним інструментом генерації відповідей у реальному часі.

Це вимагає детального аналізу того, як саме новітні технології штучного інтелекту вбудовуються в класичні вебрішення, оскільки інтеграція великих мовних моделей (Large Language Models, LLM) у сучасні вебдодатки суттєво змінює підхід до проєктування архітектури систем. На відміну від традиційних вебзастосунків, де основна логіка базується на детермінованих алгоритмах, системи з LLM передбачають роботу з ймовірнісною генерацією тексту, що виконується через зовнішні API, зокрема такі як Azure OpenAI Service. Це впливає на структуру системи, розподіл відповідальностей між компонентами та вимоги до масштабованості й безпеки [24].

Одним із базових підходів залишається монолітна архітектура, у межах якої фронтенд, бекенд і модуль взаємодії з LLM реалізуються як єдине програмне середовище. Такий підхід може бути виправданим для невеликих або навчальних проєктів, оскільки забезпечує простоту розгортання та мінімальну кількість залежностей. У контексті інтеграції LLM моноліт дозволяє безпосередньо викликати API Azure OpenAI з серверної частини додатка. Однак зі зростанням навантаження та кількості користувацьких запитів виникають

проблеми масштабування, а також ускладнюється незалежне оновлення компонентів системи [27].

Більш сучасним і гнучким рішенням є мікросервісна архітектура, яка передбачає поділ системи на незалежні сервіси. У випадку LLM-орієнтованих вебдодатків зазвичай виділяються окремі модулі: клієнтський інтерфейс, основний API-сервер, сервіс обробки запитів до LLM, а також додаткові сервіси для кешування та логування. Такий підхід дозволяє масштабувати LLM-компонент окремо від основної системи, що є особливо важливим через обмеження швидкості та вартості API-запитів до Azure OpenAI. Крім того, мікросервіси забезпечують гнучкість у виборі технологій для кожного окремого компонента [27].

Окрему роль у сучасних вебсистемах відіграє API-first архітектурний підхід, який передбачає, що вся взаємодія між клієнтом і сервером відбувається через чітко визначені API-ендпоінти. У системах з LLM це дозволяє уніфікувати процес формування запитів до моделі: клієнтська частина надсилає текстовий запит, сервер формує prompt, додає контекст (історію діалогу, системні інструкції), після чого виконується виклик Azure OpenAZURE OPENAI [28].

Важливим архітектурним аспектом є також вибір між синхронною та асинхронною моделлю обробки запитів. У синхронному підході користувач очікує відповідь LLM у режимі реального часу, що підходить для чат-інтерфейсів. Однак при високому навантаженні це може призводити до затримок. Асинхронна модель, навпаки, передбачає використання черг завдань (наприклад, Redis Queue або RabbitMQ), де запит спочатку потрапляє в чергу, обробляється окремим воркером, а результат повертається користувачу після генерації [29].

Ще одним поширеним підходом є використання Serverless архітектури, особливо у поєднанні з хмарними сервісами, такими як Azure Functions. У такій моделі логіка взаємодії з LLM винесена у функції, які виконуються лише при

надходженні запиту. Це дозволяє зменшити витрати на інфраструктуру та автоматично масштабувати систему залежно від кількості запитів [30].

Також слід враховувати підхід до побудови інтерфейсу – SPA (Single Page Application) або гібридні рендерингові моделі. SPA дозволяє реалізувати інтерактивний чат із LLM без перезавантаження сторінки, забезпечуючи безперервний користувацький досвід. У такій архітектурі запити до Azure OpenAI надсилаються через REST API або WebSocket-з'єднання, а відповіді динамічно відображаються в інтерфейсі. У деяких випадках використовується також SSR (Server-Side Rendering) для покращення початкового завантаження сторінки та SEO-оптимізації [31].

Узагальнено архітектурні підходи до побудови вебдодатків з інтеграцією LLM можна переглянути у таблиці 2.1.

Таблиця 2.1 - Архітектурні підходи до побудови вебдодатків з інтеграцією LLM

Архітектурний підхід	**Основні переваги**	**Недоліки**	**Рекомендоване застосування**
Монолітна	Простота реалізації	Погана масштабованість	Навчальні проєкти
Мікросервісна	Гнучкість, масштабування	Складність інфраструктури	Комерційні системи
API-first	Уніфікація логіки	Потребує чіткого дизайну API	Багатоплатформені системи
Serverless	Автоматичне масштабування	Затримки cold start	Нерегулярні навантаження
SPA	Інтерактивність UI	SEO-обмеження	Чат-системи, LLM інтерфейси

Вибір архітектури для вебдодатку з інтеграцією LLM напряму залежить від вимог до продуктивності, масштабованості та вартості обробки запитів до Azure OpenAI. Найбільш ефективними вважаються комбінації мікросервісного та API-first підходів із використанням асинхронної обробки запитів, що

дозволяє забезпечити стабільну роботу системи при інтенсивній взаємодії користувачів з мовною моделлю.

2.2 Обгрунтування вибору архітектури системи генерації відповідей

Вибір архітектури вебдодатку є важливим етапом проєктування, оскільки визначає спосіб організації взаємодії між клієнтською частиною, серверною логікою та зовнішніми сервісами. У розроблюваному вебдодатку інтернет-магазину з інтегрованим AI-консультантом архітектура формується з урахуванням необхідності обробки користувацьких запитів, генерації відповідей та забезпечення зручного інтерфейсу взаємодії.

У межах реалізації застосовано клієнт-серверний підхід, де клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем, а серверна – за обробку запитів і інтеграцію з AI-сервісом. Клієнтська логіка реалізована за допомогою JavaScript-файлу, який забезпечує відкриття чат-віджета, обробку введених повідомлень та їх передачу на сервер без перезавантаження сторінки. Такий підхід дозволяє реалізувати асинхронну взаємодію та покращує користувацький досвід [32].

Серверна частина системи представлена окремим PHP-файлом, який виконує роль обробника запитів. Він приймає повідомлення користувача, виконує базову валідацію вхідних даних, формує запит до зовнішнього AI-сервісу та повертає відповідь у форматі JSON. Така організація дозволяє централізувати логіку роботи з AI та забезпечити контроль над процесом генерації відповідей.

Інтеграція з мовною моделлю реалізована через використання зовнішнього API, до якого сервер звертається за допомогою HTTP-запитів. Для цього використовується механізм cURL, що дозволяє передавати дані у форматі JSON, додавати заголовки авторизації та отримувати результати роботи моделі. Вся взаємодія з AI відбувається на сервері, що забезпечує захист API-ключа та підвищує безпеку системи [28].

З огляду на таку реалізацію, у структурі додатку простежується логічне розділення функцій: клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, серверна – за обробку запитів і інтеграцію з AI, а зовнішній сервіс – за генерацію відповідей. Подібний підхід фактично реалізує спрощену модель розподілу відповідальності, близьку до принципів MVC, де окремі компоненти виконують чітко визначені функції [32].

Обрана архітектура також добре узгоджується з іншими частинами вебдодатку, зокрема сторінками каталогу товарів, кошика та профілю користувача, які реалізовані у вигляді окремих PHP-файлів із розподілом логіки та представлення. Це забезпечує модульність системи та спрощує її подальший розвиток.

Використання клієнт-серверної моделі та окремого обробника для AI-запитів дозволяє зробити структуру проєкту зрозумілою та зручною. Такий підхід дає змогу повністю контролювати кожен етап обробки даних на сервері. Окрім цього, обрана архітектура спрощує подальший розвиток додатка, оскільки нові функції можна додавати легко, не вносячи суттєвих змін у його основну частину.

2.3 Загальна архітектура системи з інтегрованим LLM

Технічна реалізація вебдодатку інтернет-магазину «UrbanWear» базується на чіткому розподілі функцій між окремими програмними файлами, кожен з яких відповідає за свою ділянку роботи: від відображення візуальних елементів до обробки логіки замовлень та інтеграції з мовними моделями. Головна сторінка сайту (index.php) побудована на поєднанні HTML5, стилів Tailwind CSS та іконок Font Awesome. Вона виконує роль презентаційного лендингу, де за допомогою PHP-конструкції include підключаються спільні для всього сайту модулі шапки та підвалу. Сторінка містить промо-блок із градієнтним фоном, адаптивну сітку переваг магазину та каркас плаваючого віджета AI-консультанта, який складається з фіксованої кнопки запуску та

прихованого вікна чату з окремими зонами для заголовка, історії повідомлень та форми введення тексту [21].

Клієнтська логіка цього інтелектуального помічника повністю винесена в окремий JavaScript-файл (ai.js), який працює як ES-модуль. Скрипт знаходить потрібні HTML-елементи в дереві DOM, керує відкриттям та закриттям вікна чату через перемикання CSS-класів та динамічно створює бульбашки повідомлень, розділяючи стилі для користувача та штучного інтелекту. При відправці форми скріпт блокує перезавантаження сторінки, зчитує введений текст і через асинхронну функцію (fetch) надсилає POST-запит на ендпоінт платформи Groq. У запиті передається токен авторизації, назва обраної мовної моделі та системна інструкція, яка змушує ШІ поводитися як продавець-консультант магазину одягу, а отримана JSON-відповідь автоматично виводиться на екран.

Паралельно з цим у системі передбачено серверний файл обробки AI-запитів (ai_chat.php), який виступає альтернативним захищеним шлюзом і зв'язується з платформою Hugging Face через механізм cURL. Цей PHP-скрипт приймає повідомлення від клієнта, очищає його від пробілів за допомогою функції trim() і перевіряє на наявність тексту, повертаючи помилку у випадку порожнього рядка. Завдяки виконанню запиту на рівні сервера секретний ключ авторизації залишається повністю прихованим від звичайних користувачів. Сервер передає запит до обраної моделі, налаштовує параметри довжини відповіді та рівня випадковості тексту, декодує отриманий від API результат і транслює його назад у браузер у стандартному форматі JSON.

Інші комерційні сторінки сайту працюють у тісній зв'язці з локальними файловими сховищами та механізмом сесій. Сторінка каталогу товарів (catalog.php) використовує статичний асоціативний масив PHP, що виконує роль бази даних, та реалізує алгоритм пагінації, який вираховує потрібну кількість сторінок і розбиває вивід асортименту на адаптивну сітку по 9 позицій на екран. При натисканні на конкретний товар користувач потрапляє на сторінку детального перегляду (product.php), де через GET-запит зчитується

унікальний ідентифікатор речі. Скрипт розкладає масив даних на змінні для виводу великого зображення, опису та ціни, активує модальне вікно розмірної сітки та підтягує чотири схожі моделі для покращення навігації.

Усі операції з відібраними речами координує файл кошика (cart.php), логіка якого повністю побудована на серверних PHP-сесіях без постійного запису в базу даних. Команда session_start() відкриває доступ до глобального масиву, де при додаванні товару створюється новий запис або збільшується кількість уже наявних одиниць, а після видалення позицій через GET-параметр спрацьовує мовна конструкція unset(). У шаблоні сторінки запускається цикл, який множить ціну кожної речі на її кількість і накопичує фінальну суму замовлення, яка згодом передається на сторінку оформлення покупки.

Процес завершення покупки реалізовано у файлі checkout.php, який спочатку перевіряє авторизацію користувача та наявність товарів у кошику, захищаючи систему від фіктивних операцій. Контактні дані покупця автоматично підтягуються з його профілю та безпечно екрануються через функцію htmlspecialchars() для захисту від шкідливого коду. При відправці форми сервер формує масив замовлення, кодує його в JSON-рядок і записує в кінець файлу orders.json, після чого повністю очищає кошик у сесії. Керування самими акаунтами відбувається у файлі особистого кабінету (profile.php), де через цикл здійснюється пошук потрібного користувача в JSON-файлі за його електронною поштою, перезаписуються оновлені текстові поля та миттєво синхронізуються дані з активною сесією без необхідності повторного входу на сайт.

2.4 Реалізація механізмів генерації відповідей на запити користувачів

Проєктування архітектури системи генерації відповідей є важливим етапом розробки вебдодатку, оскільки саме на цьому рівні визначається структура компонентів, їх взаємодія та логіка обробки запитів користувача. Для розроблюваного інтернет-магазину з AI-консультантом обрана архітектура, яка

поєднує класичний клієнт-серверний підхід із використанням зовнішніх сервісів штучного інтелекту.

Загальна структура системи передбачає поділ на кілька основних рівнів: клієнтський (frontend), серверний (backend), рівень інтеграції з AI-сервісом та рівень зберігання даних. Такий підхід дозволяє розмежувати відповідальність між компонентами, що спрощує розробку, тестування та подальшу підтримку системи.

Клієнтська частина вебдодатку реалізована за допомогою HTML, Tailwind CSS та JavaScript і відповідає за формування інтерфейсу користувача. Саме на цьому рівні відображаються сторінки каталогу товарів, кошика, профілю користувача та AI-консультанта. Особливістю реалізації є використання асинхронної взаємодії з сервером, що забезпечується через fetch API. Це дозволяє надсилати запити та отримувати відповіді без перезавантаження сторінки, що значно покращує користувацький досвід, особливо при роботі з чат-інтерфейсом.

Серверна частина системи реалізована мовою програмування PHP та виконує функції обробки запитів, реалізації бізнес-логіки та координації взаємодії між різними компонентами. Кожен функціональний модуль вебдодатку представлений окремими файлами, що відповідають за конкретні задачі, зокрема відображення каталогу товарів, обробку кошика, оформлення замовлення та управління профілем користувача. Окрему роль відіграє модуль обробки AI-запитів, який приймає повідомлення користувача, виконує його валідацію, формує запит до зовнішнього сервісу та повертає результат у форматі JSON [33].

Інтеграція з AI-сервісом є ключовою складовою системи генерації відповідей. Взаємодія з моделями штучного інтелекту здійснюється через HTTP-запити до відповідного API. Сервер виступає проміжною ланкою, яка забезпечує передачу даних між клієнтом і AI-сервісом, а також виконує обробку отриманої відповіді. Такий підхід дозволяє приховати логіку роботи з API від клієнтської частини та забезпечити більш безпечну обробку запитів.

Рівень зберігання даних у проєкті реалізовано у спрощеному вигляді. Замість використання повноцінної системи управління базами даних застосовуються JSON-файли, які містять інформацію про користувачів і замовлення. Для тимчасового зберігання даних, таких як вміст кошика, використовуються PHP-сесії. Це рішення дозволяє зменшити складність реалізації системи та є достатнім для демонстрації основного функціоналу вебдодатку. Загальна взаємодія компонентів системи наведена на рисунку 2.1.

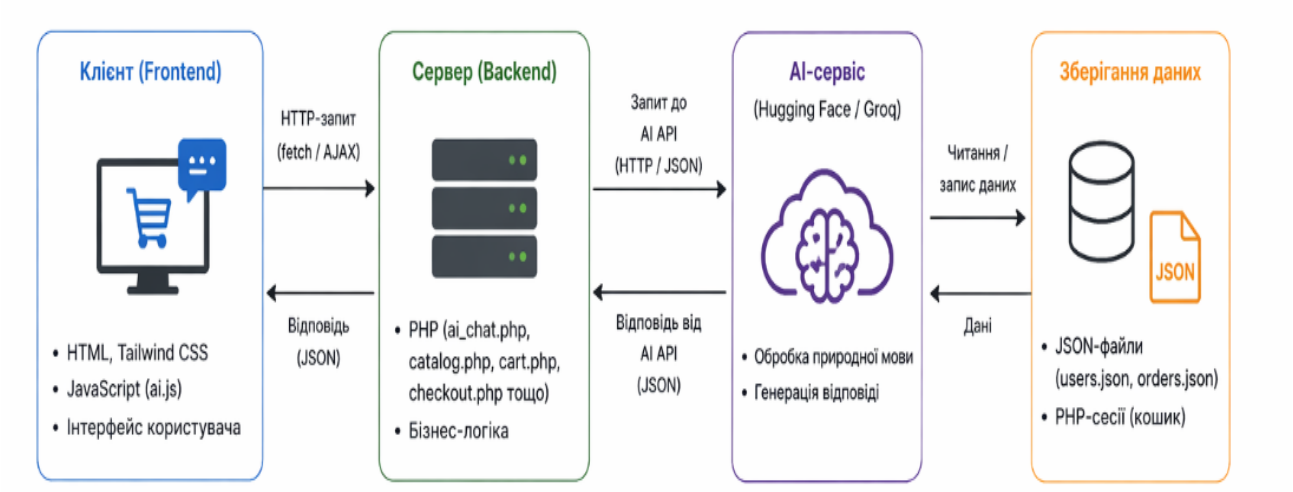

Рисунок 2.1 – Взаємодія компонентів системи

Згідно зі схемою, процес обробки запиту починається з дій користувача, який вводить повідомлення у чат-інтерфейсі або взаємодіє з іншими елементами системи. Запит передається на сервер у вигляді HTTP-запиту, де виконується його аналіз і обробка. У випадку використання AI-консультанта сервер формує запит до зовнішнього AI-сервісу, передає необхідні дані та очікує на відповідь. Після отримання результату сервер обробляє його та повертає клієнту у форматі JSON, де він відображається у вигляді повідомлення в чаті.

Окрім цього, у процесі обробки запиту система може виконувати додаткові операції, пов’язані зі збереженням або отриманням даних. Наприклад, при оформленні замовлення відбувається запис інформації у JSON-

файл, а при роботі з кошиком – зчитування та оновлення даних у сесії. Архітектура забезпечує не лише генерацію відповідей, але й повноцінну роботу інтернет-магазину.

Обрана архітектура відзначається простотою реалізації та достатньою гнучкістю. Вона дозволяє легко інтегрувати нові сервіси, змінювати логіку обробки запитів або розширювати функціонал системи без суттєвих змін у вже реалізованих компонентах. Крім того, чітке розділення на рівні сприяє кращій організації коду та підвищує зручність його підтримки.

Запропонована архітектура відповідає вимогам сучасних вебдодатків, забезпечує ефективну взаємодію з AI-сервісами та створює основу для подальшого розвитку системи генерації відповідей.

2.4.1 Моделювання структури даних та організації зберігання інформації

Ефективність розробленої системи безпосередньо зумовлена обраною моделлю структурування інформаційних потоків, загальну схему яких представлено на рисунку 2.2. У проєкті реалізовано гібридну схему зберігання, що поєднує використання структурованих файлів формату JSON для постійної фіксації даних та механізму серверних сесій для управління динамічними станами користувача. Такий підхід дозволяє забезпечити високу швидкість відгуку інтерфейсу та спростити архітектуру системи, уникаючи надлишкового навантаження, характерного для реляційних баз даних у межах локальних вебрішень.

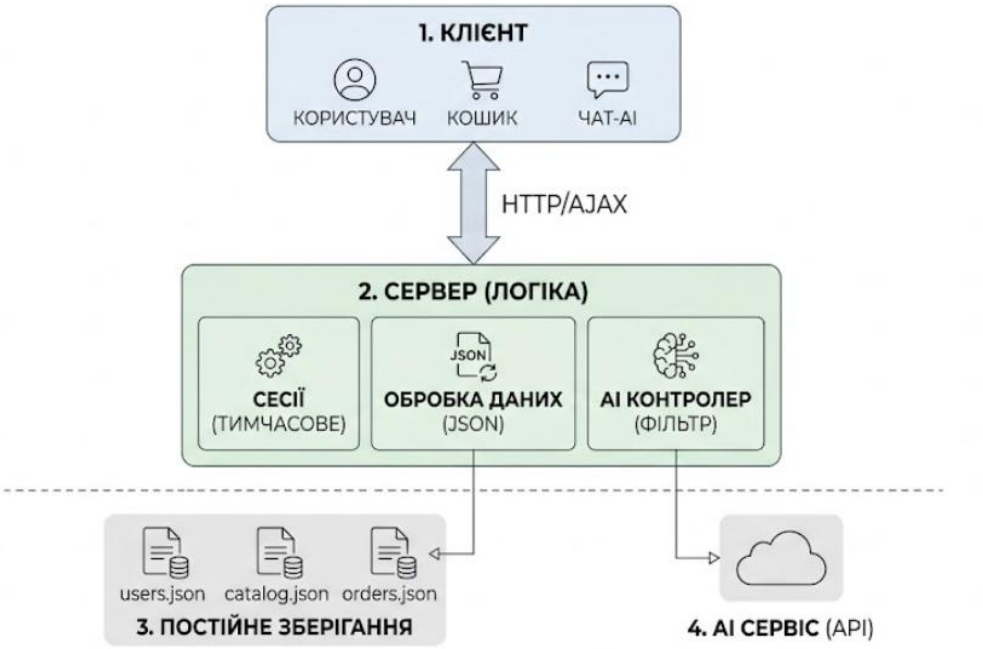

Рисунок 2.2 – Схема організації даних та інформаційних потоків у системі

Постійне зберігання інформації базується на ієрархічній файловій моделі. Ключові сутності системи, такі як персональні профілі користувачів та реєстр комерційних операцій, форматовані у вигляді об’єктів JSON. Це забезпечує уніфікацію даних та дозволяє серверній частині додатка оперативно трансформувати текстову інформацію у структуровані масиви для подальшої обробки. Завдяки автоматизації процесів кодування та декодування, система підтримує цілісність даних при виконанні операцій читання та оновлення, що критично важливо для верифікації історії замовлень.

Для управління тимчасовими станами та підтримання контексту взаємодії застосовано механізм серверних сесій. Це дозволяє створити захищене сховище для динамічних даних, зокрема для віртуального кошика та параметрів авторизації. Використання сесійного масиву гарантує збереження вибору користувача під час навігації між різними модулями системи, а також виступає засобом розмежування прав доступу.

Таке рішення запобігає втраті контексту при переході між сторінками та забезпечує безпеку персональних розділів додатка, оскільки ідентифікація

відбувається на рівні сервера. Практична реалізація інтерфейсу точки входу користувача до системи, яка активує сесійні механізми та зчитує дані з файлових сховищ, демонструється за допомогою модального вікна авторизації (рис. 2.3).

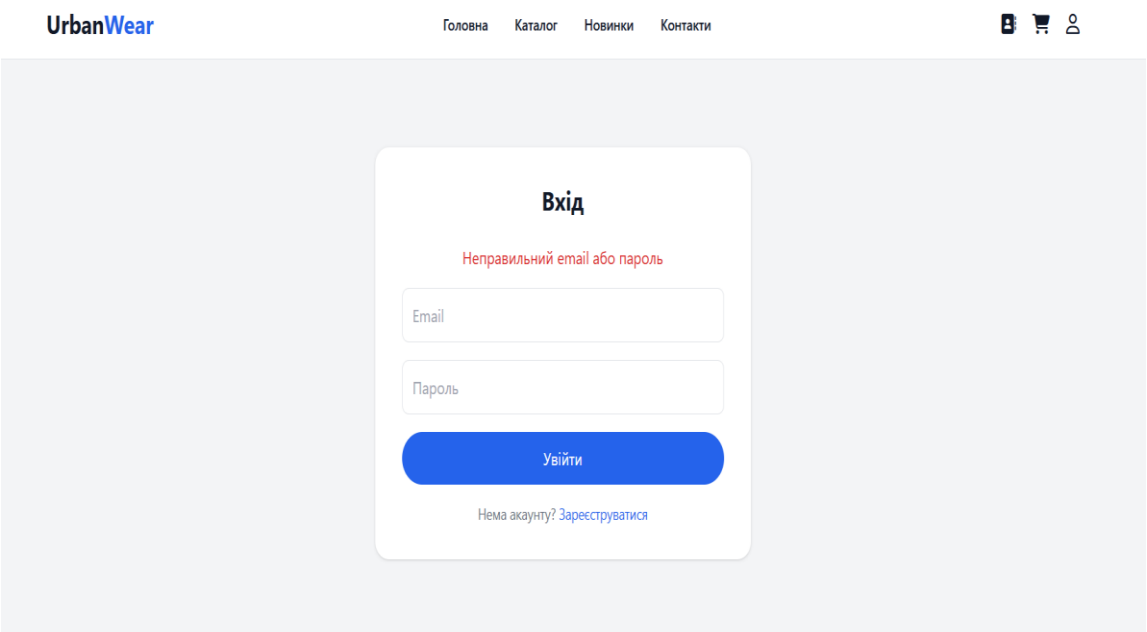

Рисунок 2.3 – Інтерфейс модального вікна авторизації та реєстрації користувачів у системі

Окрему увагу приділено моделюванню обміну даними в межах інтелектуального модуля. Взаємодія між клієнтською частиною та серверним обробником реалізована за допомогою асинхронних запитів, що забезпечує безперервність користувацького досвіду. Перед відправкою даних до зовнішніх інтелектуальних сервісів система здійснює попередню фільтрацію вхідних повідомлень та формування запиту за чітко визначеними технічними параметрами. Це дозволяє контролювати рівень креативності генерації та обсяг отриманих відповідей, адаптуючи роботу нейромережі до специфіки конкретних запитів.

Спроєктована структура даних, де об'єкти каталогу та користувацькі записи мають сталу сукупність атрибутів, забезпечує системі необхідну

гнучкість. Модульний підхід до організації інформації дозволяє легко масштабувати функціонал та оновлювати контент без втручання в основну програмну логіку. Обрана модель зберігання та обробки даних формує надійний технологічний фундамент для стабільного функціонування сучасного вебдодатку з елементами штучного інтелекту.

2.4.2 Проєктування алгоритмів обробки запитів та генерації відповідей

Ефективність функціонування інтелектуального модуля у вебдодатку базується на чітко визначених алгоритмах обробки вхідних даних та взаємодії з мовними моделями. Процес обробки запиту користувача та формування відповіді реалізовано як послідовність взаємопов’язаних етапів, що охоплюють клієнтську перевірку даних, серверну обробку та інтеграцію із зовнішніми AI-сервісами.

Загальну логіку виконання цього процесу наведено на рисунку 2.4.

Як показано на схемі, початковий етап алгоритму реалізується на клієнтській стороні, де відбувається обробка дій користувача. Після введення повідомлення відповідний JavaScript-скрипт перехоплює подію його відправки та виконує попередню обробку тексту. Зокрема, здійснюється очищення рядка від зайвих пробілів і перевірка на порожність, що дозволяє уникнути надсилання некоректних або порожніх запитів.

У разі успішного проходження первинної валідації алгоритм переходить до наступного етапу – передачі даних на сервер. Для цього використовується асинхронний механізм fetch API, який забезпечує обмін даними у фоновому режимі без перезавантаження сторінки. Такий підхід дозволяє зберегти безперервність взаємодії користувача з інтерфейсом і підвищує загальну зручність роботи із системою.

Рисунок 2.4 – Алгоритм обробки запиту користувача та генерації відповіді

Важливим аспектом проєктування інтерфейсу в межах асинхронної моделі є динамічне інформування користувача про поточний стан обробки запиту. Наочна демонстрація роботи діалогового вікна ШІ-консультанта в моменти ініціалізації сесії, відправки репліки користувачем та відображення проміжного статусу генерації («AI думає...») наведена на рисунку 2.5.

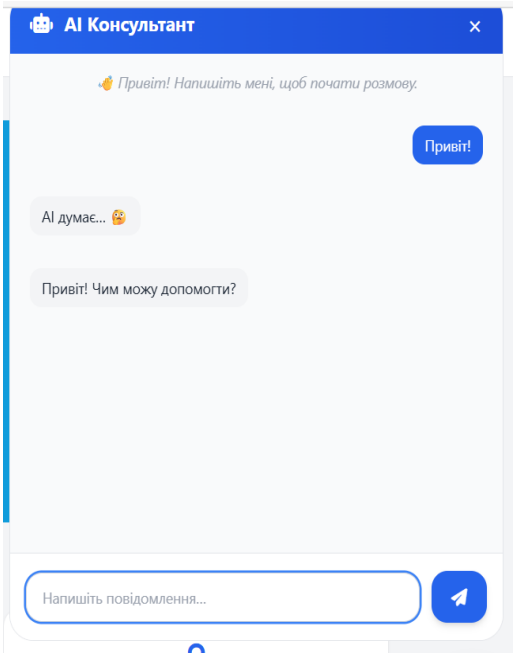

Рисунок 2.5 – Інтерфейс чат-віджета в процесі асинхронного обміну даними та очікування відповіді від API

Після надходження запиту серверна частина, реалізована на PHP, виконує його обробку. На цьому етапі здійснюється повторна перевірка коректності вхідних даних, що є важливим елементом забезпечення безпеки. Далі формується структурований HTTP-запит до зовнішнього AI-сервісу, який містить текст повідомлення користувача та параметри генерації відповіді.

Наступним кроком є безпосередній виклик API мовної моделі. Сервер передає сформований запит до зовнішнього сервісу та очікує на результат обробки. Отримана відповідь надходить у форматі JSON, що зумовлює необхідність її подальшого аналізу та обробки [28].

Після отримання відповіді виконується її декодування, у процесі якого із загальної структури даних виділяється текстовий результат генерації. У випадках виникнення помилок, пов’язаних із мережею або відсутністю відповіді від сервісу, алгоритм передбачає формування альтернативного повідомлення, що забезпечує стабільність роботи системи та коректну взаємодію з користувачем.

Оброблений результат передається назад на клієнтську сторону, де він динамічно інтегрується в інтерфейс чату. Відповідний скрипт додає нове

повідомлення до історії діалогу та автоматично прокручує вікно до останнього елемента, що забезпечує зручність подальшої взаємодії.

2.5 Висновки до другого розділу

У результаті проведеного проєктування було сформовано цілісну архітектурну модель вебдодатку, яка базується на принципах модульності, безпеки та високої продуктивності. Обраний клієнт-серверний підхід дозволив чітко розмежувати функції інтерфейсу, серверної логіки та зовнішніх інтелектуальних сервісів. Завдяки використанню асинхронної взаємодії через JavaScript та технологію fetch, вдалося реалізувати динамічний чат-інтерфейс, який функціонує без перезавантаження сторінок, що суттєво покращує користувацький досвід та забезпечує миттєву реакцію системи на запити.

Технологічний стек, що поєднує мову PHP для серверних операцій, Tailwind CSS для адаптивної стилізації та формат JSON для збереження інформації, продемонстрував оптимальний баланс між швидкістю розробки та ефективністю роботи програми. Відмова від складних реляційних баз даних на користь структурованих JSON-файлів та механізму PHP-сесій дозволила створити легковагову, але надійну систему зберігання даних, яка забезпечує швидкий доступ до профілів користувачів, історії замовлень та вмісту кошика.

Особлива увагу у проєкті приділено алгоритмізації процесів генерації відповідей. Розроблена схема обробки запитів включає багаторівневу валідацію вхідних даних та гнучке налаштування параметрів мовних моделей, що гарантує релевантність та коректність отриманих результатів. Важливим досягненням стало забезпечення безпеки системи: винесення взаємодії з AI-сервісами на серверний рівень дозволило надійно захистити ключі авторизації та запобігти зловмисним маніпуляціям із запитами на стороні клієнта.

Загалом, запропоновані підходи до побудови архітектури та моделювання даних створили стабільний фундамент для функціонування сучасного вебресурсу. Поєднання асинхронних методів обміну інформацією, серверної

фільтрації та інтеграції з передовими мовними моделями дозволило реалізувати інтелектуальну систему, яка є одночасно безпечною, продуктивною та готовою до подальшого масштабування функціональних можливостей.

3 РЕАЛІЗАЦІЯ ВЕБДОДАТКА ТА ІНТЕГРАЦІЯ МОДУЛЯ АВТОМАТИЗАЦІЇ ВІДПОВІДЕЙ

3.1 Експериментальні дослідження та методика тестування якості відповідей LLM

У межах даного дослідження було проведено експериментальне тестування якості відповідей великої мовної моделі (LLM), інтегрованої у вебдодаток інтернет-магазину UrbanWear. Архітектура системи, яка забезпечує роботу основних модулів вебдодатку (клієнтська частина, серверна логіка, API-запити та інтеграція AI-консультанта), що відображає загальну структуру взаємодії компонентів системи та дозволяє зрозуміти місце LLM у загальній архітектурі.

Методика тестування базується на симуляції реальних користувацьких запитів у межах функціоналу вебсистеми. Оскільки AI-консультант інтегрований у сторінки магазину (головну сторінку, каталог та сторінку товару), тестування охоплює як загальні запити, що стосуються роботи магазину та інтерфейсу, так і предметно-орієнтовані запити, пов’язані з товарами, їх характеристиками, ціною та процесом оформлення замовлення. Це дозволяє оцінити поведінку моделі в умовах, максимально наближених до реального використання.

Тестування проводилося у кілька етапів, що включали перевірку базової здатності моделі генерувати текстові відповіді, оцінювання її роботи в контексті електронної комерції, а також перевірку стабільності взаємодії з API та коректності обробки помилок. У якості мовної моделі використовувалися рішення, підключені через зовнішні API, зокрема Hugging Face Router API та Groq OpenAI-сумісний інтерфейс. Такий підхід дозволив додатково оцінити не лише якість відповідей, але й технічні характеристики взаємодії, включаючи затримки та стабільність мережевих запитів [25].

Тестові запити були сформовані таким чином, щоб охопити різні типи користувацьких сценаріїв. Вони включали інформаційні запити загального характеру, товарні запити щодо характеристик і вартості продукції, навігаційні запити, пов’язані з використанням сайту, сценарні запити, які імітують повний цикл покупки, а також стресові запити, що містять неповні, некоректні або неоднозначні формулювання. Така класифікація дозволяє всебічно оцінити здатність моделі адаптуватися до різних умов взаємодії.

Оцінювання якості відповідей здійснювалося за кількома критеріями, серед яких релевантність відповіді запиту, точність поданої інформації, контекстність у межах e-commerce середовища, читабельність і зрозумілість формулювань, стійкість до помилкових або некоректних введень, а також швидкість генерації відповіді. Для кожного з критеріїв використовувалася умовна шкала оцінювання, що дозволило узагальнити результати та визначити загальний рівень ефективності системи.

Результати тестування показали, що модель демонструє високу якість відповідей у стандартних інформаційних сценаріях, зокрема при консультуванні користувачів щодо товарів, навігації по сайту та процесу оформлення замовлення. Відповіді здебільшого є логічними, структурованими та зрозумілими, що позитивно впливає на користувацький досвід. Водночас у випадках стресових або нечітких запитів спостерігається тенденція до генерації узагальнених відповідей, що може знижувати точність інтерпретації наміру користувача.

Додатково було проаналізовано інтеграційний рівень роботи системи. Встановлено, що взаємодія між клієнтською частиною та серверною логікою реалізована коректно: API-запити стабільно виконуються через cURL та fetch, обробка JSON-відповідей відбувається без помилок, а інтерфейс чату не впливає негативно на продуктивність сторінки. Також система адекватно реагує на помилки мережі або зовнішнього API, забезпечуючи користувачу зрозуміле повідомлення замість критичного збою інтерфейсу [25].

У підсумку можна зробити висновок, що інтегрована LLM-модель забезпечує достатній рівень якості відповідей для використання в межах e-commerce вебдодатку. Найбільш ефективною вона є у стандартних інформаційних та консультаційних сценаріях. Разом із тим, для підвищення точності у складніших випадках доцільним є вдосконалення системи системних інструкцій та розширення контекстної обробки запитів, що дозволить підвищити адаптивність моделі та покращити загальну якість взаємодії з користувачем.

3.2 Порівняння роботи системи з традиційними методами пошуку

У процесі реалізації вебдодатку UrbanWear було виконано порівняння запропонованого AI-орієнтованого підходу до пошуку товарів із традиційними методами, які зазвичай використовуються в класичних інтернет-магазинах. Метою такого порівняння є визначення різниці в принципах обробки запитів користувача, зручності взаємодії та рівні інтелектуальності системи.

Традиційний підхід у вебдодатках зазвичай базується на фільтрації товарів за ключовими словами або параметрами, такими як назва, категорія або ціна. У межах системи UrbanWear такий механізм може бути реалізований через обробку масиву товарів у PHP із перевіркою відповідності рядків. Основним принципом роботи є пряме порівняння введеного користувачем запиту з полями товарів, що зберігаються у системі. Такий підхід є досить швидким і простим у реалізації, однак він не враховує зміст запиту та наміри користувача.

У традиційній моделі пошуку результат залежить від точності введеного запиту. Якщо користувач вводить неточний або узагальнений запит, система не завжди здатна коректно підібрати відповідні товари, оскільки відсутній семантичний аналіз тексту. У результаті користувач може отримати неповну або нерелевантну вибірку товарів, що знижує ефективність взаємодії з інтернет-магазином.

На відміну від цього, у вебдодатку UrbanWear реалізовано AI-орієнтований підхід, який базується на інтеграції мовної моделі через API сервісу OpenAI, розгорнутого в інфраструктурі Microsoft Azure. У цьому випадку запит користувача не обмежується ключовими словами, а передається як повноцінне текстове повідомлення з контекстом.

AI-модель аналізує зміст запиту, визначає наміри користувача та формує відповідь у вигляді рекомендацій або пояснень. Завдяки цьому користувач може отримувати результати навіть у випадках нечітко сформульованих запитів, наприклад, коли він описує бажаний товар загальними словами, а не конкретною назвою.

Важливою особливістю AI-підходу є можливість ведення діалогу, коли користувач може уточнювати свій запит, а система – адаптувати відповідь відповідно до нової інформації. Це суттєво відрізняється від традиційного пошуку, де кожен запит обробляється окремо і не враховує попередню взаємодію.

Традиційний підхід у системі забезпечує високу швидкість та простоту реалізації, тоді як AI-орієнтований механізм надає більш гнучкий та інтелектуальний спосіб взаємодії з користувачем. Найбільш ефективним у межах подібних систем є поєднання обох підходів, коли базова фільтрація використовується для швидкого відбору товарів, а AI-модель – для уточнення запитів та формування рекомендацій.

3.3 Програмна реалізація та опис інструкції користувача

3.3.1 Реалізація алгоритмів пошуку та зббереження товарів

У вебдодатку UrbanWear реалізовано базові алгоритми роботи з товарами, які включають вибірку, відображення в каталозі та збереження обраних позицій у кошику користувача. Основна логіка побудована на використанні PHP та роботи з масивами даних, що дозволяє реалізувати функціональність інтернет-

магазину без використання повноцінної реляційної бази даних. Такий підхід забезпечує простоту структури проєкту та швидку обробку запитів користувача.

Інтерфейс каталогу товарів реалізовано таким чином, щоб користувач міг зручно переглядати доступні позиції, фільтрувати їх та переходити до детального перегляду. Загальний вигляд сторінки каталогу представлено на рисунку 3.1.

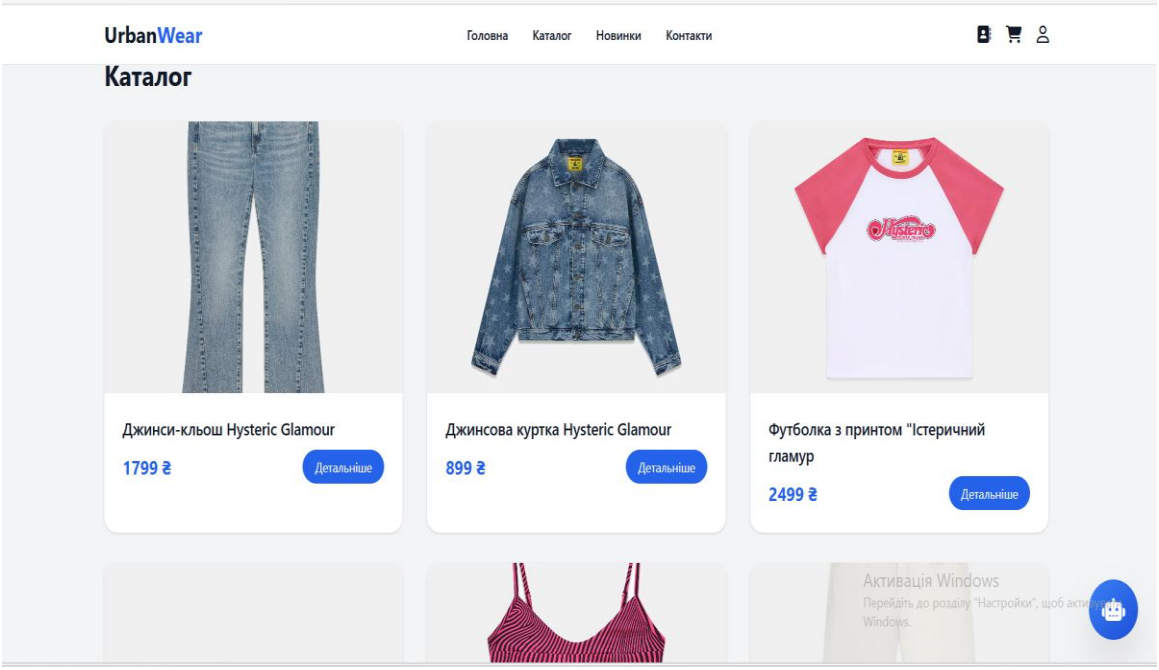

Рисунок 3.1 – Інтерфейс каталогу товарів вебдодатку UrbanWear

Також у межах реалізації передбачено сторінку детального перегляду товару, де користувач може ознайомитися з повною інформацією про обрану позицію, її характеристиками та ціною. Даний інтерфейс представлено на рисунку 3.2.

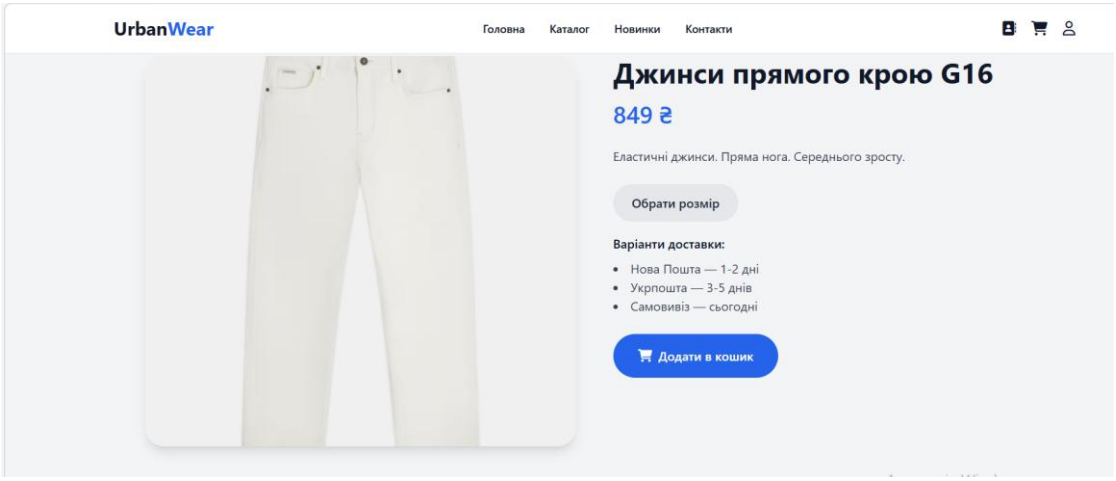

Рисунок 3.2 – Сторінка детального перегляду товару

Окремим елементом інтерфейсу є кошик користувача, який дозволяє переглядати додані товари, змінювати їх кількість або видаляти позиції. Його вигляд показано на рисунку 3.3.

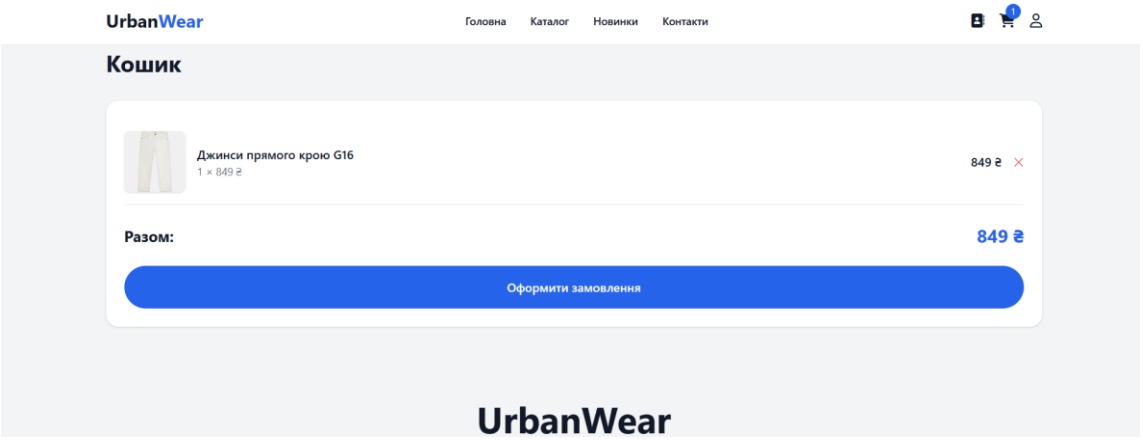

Рисунок 3.3 – Інтерфейс кошика вебдодатку UrbanWear

Алгоритм пошуку та відображення товарів у каталозі реалізовано через формування масиву товарів та подальшу його обробку із застосуванням механізму пагінації. Це дозволяє обмежувати кількість товарів, що відображаються на одній сторінці, та підвищує зручність взаємодії користувача з інтерфейсом. Поточна сторінка визначається через GET-параметр, після чого обчислюється початковий індекс вибірки та формується відповідний набір товарів.

Лістинг 3.1 – Алгоритм пагінації та вибірки товарів

```
$perPage = 9;
$total = count($products);
$totalPages = ceil($total / $perPage);
$page = max(1, min($totalPages, (int)($_GET['page'] ?? 1)));
$start = ($page - 1) * $perPage;
$items = array_slice($products, $start, $perPage);
```

Таким чином забезпечується рівномірний розподіл товарів по сторінках каталогу та зменшується навантаження на інтерфейс користувача. Кожна товарна позиція відображається у вигляді окремої картки, що містить основну інформацію про товар, включаючи назву, ціну та зображення.

Окремим елементом реалізації є механізм збереження обраних товарів користувача у кошику. Для цього використовується PHP-сесія, яка дозволяє зберігати стан користувача між різними сторінками вебдодатку. При додаванні товару до кошика система перевіряє, чи існує вже така позиція, і у разі повторного додавання збільшує її кількість [33].

Лістинг 3.2 – Додавання товару у кошик

```
$_SESSION['cart'][$id] = ($_SESSION['cart'][$id] ?? 0) + 1;
```

Такий механізм дозволяє ефективно реалізувати базову логіку кошика без необхідності використання зовнішньої бази даних. Додатково реалізовано можливість видалення товарів та оновлення стану кошика через GET-запити, що забезпечує повний цикл роботи з обраними товарами.

3.3.2 Система підказок та історії пошуку

У вебдодатку UrbanWear реалізовано систему інтерактивної взаємодії користувача з AI-консультантом, яка виконує функції підказок та обробки запитів у режимі діалогу. Основою цієї системи є інтеграція чат-модуля, який дозволяє користувачу отримувати рекомендації щодо товарів, уточнення інформації та допомогу у навігації по сайту.

Процес обробки повідомлення користувача в AI-консультанті передбачає послідовну передачу запиту від клієнта до серверної частини та подальшу взаємодію з AI-модулем, що показано на рисунку 3.4.

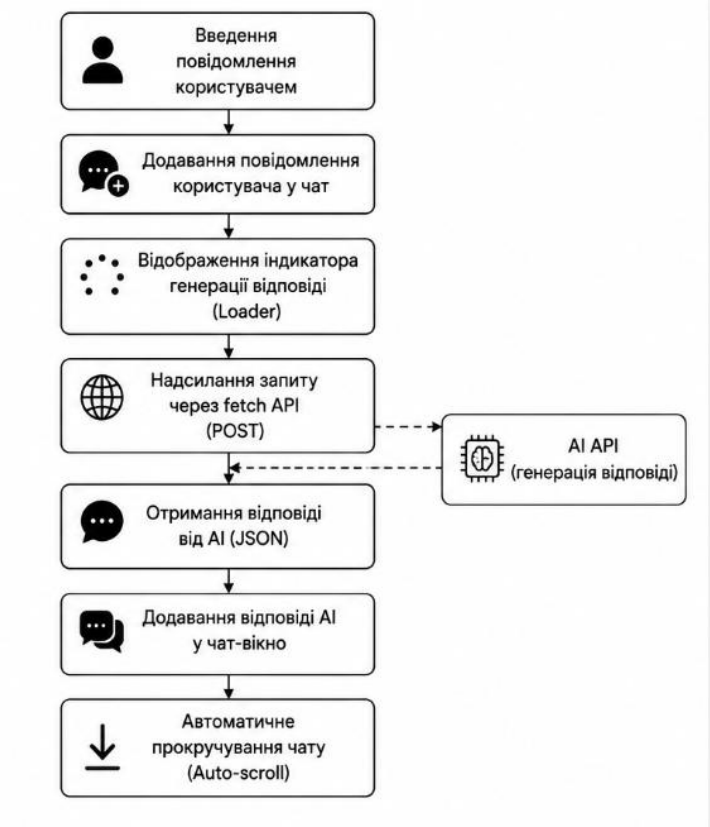

Рисунок 3.4 – Процес обробки повідомлення користувача в AI-консультанті

Взаємодія між користувачем і системою реалізована через асинхронні HTTP-запити у форматі JSON. Клієнтська частина формує запит і передає його на сервер, після чого відбувається обробка через AI-модуль, який інтегровано з зовнішнім сервісом OpenAI, розгорнутим у хмарній інфраструктурі Microsoft Azure [24].

Схема взаємодії користувача з AI-консультантом демонструє загальну архітектуру обміну даними між клієнтом, сервером та AI-сервісом, що наведено на рисунку 3.5.

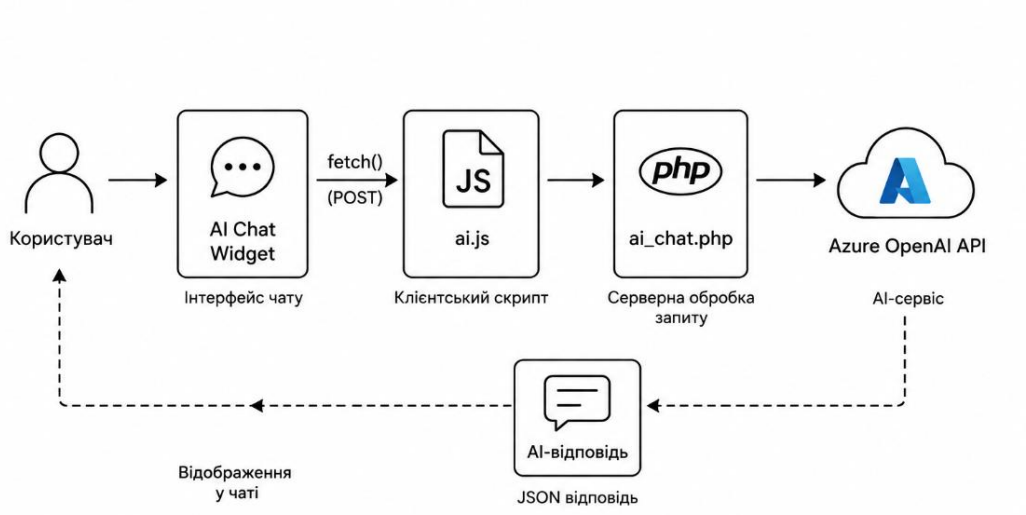

Рисунок 3.5 – Схема взаємодії користувача з AI-консультантом

На клієнтському рівні кожне повідомлення користувача проходить попередню обробку: видалення зайвих пробілів та перевірку на пусте значення. Після цього формується структура запиту, яка містить роль system для визначення поведінки моделі та роль user для передавання тексту запиту.

Лістинг 3.3 – Формування структури запиту до AI

```
    messages:                                                    [
  {       role:       "system",       content:       "..."       },
  {        role:        "user",        content:        text        }
]
```

Серверна частина відповідає за передачу запиту до AI-сервісу та отримання відповіді у форматі JSON. Після декодування отриманих даних результат повертається клієнтській частині, де відображається у чаті без перезавантаження сторінки. Такий підхід забезпечує інтерактивність системи та створює ефект реального діалогу.

Окремо слід зазначити, що система не лише обробляє запити, але й фактично формує історію взаємодії користувача з AI-консультантом у межах поточного сеансу. Кожне нове повідомлення додається до діалогу, що дозволяє моделі враховувати попередній контекст і надавати більш точні відповіді.

Інтерфейс системи підказок реалізовано у вигляді плаваючого чат-віджета, який доступний на основних сторінках вебдодатку. Відкриття та закриття чату здійснюється без перезавантаження сторінки шляхом зміни CSS-класів, що забезпечує швидку та зручну взаємодію користувача з системою.

Реалізована система підказок та історії пошуку дозволяє значно підвищити зручність використання вебдодатку, забезпечує інтерактивну підтримку користувача та створює основу для подальшого розвитку персоналізованих рекомендацій і розширеної аналітики запитів.

3.4 Оптимізація алгоритмів пошуку інформації на вебсайті за допомогою машинного навчання

3.4.1 Адміністративна панель сайту

Для забезпечення контролю роботи AI-консультанта та управління контентом у вебдодатку UrbanWear передбачено реалізацію адміністративної панелі. Основне призначення даного модуля полягає у моніторингу взаємодії користувачів із системою, перегляді AI-запитів, контролі відповідей мовної моделі та адмініструванні інформаційного наповнення інтернет-магазину. Архітектура модуля побудована за принципом розмежування клієнтської та серверної логіки, що забезпечує зручність масштабування та подальшого розвитку системи.

Серверна частина адміністративної панелі використовує PHP-сесії для перевірки авторизації адміністратора та обмеження доступу до службового функціоналу. Після успішного входу система надає доступ до інтерфейсу моніторингу, де можуть відображатися дані про активність AI-консультанта, кількість звернень користувачів та результати обробки повідомлень. Для зберігання інформації про користувачів і запити можуть використовуватися JSON-файли, що відповідає загальній архітектурі вебдодатку.

Одним із ключових елементів адміністративного модуля є контроль роботи AI-консультанта. Після отримання повідомлення користувача система може фіксувати текст запиту, час звернення та відповідь AI-моделі. Це дозволяє аналізувати ефективність роботи мовної моделі, виявляти помилки генерації відповідей та покращувати якість консультацій. Основою взаємодії із AZURE OPENAI є структурований JSON-запит, що містить повідомлення користувача та службову інформацію для моделі.

Лістинг 3.8 – Формування AI-запиту для обробки повідомлень

```
messages: [
   { role: "system", content: "..." },
   { role: "user", content: text }
]
```

Адміністративна панель також може використовуватись для керування контентом інтернет-магазину. Завдяки модульній структурі системи адміністратор отримує можливість змінювати інформацію про товари, редагувати описи, оновлювати ціни та контролювати відображення контенту у каталозі. Оскільки товари у поточній реалізації зберігаються у вигляді PHP-масиву, оновлення інформації виконується через серверну логіку відповідних файлів.

Інтерфейс адміністративної панелі реалізується із використанням Tailwind CSS та адаптивної структури сторінки. Для відображення інформації можуть використовуватись таблиці, картки статистики та блоки повідомлень, що забезпечують швидкий доступ до даних системи. Важливою особливістю є підтримка асинхронної взаємодії через JavaScript, що дозволяє оновлювати окремі елементи інтерфейсу без перезавантаження сторінки.

Реалізований підхід дозволяє організувати централізований контроль AI-функціоналу та контенту вебдодатку. Наявність адміністративної панелі підвищує ефективність підтримки системи, спрощує аналіз роботи LLM та

створює основу для подальшого розвитку механізмів аналітики, автоматичного модераторства й управління користувацькими запитами.

3.4.2 Реалізація головної сторінки сайту

Головна сторінка вебдодатку UrbanWear реалізована у файлі home page та виконує роль презентаційного елемента інтернет-магазину. Її основне призначення полягає у демонстрації бренду, відображенні ключових переваг магазину та забезпеченні швидкого доступу до каталогу товарів. Інтерфейс побудований із використанням HTML5, PHP та Tailwind CSS, що дозволило реалізувати сучасний адаптивний дизайн із підтримкою різних типів пристроїв. Загальний вигляд головної сторінки представлено на рисунку 3.6.

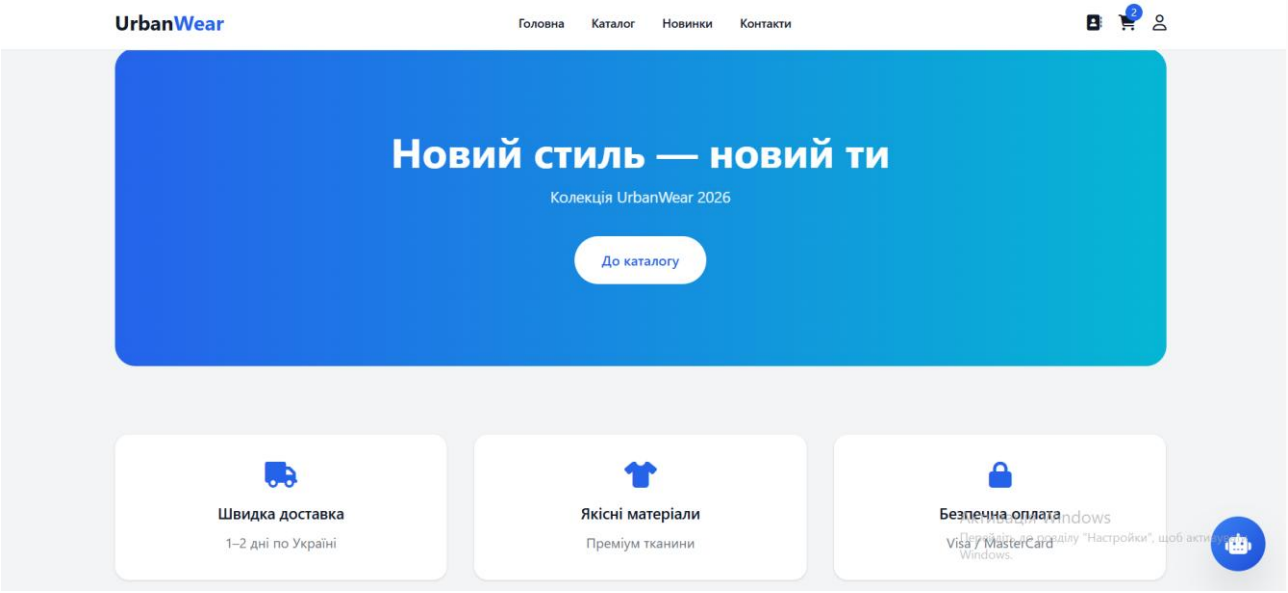

Рисунок 3.6 – Інтерфейс головної сторінки вебдодатку UrbanWear

Структура сторінки побудована за модульним принципом. На початку файлу підключається спільний компонент header.php, який містить навігаційне меню, стилі та службові елементи сторінки. Аналогічно в нижній частині підключається footer.php. Такий підхід дозволяє уникнути дублювання коду та забезпечує єдиний стиль для всіх сторінок вебдодатку. Основний контент

головної сторінки складається з декількох функціональних блоків: промо-секції, блоку переваг магазину та інтегрованого AI-консультанта [21].

Першим елементом інтерфейсу є Hero Section – великий промо-блок, який привертає увагу користувача після відкриття сайту. Для його реалізації використовується градієнтний фон, адаптивні відступи та анімації наведення. Основна структура блоку реалізована наступним чином:

Лістинг 3.4 – Hero Section головної сторінки

```
<section class="py-24 text-center bg-gradient-to-r from-blue-600 to-cyan-500 text-white rounded-3xl mt-10">
```

У межах Hero Section розміщено заголовок нової колекції, короткий опис бренду та кнопку переходу до каталогу товарів. Для стилізації використано класи Tailwind CSS, зокрема bg-gradient-to-r для створення градієнта, rounded-3xl для заокруглення кутів та hover-ефекти для інтерактивності елементів.

Наступним компонентом сторінки є блок переваг магазину, який реалізовано у вигляді адаптивної сітки CSS Grid. Кожен елемент містить іконку Font Awesome, заголовок та короткий опис переваги магазину: швидка доставка, якісні матеріали та безпечна оплата. Завдяки використанню класів grid-cols-1 та md:grid-cols-3 інтерфейс автоматично адаптується до розміру екрана, змінюючи кількість колонок для мобільних пристроїв та десктопів.

Окрему роль у структурі головної сторінки виконує AI-консультант, реалізований у вигляді плаваючого чат-віджета. Кнопка відкриття чату закріплена у правому нижньому куті екрана та використовує hover-анімації масштабування. Сам чат реалізований як fixed-блок із власною структурою повідомлень, формою введення та автоматичним скролом. Взаємодія із AI виконується через JavaScript-модуль ai.js, який відповідає за відкриття чату, надсилання повідомлень та оновлення DOM без перезавантаження сторінки.

Важливою особливістю реалізації інтерфейсу є його адаптивність та орієнтація на сучасний UX/UI-підхід. Для стилізації використовуються

компоненти Tailwind CSS, кастомні CSS-анімації та media-запити, що забезпечують коректне відображення сторінки на смартфонах, планшетах і десктопних пристроях. Реалізований інтерфейс створює зручне середовище для взаємодії користувача з інтернет-магазином та формує основу для подальшого розширення функціоналу системи.

3.4.3 Реалізація особистого кабінету користувача

Особистий кабінет користувача реалізований у файлі profile.php та призначений для перегляду й редагування персональної інформації користувача інтернет-магазину UrbanWear. Основною функцією модуля є забезпечення централізованого доступу до даних профілю, а також підтримка механізму оновлення інформації без повторної авторизації у системі. Реалізація модуля поєднує серверну логіку PHP, роботу із сесіями та адаптивний інтерфейс на основі Tailwind CSS.

На початку роботи файл запускає PHP-сесію через session_start() та перевіряє факт авторизації користувача. Якщо користувач не увійшов у систему, виконується автоматичне перенаправлення на сторінку login.php. Такий механізм дозволяє обмежити доступ до персональних даних стороннім користувачам та забезпечує базовий рівень безпеки системи.

Дані користувача отримуються із сесії та автоматично підставляються у форму редагування профілю. У формі передбачено поля для імені, прізвища, електронної пошти та номера телефону. Для запобігання некоректному відображенню даних використовується функція htmlspecialchars(), яка забезпечує безпечне виведення текстової інформації у HTML-структуру сторінки.

Після надсилання форми система виконує обробку POST-запиту та отримує оновлені дані користувача. Для очищення введених значень застосовується функція trim(), яка видаляє зайві пробіли. Далі серверна частина

відкриває JSON-файл users.json, що виконує роль спрощеної бази даних користувачів, та виконує пошук необхідного запису за email користувача.

Лістинг 3.5 – Оновлення даних користувача

```
foreach ($users as &$u) {
   if ($u['email'] === $user['email']) {

      $u['first_name'] = $first_name;
      $u['last_name']  = $last_name;
      $u['phone']      = $phone;
   }
}
```

Після оновлення необхідних полів система виконує перезапис JSON-файлу через file_put_contents(), що забезпечує збереження змін. Додатково виконується синхронізація сесії користувача, завдяки чому оновлені дані відображаються без повторного входу у систему. У випадку успішного збереження користувачу виводиться повідомлення про успішне оновлення профілю.

Інтерфейс особистого кабінету реалізовано у вигляді центрованої адаптивної форми із використанням Tailwind CSS. Для полів введення застосовано заокруглення, тіні та focus-ефекти, що покращують візуальне сприйняття інтерфейсу. Додатково реалізовано кнопку виходу з акаунту, яка виконує завершення сесії користувача та перенаправляє його на сторінку авторизації.

Реалізований модуль особистого кабінету забезпечує базову систему керування персональними даними користувача, підтримує модульну архітектуру системи та створює основу для подальшого розширення функціоналу, зокрема додавання історії замовлень, налаштувань акаунта та персоналізованих рекомендацій.

3.5 Забезпечення продуктивності та безпеки вебдодатку

Забезпечення продуктивності та безпеки вебдодатку інтернет-магазину є одним із ключових аспектів при розробці сучасних вебсистем, оскільки безпосередньо впливає на швидкість роботи сервісу, стабільність взаємодії користувачів та захист даних. У межах реалізації проєкту UrbanWear було застосовано комплексний підхід, який охоплює як оптимізацію серверної та клієнтської частини, так і впровадження базових механізмів захисту від типових загроз вебсередовища.

Підвищення продуктивності вебдодатку досягається за рахунок використання асинхронних запитів між клієнтською та серверною частиною. Застосування AJAX та fetch-запитів дозволяє виконувати обмін даними без перезавантаження сторінки, що значно зменшує навантаження на сервер і покращує швидкість відгуку інтерфейсу. Особливо це помітно у функціях пошуку товарів, автодоповнення та роботі AI-консультанта, де результати генеруються та відображаються в режимі реального часу.

Додатковим фактором оптимізації є використання обмеження кількості даних, що передаються між сервером і клієнтом. Наприклад, при пошуку товарів повертається лише частина результатів (найбільш релевантні позиції), що дозволяє зменшити обсяг переданих даних та пришвидшити обробку запитів. Аналогічний підхід використовується у відображенні каталогу товарів із застосуванням пагінації, що запобігає перевантаженню сторінки великою кількістю елементів одночасно.

Окрему роль у забезпеченні продуктивності відіграє правильна організація структури коду. Поділ функціоналу на модулі (header, footer, окремі PHP-сторінки для каталогу, товару, кошика та профілю) дозволяє уникнути дублювання коду та спрощує підтримку системи. Це також позитивно впливає на масштабованість вебдодатку та полегшує подальше розширення функціоналу.

Щодо безпеки, у проєкті реалізовано базові механізми захисту користувацьких даних. Для роботи з формами використовується перевірка вхідних даних, зокрема очищення текстових полів від зайвих символів та перевірка на порожні значення. Це дозволяє мінімізувати ризики некоректного введення даних та запобігти помилкам у роботі системи.

Також застосовується захист від підміни даних через URL-параметри шляхом перевірки існування товарів у базових структурах перед їх відображенням. У випадку кошика та оформлення замовлення використовується серверна перевірка сесій, що дозволяє гарантувати, що доступ до функцій мають лише авторизовані користувачі.

Значна увага приділяється роботі з сесіями користувачів. Дані кошика та профілю зберігаються у PHP-сесіях, що забезпечує їх ізольованість для кожного користувача та запобігає несанкціонованому доступу до чужої інформації. Додатково при оформленні замовлення виконується перевірка авторизації, що унеможливлює виконання критичних операцій без входу в систему.

У контексті взаємодії з зовнішніми сервісами, зокрема AI-модулем, використовується передача даних у форматі JSON через HTTP-запити. Для цього реалізовано базову валідацію введених користувачем даних перед відправкою на сервер, що знижує ризик некоректних або шкідливих запитів до API. Водночас передбачено обробку помилок з боку сервера, що дозволяє уникнути критичних збоїв у роботі інтерфейсу.

Окремо варто відзначити використання захисту від дублювання запитів та повторного надсилання форм, зокрема через перенаправлення після виконання операцій (наприклад, після додавання товару до кошика або оформлення замовлення). Це дозволяє уникнути випадкового повторного виконання дій користувача.

Реалізовані механізми продуктивності та безпеки забезпечують стабільну роботу вебдодатку навіть при активному навантаженні, а також створюють базовий рівень захисту даних користувачів. Подальший розвиток системи може включати впровадження більш складних механізмів безпеки, таких як CSRF-

захист, шифрування чутливих даних та використання розширених систем автентифікації.

3.6 Висновки до третього розділу

У межах даного розділу було здійснено практичну реалізацію основних функціональних компонентів веб-сайту з інтегрованою пошуковою системою. Розробка охопила всі ключові елементи сучасного інтерфейсу взаємодії користувача з торговельною платформою: від стартової сторінки з рекомендаційними блоками до особистого кабінету, адміністративної панелі та системи динамічних підказок.

Особливу увагу приділено архітектурному підходу до побудови веб-додатку, що базується на розподіленій логіці клієнт-серверної взаємодії, використанні асинхронних запитів та гнучкому компонентному поділі. Це дозволило забезпечити масштабованість, високу швидкодію й адаптивність ресурсу для різних категорій користувачів.

Розроблена пошукова система з автоматичним доповненням запиту та історією пошуку створює умови для швидкого доступу до релевантних товарів, що значно покращує користувацький досвід. Впроваджені рекомендаційні блоки та персоналізовані елементи інтерфейсу сприяють зростанню залученості відвідувачів і підвищенню конверсії.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Обов’язкові Обовязки роботодавця щодо створення безпечних і нешкідливих умов праці

Згідно з чинним законодавством роботодавець зобов’язаний створити на робочому місці в кожному структурному підрозділі належні умови праці відповідно до нормативно-правових актів, а також забезпечити дотримання вимог законодавства щодо прав працівників з охорони праці [50].

З цією метою роботодавець забезпечує функціонування системи управління охороною праці, а саме:

– створює відповідні служби і призначає посадових осіб, які забезпечують вирішення конкретних питань охорони праці, затверджує інструкції про їх обов'язки, права та відповідальність за виконання покладених на них функцій, а також контролює їх додержання;

– розробляє за участю сторін колективного договору і реалізує комплексні заходи для досягнення встановлених нормативів та підвищення існуючого рівня охорони праці;

– забезпечує виконання необхідних профілактичних заходів відповідно до обставин, що змінюються;

– впроваджує прогресивні технології, досягнення науки і техніки, засоби механізації та автоматизації виробництва, вимоги ергономіки, позитивний досвід з охорони праці тощо;

– забезпечує належне утримання будівель і споруд, виробничого обладнання та устаткування, моніторинг за їх технічним станом;

– забезпечує усунення причин, що призводять до нещасних випадків, професійних захворювань, та здійснення профілактичних заходів, визначених комісіями за підсумками розслідування цих причин;

– організовує проведення аудиту охорони праці, лабораторних досліджень умов праці, оцінку технічного стану виробничого обладнання та

устаткування, атестацій робочих місць на відповідність нормативно-правовим актам з охорони праці в порядку і строки, що визначаються законодавством, та за їх підсумками вживає заходів до усунення небезпечних і шкідливих для здоров'я виробничих факторів;

– розробляє і затверджує положення, інструкції, інші акти з охорони праці, що діють у межах підприємства (далі - акти підприємства), та встановлюють правила виконання робіт і поведінки працівників на території підприємства, у виробничих приміщеннях, на будівельних майданчиках, робочих місцях відповідно до нормативно-правових актів з охорони праці, забезпечує безоплатно працівників нормативно-правовими актами та актами підприємства з охорони праці;

– здійснює контроль за додержанням працівником технологічних процесів, правил поводження з машинами, механізмами, устаткуванням та іншими засобами виробництва, використанням засобів колективного та індивідуального захисту, виконанням робіт відповідно до вимог з охорони праці;

– організовує пропаганду безпечних методів праці та співробітництво з працівниками у галузі охорони праці;

– вживає термінових заходів для допомоги потерпілим, залучає за необхідності професійні аварійно-рятувальні формування у разі виникнення на підприємстві аварій та нещасних випадків.

Роботодавець зобов’язаний забезпечити за свій рахунок придбання, комплектування, видачу та утримання засобів індивідуального захисту відповідно до нормативно-правових актів з охорони праці та колективного договору, а уразі передчасного зношення цих засобів не з вини працівника, замінити їх за свій рахунок [51].

Особлива увага має приділятися виявленню та усуненню причин, що можуть призвести до нещасних випадків, професійних захворювань, здійсненню профілактичних заходів з метою недопущення аварії на виробництві. Для цього проводяться лабораторні дослідження умов праці,

аналізується технічний стан виробничого обладнання та устаткування, здійснюється атестація робочих місць на відповідність їх нормативно-правовим актам з охорони праці, за підсумками якої роботодавець розробляє та впроваджує заходи усунення небезпечних і шкідливих для здоров'я виробничих факторів [50].

Роботодавець через створену ним службу з охорони праці, комісію з питань охорони праці здійснює контроль за додержанням працівниками вимог виробничої санітарії, гігієни праці, техніки безпеки, використання засобів колективного та індивідуального захисту, виконання робіт згідно з розробленими і затвердженими на підприємстві положеннями, інструкціями та іншими актами з охорони праці.

У свою чергу, працівники, виконуючи свої трудові обов'язки, повинні дотримуватись трудової і технічної дисципліни, підвищувати продуктивність та якість праці.

Згідно з Законодавством України про охорону праці, зокрема статтею 18 Закону України «Про охорону праці», роботодавець зобов'язаний організувати навчання, інструктаж і перевірку знань з питань охорони праці для всіх працівників. Також це передбачено типовим положенням про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженим наказом Держгірпромнагляду № 15 від 26.01.2005 року а потім, Державній службі України з питань праці (Держпраці) згідно з розпорядженням Кабінету Міністрів України № 1021-р від 30 вересня 2015 року.

Основним документом, який регламентує взаємовідносини між трудовим колективом і роботодавцем, є колективний договір. Цей договір розробляється роботодавцем та профспілковою організацією підприємства і затверджується на зборах (конференції) трудового колективу.

Для запланованих заходів з охорони праці роботодавець зобов'язаний виділити цільові кошти та необхідні матеріальні ресурси, витрачати які на інші цілі заборонено. Порядок використання цих коштів визначається у колективному договорі і контролюється трудовим колективом.

До обов’язків роботодавця входить своєчасне проведення загальнообов’язкового державного соціального страхування працівників від нещасного випадку на виробництві та професійного захворювання, вживання термінових заходів для допомоги потерпілим, у т.ч. залучення за необхідності професійних аварійно-рятувальних формувань, вести облік і розслідування нещасних випадків, професійних захворювань і аварій на виробництві.

Роботодавець також може за рахунок власних коштів здійснювати додаткові виплати потерпілим працівникам і членам їх сімей відповідно до колективного або трудового договору.

4.2 Правило техніки безпеки при експлуатації обладнання

На підприємствах різного типу зокрема, в офісних приміщеннях для роботи широко застосовується комп’ютерне та офісне електронне обладнання (ПК, принтери, сервери, мережеві пристрої тощо). Безпечна експлуатація такого обладнання регламентується чинними нормативно-правовими актами, зокрема ДСТУ EN 60204-1:2015 «Безпечність машин. Електрообладнання машин. Загальні вимоги», ДСТУ IEC 60364-7-705:2006 щодо електробезпеки, а також вимогами Правил охорони праці під час використання комп’ютерної техніки в офісах. Ці документи встановлюють основні вимоги до монтажу, експлуатації, заземлення, організації електроживлення та запобігання електротравмам працівників [52].

До експлуатації цього устаткування допускаються особи, навчені за програмою технічного мінімуму, що пройшли інструктаж на робочому місці та стажування. Вони повинні бути забезпечені інструкціями безпечної експлуатації обслуговуваного устаткування і ознайомлені під розписку з їх змістом.

Небезпека електромеханічного устаткування для обслуговуючого персоналу обумовлена наявністю в ньому електроприводу і робочих органів, які

рухаються з великою швидкістю, а також і можливістю розлітання частинок оброблюваних сировини і продуктів.

Перед ввікненням машин в роботу необхідно перевіряти стан заземлення (занулення), цілісність ізоляції кабелю, надійність кріплення змінних механізмів, наявність і справність огорож, відсутність яких-небудь предметів у завантажувальних пристроях і робочих камерах.

Перед початком роботи з комп’ютерним обладнанням (ПК, принтерами, мережевими пристроями тощо) слід переконатися в його справності: перевірити цілісність кабелів живлення, відсутність пошкоджень на корпусах і роз'ємах, а також справну роботу вентиляції. Роботу з технікою варто починати тільки після увімкнення обладнання у звичайному режимі без навантаження (холостий запуск), щоб виключити сторонні шуми або збої.

У разі появи сторонніх шумів, підвищеного нагрівання корпусу або кабелів, запаху гару чи збоїв у роботі пристроїв необхідно негайно вимкнути обладнання за допомогою кнопки живлення або витягнути вилку з розетки та повідомити відповідальну особу.

Після завершення роботи комп’ютерне обладнання слід коректно вимкнути через програмні засоби («Завершення роботи»), після чого вимкнути живлення (через кнопку на корпусі або від'єднання від мережі). Обладнання необхідно очистити від пилу та забруднень відповідними засобами (серветками, щітками, антистатичними засобами).

Для запобігання перегріву комп’ютерної техніки, особливо серверів, блоків живлення, процесорів, принтерів тощо, необхідно забезпечити належну вентиляцію. Рекомендується використовувати локальні вентиляційні або охолоджувальні системи, зокрема витяжні вентилятори або кондиціонери, а також регулярно очищувати вентиляційні отвори.

Комп’ютерне обладнання, як і інші електроприлади, становить потенційну небезпеку через наявність джерел електричного струму та нагрівальних елементів. Тому важливо дотримуватись правил електробезпеки

та експлуатації згідно з технічною документацією та чинними нормативними актами.

Перед ввімкненням будь-якого апарату в роботу необхідно перевірити наявність і стан запобіжних захисних засобів і сигнальних пристроїв, заземлення (занулення) корпусу, а також цілісність ізоляції кабелю і елементів штепсельного роз'єму. Для кожного виду електронагрівальних апаратів розроблені особливі вимоги безпеки під час експлуатації [49].

Експлуатація таких апаратів має бути припинена у разі відмови запобіжних захисних засобів і сигнальних пристроїв, неконтрольованого підвищення температури електронагрівників, а деяких з них – у разі пониження рівня води нижче допустимого значення, надмірного підвищення тиску пари [49].

Після закінчення роботи електронагрівальний апарат вимикають перемикачем, що є на його корпусі, і вимикають з електричної мережі. Потім видаляють з нього залишки води або продукту.

Технічне обслуговування і ремонт електромеханічного і електронагрівального устаткування проводять за договором механіки ремонтно-монтажних комбінатів, сервісних організацій.

Отже, безпечна експлуатація комп’ютерного та офісного електронного обладнання вимагає дотримання встановлених норм і правил, що регламентуються відповідними нормативно-правовими актами. Застосування технічних засобів контролю, своєчасне технічне обслуговування та належна організація робочого процесу дозволяють знизити ризики електротравм і технічних збоїв в офісних умовах.

4.3 Забезпечення захисту населення від впливу іонізуючих випромінювань

Іонізуюче випромінювання – одна з найбільш серйозних загроз для здоров’я людини у сучасному технологічному світі. У зв’язку з розвитком

атомної енергетики, використанням джерел іонізуючого випромінювання у промисловості, медицині та науці, питання забезпечення радіаційного захисту населення набуває першочергового значення для державної політики в галузі охорони здоров'я, екології та національної безпеки.

Право кожної особи, яка перебуває на території України, на захист від впливу іонізуючого випромінювання закріплене у статті 3 Закону України «Про захист людини від впливу іонізуючого випромінювання». Це право реалізується через комплекс превентивних і компенсаційних заходів, спрямованих на обмеження доз опромінення, інформування населення, державний контроль за джерелами випромінювання та регламентоване нормування ризиків [46].

Основу радіаційного захисту становить система дозових обмежень. Для осіб з населення встановлено ліміт ефективної дози опромінення в розмірі 1 мЗв на рік, а для професійного персоналу – до 20 мЗв на рік у середньому, з можливістю досягнення 50 мЗв у межах окремого року. Окремо враховуються особливі категорії: вагітні жінки, молодь віком до 18 років, здобувачі освіти, медичний персонал, а також працівники, що виконують тимчасові завдання з джерелами випромінювання. Для кожної з цих груп встановлюються конкретні межі доз навантаження на критичні органи з урахуванням радіочутливості тканин та довгострокових ризиків [46].

Ключову роль у захисті населення відіграють державні норми радіаційної безпеки – НРБУ-97, що регламентують класифікацію осіб за категоріями опромінення (А, Б, В) і встановлюють допустимі рівні доз. До лімітів не включаються природні джерела фону, медичні обстеження, техногенні фонові джерела та аварійні дози. Водночас, у разі настання радіаційних інцидентів передбачено процедури укриття, йодної профілактики та тимчасової евакуації населення, залежно від очікуваних доз навантаження на щитовидну залозу або ефективної дози тіла [47].

Особливу увагу приділено трудовій діяльності з джерелами іонізуючого випромінювання. Юридичні особи та ФОП, які здійснюють таку діяльність, зобов'язані організовувати систематичний контроль за радіаційним фоном,

вести облік індивідуальних доз, інформувати персонал, розробляти заходи безпеки та планувати дії у разі радіаційної аварії. До контролю також входять медичні огляди, звітність до державного дозиметричного реєстру та дотримання умов ліцензування.

Важливою складовою є захист під час медичного опромінення, що підпорядковується принципам оптимізації й обґрунтування. Це означає, що будь-яке втручання повинно бути не тільки виправданим, а й максимально безпечним з урахуванням ефективності альтернативних методів. Інформація про дози, отримані під час медичних процедур, підлягає збереженню протягом 50 років, а пацієнт має право бути поінформованим про ризики і навіть відмовитися від процедури, за винятком випадків епідеміологічної небезпеки.

Суттєвим елементом системи захисту є механізм компенсації та відшкодування шкоди, передбачений статтями 19 і 20 Закону. У разі перевищення дозових лімітів з незалежних від особи причин, їй надається грошова компенсація. У разі заподіяння шкоди здоров'ю, життю чи майну внаслідок іонізуючого випромінювання, передбачено обов'язок винного суб'єкта здійснити повне відшкодування. Такі випадки підлягають судовому розгляду з урахуванням доказів дозового перевищення [48].

Забезпечення ефективного захисту населення від іонізуючого випромінювання неможливе без урахування міжнародного досвіду та співпраці із профільними організаціями, які встановлюють глобальні стандарти у цій сфері. Одними з ключових суб'єктів міжнародного регулювання є Міжнародне агентство з атомної енергії (МАГАТЕ), Всесвітня організація охорони здоров'я (ВООЗ), Європейська комісія, а також Міжнародна комісія з радіаційного захисту (ICRP).

МАГАТЕ відіграє провідну роль у формуванні загальнообов'язкових стандартів безпеки, координує зусилля країн-членів у розробці політик захисту та забезпечує технічну підтримку у впровадженні відповідного законодавства. Документи МАГАТЕ, зокрема "Основні міжнародні стандарти безпеки" (GSR Part 3), містять вимоги до захисту здоров'я і безпеки працівників, населення та

навколишнього середовища від шкідливого впливу іонізуючого випромінювання.

ВООЗ, у свою чергу, акцентує увагу на питаннях громадського здоров'я та сприяє гармонізації підходів до медичного опромінення населення. Її рекомендації враховуються при розробці політик оптимізації доз опромінення в системах охорони здоров'я.

Значний внесок у розвиток доктрини радіаційного захисту здійснює Міжнародна комісія з радіаційного захисту (ICRP), яка формулює науково обґрунтовані принципи, зокрема, один із основоположних – принцип ALARA (As Low As Reasonably Achievable). Цей принцип полягає у мінімізації доз опромінення до настільки низьких рівнів, наскільки це можливо і доцільно, враховуючи економічні та соціальні фактори. ALARA є основою для багатьох національних політик та практик у сфері радіаційного захисту.

Відповідно, Україна як учасник міжнародного співтовариства активно інтегрує положення міжнародних нормативних документів у національне законодавство, зокрема через адаптацію Державних норм радіаційної безпеки (НРБУ-97) до сучасних вимог. Такий підхід дозволяє забезпечити відповідність системи радіаційного захисту України сучасним європейським та світовим стандартам, а також гарантувати ефективну взаємодію у випадку транскордонних радіаційних інцидентів або надзвичайних ситуацій.

Ефективне функціонування системи радіаційного захисту неможливе без належного рівня освітньої підготовки та інформованості населення. Знання основних принципів поводження з джерелами іонізуючого випромінювання, розуміння механізмів впливу на організм та обізнаність щодо алгоритмів дій у надзвичайних ситуаціях є важливими елементами національної безпеки та захисту здоров'я громадян.

Інформаційно-просвітницька діяльність охоплює широке коло завдань – від базових шкільних курсів із фізики та основ безпеки життєдіяльності до спеціалізованих програм професійної підготовки для медичних працівників, рятувальних служб, персоналу атомної енергетики та інших критичних галузей.

Особливу увагу слід приділяти розробці навчальних модулів, які адаптовані до вікових особливостей та професійних ризиків відповідних аудиторій.

У закладах освіти рекомендовано включати в навчальні програми тематику, пов’язану з радіаційною безпекою, зокрема з історичним аналізом радіаційних аварій (таких як Чорнобиль чи Фукусіма), основами дозиметрії, захисними методами та першочерговими діями у разі радіаційної небезпеки. Застосування інтерактивних методів навчання, таких як симуляції аварійних ситуацій, електронні курси та тренінги, сприяє підвищенню рівня сприйняття та засвоєння матеріалу.

Медичні працівники мають володіти не лише знаннями щодо безпечного використання діагностичного та терапевтичного обладнання, а й бути здатними надавати роз’яснення пацієнтам щодо радіаційних ризиків, особливо у випадках застосування процедур, пов’язаних із високим рівнем опромінення. Для цього запроваджуються системи післядипломної освіти, періодичної сертифікації та участі у тематичних конференціях.

З метою формування стійкої культури безпеки також проводяться громадські інформаційні кампанії, видаються брошури, буклети, створюються теле- та радіопрограми, що висвітлюють питання радіаційного захисту у доступній для широкої аудиторії формі. Розвиток цифрових платформ дає змогу оперативно поширювати важливу інформацію у випадку реальної або потенційної радіаційної загрози.

У контексті воєнного часу або загрози терористичних актів із використанням радіоактивних матеріалів, підвищення обізнаності населення набуває ще більшого значення. Навчене населення здатне зменшити ступінь паніки, діяти більш організовано та знизити рівень потенційного ураження.

Загалом, система забезпечення радіаційної безпеки в Україні є комплексною, багаторівневою і передбачає як нормативно-правове регулювання, так і науково обґрунтовану стратегію технічного контролю. З огляду на наявність радіаційних ризиків, зокрема внаслідок Чорнобильської катастрофи, в умовах воєнного часу або розвитку ядерної енергетики,

забезпечення ефективного захисту населення від впливу іонізуючих випромінювань залишається одним із пріоритетних напрямів державної політики у сфері охорони здоров’я та безпеки довкілля [47].

4.4 Висновки до четвертого розділу

У четвертому розділі було розглянуто питання про охорону праці, що мають ключове значення в забезпеченні належних умов для збереження життя, здоров’я та працездатності працівників і населення. Зокрема, було досліджено обов’язки роботодавця щодо створення безпечних і нешкідливих умов праці, які передбачають організацію системного управління охороною праці, забезпечення працівників інструкціями та засобами індивідуального захисту, проведення інструктажів і навчань, контроль за дотриманням правил техніки безпеки, а також вжиття заходів з профілактики виробничого травматизму та професійних захворювань.

Окрему увагу було приділено питанням техніки безпеки під час експлуатації комп’ютерного та електронного офісного обладнання. Згідно з чинними нормативно-правовими актами, безпечна робота з таким устаткуванням вимагає дотримання вимог електробезпеки, технічної справності обладнання, належної організації вентиляції, проведення регулярного технічного обслуговування та усунення потенційних ризиків, пов’язаних із перегрівом або пошкодженням електричних компонентів.

Крім того, розділ містив аналіз правових та організаційних засад забезпечення захисту населення від іонізуючого випромінювання. Було акцентовано на важливості дотримання нормативних обмежень ефективної дози опромінення для різних категорій осіб, запровадженні превентивних заходів, контролі за джерелами випромінювання, а також ролі державних норм радіаційної безпеки у мінімізації негативного впливу радіації на здоров’я населення.

ВИСНОВКИ

У кваліфікаційній роботі здійснено дослідження та практичну реалізацію вебдодатка з інтеграцією великих мовних моделей для автоматизації генерації відповідей на запити користувачів. Робота складається з чотирьох розділів, у яких розглянуто теоретичні, проєктні та практичні аспекти інтеграції LLM у вебсистеми.

У першому розділі кваліфікаційної роботи здійснено огляд сучасних великих мовних моделей та визначено особливості їх використання у вебсередовищі. Розглянуто методи інтеграції LLM у вебдодатки через API та мікросервісні підходи, що забезпечують масштабованість і гнучкість системи. Окрему увагу приділено можливостям платформи Azure OpenAI та перевагам застосування LLM для автоматизації взаємодії з користувачем. На основі аналізу обґрунтовано вибір технологій і сформовано вимоги до майбутньої системи.

У другому розділі кваліфікаційної роботи описано архітектуру та функціональну структуру вебдодатка. Розроблено механізми обробки та передачі запитів користувачів, що забезпечують ефективну взаємодію між клієнтською частиною, сервером та AI-сервісом. Спроєктовано логіку інтеграції чат-інтерфейсу з асинхронними процесами, що дозволяє оптимізувати роботу системи та підвищити якість обслуговування користувачів.

У третьому розділі кваліфікаційної роботи представлено реалізацію вебдодатка із чат-інтерфейсом, каталогом товарів, кошиком та особистим кабінетом. Здійснено інтеграцію із сервісом Azure OpenAI для генерації відповідей у режимі реального часу. Спроєктовано механізми обробки асинхронних запитів, що забезпечують стабільність роботи системи. Проведено тестування працездатності вебдодатка та здійснено оцінювання якості генерації відповідей, що підтвердило ефективність запропонованих рішень.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано нормативно-правову базу щодо обов’язкових медичних оглядів працівників окремих категорій, зокрема осіб, які працюють із комп’ютерною технікою. Описано особливості організації заходів безпеки у разі виникнення надзвичайних ситуацій техногенного та природного характеру. Розглянуто підходи до забезпечення стійкої роботи комп’ютеризованих систем в умовах дії електромагнітного імпульсу ядерного вибуху та запропоновано відповідні технічні й організаційні рішення.

ПЕРЕЛІК ДЖЕРЕЛ

1. Vyacheslav Nykytyuk, Vasyl Dozorskyy, Nataliia Kunanets, Volodymyr Pasichnyk, Oleksandr Matsiuk, Ihor Bodnarchuk: Electrical Probe-Signal Processing and Criterion for the Determination of Time Parameters of the Teeth Filling Material Polymerization Process in Dentistry. 4th IDDM 2021: Valencia, Spain. P. 54-63.

2. Oleksii Duda, Nataliia Kunanets, Serhii Martsenko, Vyacheslav Nykytyuk, Volodymyr Pasichnyk. Information technology platform for the selection and analytical processing of information on COVID-19. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). Volume 2, Lviv, Ukraine 22-25 Sept. 2021. P. 231-328. Electronic ISBN:978-1-6654-4257-2, Print on Demand(PoD) ISBN:978-1-6654-4258-9, Electronic ISSN: 2766-3639, Print on Demand(PoD) ISSN: 2766-3655. DOI: 10.1109/CSIT52700.2021.9648839.

3. Oleksii Duda, Nataliia Kunanets, Serhii Martsenko, Vyacheslav Nykytyuk, Volodymyr Pasichnyk. COVID-19 data collections and analytical processing. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). Volume 2, Lviv, Ukraine 22-25 Sept. 2021. P. 252-257. Electronic ISBN:978-1-6654-4257-2, Print on Demand (PoD) ISBN:978-1-6654-4258-9, Electronic ISSN: 2766-3639, Print on Demand (PoD) ISSN: 2766-3655. DOI: 10.1109/CSIT52700.2021.9648839.

4. Vyacheslav Nykytyuk, Vasil Dozorskyi, Oksana Dozorska, Andrii Karnaukhov and Liubomyr Matiichuk. The Method of User Identification by Speech Signal. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022) Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P.225-232. ISSN 1613-0073 DOI: 10.1425/jsdtl.

5. Ihor Bodnarchuk, Yuriy Skorenkyy, Taras Kramar, Oleksii Duda and Vyacheslav Nykytyuk. Use of Analytical Hierarchy Process in Scenarios Design for a Digital Museum with XR components. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022)

Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P. 414-425. ISSN 1613-0073 DOI: 10.1425/jsdtl.

6. Kryazhych O., Itskovych V., Iushchenko K., Hrytsyshyna V., Bruvier D., Nykytyuk V., Bodnarchuk I. (2023) The use of abstract moore automaton to control the sensors of a service-oriented alarm and emergency notification network. Scientific Journal of TNTU (Tern.), vol 109, no 1, pp. 111–120. ISSN 2522-4433

7. Dediv, L., Dozorska, O., Kukuruza, V., Nykytyuk, V., Kovalyk, S. Computer Simulation Modeling of Voice Signals in the Matlab Environment for the Task of Computerized Diagnostic Systems Testing. The 1st International Workshop on “Computer information technologies in Industry 4.0” (CITI-2023) will be held in Ternopil, Ukraine, from June 14 to 16, 2023. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. 2023, 3468, pp. 257–262. Vol-3468 urn:nbn:de:0074-3468-8, ISSN 1613-0073**.**

8. Dozorskyi, V., Dediv, I., Sverstiuk, S., Nykytyuk, V., Karnaukhov, A. The Method of Commands Identification to Voice Control of the Electric Wheelchair. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. The 1st International Workshop on “Computer information technologies in Industry 4.0” (CITI-2023) will be held in Ternopil, Ukraine, from June 14 to 16, 2023. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. *2023*, 3468, pp. 233–240. Vol-3468 urn:nbn:de:0074-3468-8, ISSN 1613-0073**.**

9. Sverstiuk, A., Matiichuk, L., Polyvana, U., Stanko, A., Nykytyuk, V.. Analytical analysis of approaches to assessing the quality of life in smart cities. BAIT’2024: The 1st International Workshop on “Bioinformatics and applied information technologies”, October 02-04, 2024, Zboriv, Ukraine. CEUR Workshop Proceedings, 2024, 3842, pp. 75–91. ISSN: 1613-0073**.**

10. Що таке велика мовна модель?, URL: https://www.sap.com/ukraine/resources/what-is-large-language-model (дата звернення: 17.05.2026)

11. Великі мовні моделі (LLM) та їх застосування, URL: https://wezom.com.ua/ua/blog/veliki-movni-modeli-llm (дата звернення: 17.05.2026)

12. Переваги власної LLM, URL: https://ua.linkedin.com/pulse/advantages-having-your-own-llm-devopsbay-1vuhc?tl=uk (дата звернення: 17.05.2026)

13. Як моделі великих мов змінюють електронну комерцію, URL: https://ua.linkedin.com/pulse/how-large-language-models-changing-ecommerce-algolia-dg1qe?tl=uk (дата звернення: 17.05.2026)

14. Large language models in education: a systematic review of empirical applications, benefits, and challenges, URL: https://www.sciencedirect.com/science/article/pii/S2666920X25001699 (дата звернення: 17.05.2026)

15. Розробляємо українську велику мовну модель (LLM) – покращуємо державні сервіси, підтримуємо бізнес і крокуємо до лідерства в ШІ, URL: https://thedigital.gov.ua/news/technologies/rozroblyaemo-ukrainsku-veliku-movnu-model-llm-pokrashchuemo-derzhavni-servisi-pidtrimuemo-biznes-i-krokuemo-do-liderstva-v-shi (дата звернення: 17.05.2026)

16. ClarifyGPT: A Framework for Enhancing LLM-Based Code Generation via Requirements Clarification, URL: https://dl.acm.org/doi/full/10.1145/3660810 (дата звернення: 17.05.2026)

17. Безпека LLM: Захист від Prompt Injection для промислових систем, URL: https://introl.com/uk/blog/llm-security-prompt-injection-defense-production-guide-2025 (дата звернення: 17.05.2026)

18. Four approaches to creating a specialized LLM, URL: https://stackoverflow.blog/2024/12/05/four-approaches-to-creating-a-specialized-llm/ (дата звернення: 17.05.2026)

19. What Is Prompt Engineering?, URL: https://www.oracle.com/ua/artificial-intelligence/prompt-engineering/ (дата звернення: 17.05.2026)

20. Як prompt engineering підвищує продуктивність AI моделей, URL: https://webbylab.com/uk/news/prompt-engineering/ (дата звернення: 17.05.2026)

21. Що таке фронтенд розробка: складові, етапи та технології, URL: https://wezom.com.ua/ua/blog/chto-takoe-front-end-razrabotka (дата звернення: 17.05.2026)

22. Що оцінюють LLM, URL: https://ua.linkedin.com/pulse/llms-evaluating-demystifying-autorater-nikita-namjoshi-38cic?tl=uk (дата звернення: 17.05.2026)

23. Як оцінити ефективність великих мовних моделей, URL: https://speka.ua/artificial-intelligence/yak-ociniti-efektivnist-velikix-movnix-modelei-9wnkgn (дата звернення: 17.05.2026)

24. Azure OpenAI, URL: https://azure.microsoft.com/en-us/products/ai-foundry/models/openai (дата звернення: 17.05.2026)

25. Що таке API?, URL: https://robotdreams.cc/uk/blog/192-shcho-take-api (дата звернення: 17.05.2026)

26. Google Cloud, URL: https://lnk.ua/WIlMYYHzf (дата звернення: 17.05.2026)

27. Монолітна та мікросервісна архітектура, URL: https://corewin.ua/blog/monolithic-vs-microservices-architecture-which-is-better-for-security/ (дата звернення: 17.05.2026)

28. Що таке API-орієнтована розробка?, URL: https://lnk.ua/pljtXmvGn (дата звернення: 17.05.2026)

29. Синхронні vs асинхронні підходи програмування, URL: https://robotdreams.cc/uk/blog/579-synhronni-vs-asynhronni-pidhody-programuvannya (дата звернення: 17.05.2026)

30. Serverless: що це таке і як це змінює підхід до розробки, URL: https://itproger.com/ua/news/serverless-chto-eto-takoe-i-kak-eto-menyaet-podhod-k-razrabotke (дата звернення: 17.05.2026)

31. Що таке Single Page Application?, URL: https://asabix.com.ua/what-is-a-single-page-application/ (дата звернення: 17.05.2026)

32. Модель клієнт-сервер, URL: https://www.vpnunlimited.com/ua/help/cybersecurity/client-server-model?srsltid=AfmBOop7L_jjTZRVbqMQv2tx8hBovwCMfwkrhOVobdOtOmmOzT4kOu1A (дата звернення: 17.05.2026)

33. PHP docs, URL: https://www.php.net/docs.php (дата звернення: 17.05.2026)

34. Ієрархічна база даних, URL: https://bazidanih5.blogspot.com/p/blog-page.html (дата звернення: 17.05.2026)

35. HTTP, URL: https://code.tutsplus.com/uk/http-the-protocol-every-web-developer-must-know-part-1--net-31177t (дата звернення: 17.05.2026)

36. В.В. Пасічник. Організація баз даних та знань / В.В. Пасічник, В.А. Резніченко, 2015 – с. 384.

37. А.Ю. Берко. Системи баз даних та знань / А.Ю. Берко, О.М. Верес, В.В. Пасічник, 2019 – с. 440.

38. М.М. Глибовець. Основи програмування / М.М. Глибовець, 2020 – с. 240.

39. М.М. Глибовець. Штучний інтелект / М.М. Глибовець, О.В. Олецький, 2002 – с. 366.

40. С.О. Субботін. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень / С.О. Субботін, 2008 – с. 228.

41. К. Шифлетт. Безпека Web-застосувань на PHP (Essential PHP Security) / К. Шифлетт, 2005 – с. 128.

42. M. Richards. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford, 2020 – с. 432.

43. M. Fowler. Patterns of Enterprise Application Architecture / M. Fowler, 2012 – с. 560.

44. R. Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (6th ed.) / R. Nixon, 2021 – с. 832.

45. J. Lockhart. Modern PHP: New Features and Good Practices / J. Lockhart, 2015 – с. 264.

46. P. Siriwardena. Microservices Security in Action / P. Siriwardena, 2020 – с. 420.

47. M. McDonald. Web Security for Developers: Real-World Misfortunes and How to Fix Them / M. McDonald, 2022 – с. 216.

48. C. Sharma. Securing Generative AI Applications: Prompt Injection, Data Leakage, and LLM Security / C. Sharma, 2024 – с. 300.

49. Техніка безпеки під час експлуатації обладнання [Електронний ресурс]. – Режим доступу до ресурсу: https://oppb.com.ua/news/tehnika-bezpeky-pid-chas-ekspluataciyi-obladnannya-v-remontnyh-maysternyah // Дата доступу до ресурсу: 20.05.2026р.

50. Закон України про захист людини від впливу іонізуючого випромінювання. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/15/98-%D0%B2%D1%80#Text // Дата доступу до ресурсу: 20.05.2026р.

51. Про захист людини від впливу іонізуючого випромінювання. Режим доступу до ресурсу: https://protocol.ua/ua/pro_zahist_lyudini_vid_vplivu_ionizuyuchogo_viprominyuvannya/ // Дата доступу до ресурсу: 20.05.2026р.

52. Створення безпечних і нешкідливих умов праці. Режим доступу до ресурсу: https://oppb.com.ua/news/stvorennya-bezpechnyh-i-neshkidlyvyh-umov-praci // Дата доступу до ресурсу: 20.05.2026р.

53. Koroliuk, R., Nykytyuk, V., Tymoshchuk, V., Soyka, V., Tymoshchuk, D.. Automated monitoring of bee colony movement in the hive during winter season. BAIT’2024: The 1st International Workshop on “Bioinformatics and applied

information technologies", October 02-04, 2024, Zboriv, Ukraine. CEUR Workshop Proceedings, 2024, 3842, pp. 147–156. ISSN: 1613-0073**.**

54. Oleh Yasniy, Iryna Didych, Dmytro Tymoshchuk, Iaroslav Pasternak, Vyacheslav Nykytyuk, Hryhorii Shymchuk, Dmytro Radyk. Fatigue crack growth prediction of automotive steels using ensemble-based machine learning methods. Procedia Structural Integrity, VIII International Conference "In-service Damage of Materials: Diagnostics and Prediction" Volume 81, (15 -17 October 2025.) 2026, P.116-122.

ДОДАТКИ

Тези першої конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Краківський економічний університет (Польща)
Вроцлавський економічний університет (Польща)
Університет «Опольська Політехніка» (Польща)
Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Вінницький національний аграрний університет
Львівський національний університет ім. І. Франка
Головне управління Пенсійного фонду в Тернопільській області
Наукове товариство ім. Шевченка
Тернопільський обласний комунальний інститут післядипломної педагогічної освіти
Сумський державний педагогічний університет
Запорізький національний університет

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів

11-12 грудня 2025 року

УКРАЇНА
ТЕРНОПІЛЬ – 2025

СЕКЦІЯ 5
КОМП'ЮТЕРНО-ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ЗВ'ЯЗКУ

УДК 004.4
Д.Т. Антонюк
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВПРОВАДЖЕННЯ LLM У ВЕБСЕРВІС ДЛЯ ВІДПОВІДЕЙ ЗА ДОПОМОГОЮ AZURE OPENAI

D.T. Antoniuk

IMPLEMENTATION OF LLM IN A WEB SERVICE FOR RESPONSES USING AZURE OPENAI

Сучасний етап розвитку цифрових технологій характеризується зростанням потреби у швидкій, персоналізованій та автоматизованій взаємодії між користувачем і програмними системами. Зі збільшенням кількості інформаційних запитів та ускладненням структури вебдодатків виникає необхідність у впровадженні інтелектуальних рішень, здатних аналізувати контекст звернень та забезпечувати оперативну генерацію релевантних відповідей. Інтеграція великих мовних моделей (LLM) у поєднанні з хмарними сервісами, такими як Azure OpenAI, відповідає цим вимогам, дозволяючи підвищити ефективність обробки запитів, зменшити навантаження на операторів підтримки та покращити взаємодію користувача з системою [1].

Типова архітектура інтеграції LLM у вебдодаток включає фронтенд-компонент, що забезпечує інтерфейс користувача та передачу запитів, бекенд, відповідальний за логіку обробки, і API-з'єднання з сервісом Azure OpenAI. Під час запиту користувача фронтенд формує повідомлення і передає його на сервер, де бекенд аналізує інформацію, формує промпт з урахуванням системних інструкцій і контексту, після чого здійснює API-виклик до моделі. Azure OpenAI генерує відповідь, яка повертається на бекенд, проходить можливу постобробку (виправлення помилок, форматування) і надсилається на фронтенд для відображення користувачеві. Такий підхід дозволяє реалізувати адаптивну взаємодію, забезпечує контроль над якістю відповідей та можливість масштабування рішення [2].

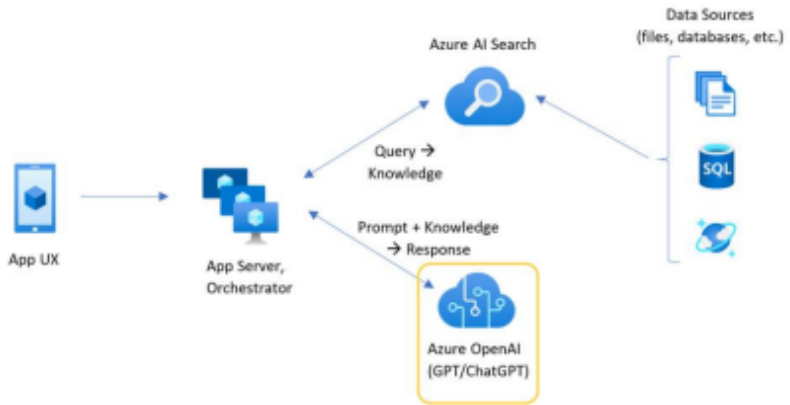

Рисунок 1. Схема інтеграції Azure Open AI

Інтеграція великих мовних моделей у вебдодатки забезпечує можливість автоматизованої генерації відповідей на основі аналізу контексту користувацьких запитів. LLM здатні виконувати семантичне розуміння звернень, генерувати адаптивні відповіді та працювати з неповними або неоднозначними запитами, що значно підвищує ефективність взаємодії з системою. Використання Azure OpenAI додатково забезпечує доступ до оптимізованих моделей у хмарній інфраструктурі Microsoft, з підтримкою масштабування, високою продуктивністю та вбудованими механізмами безпеки. Платформа дозволяє застосовувати як готові моделі (GPT-3.5, GPT-4 тощо), так і їх модифікації з урахуванням специфіки предметної області, що робить інтеграцію більш гнучкою та адаптованою до вимог вебдодатка [3].

Інтеграція LLM у вебдодатки за допомогою Azure OpenAI дозволяє суттєво оптимізувати бізнес-процеси шляхом автоматизації повторюваних звернень та скорочення часу реагування на запити користувачів. На відміну від традиційних чат-ботів із жорстко визначеною логікою, мовна модель здатна аналізувати контекст, адаптувати відповіді до конкретної ситуації та надавати більш точну та персоналізовану інформацію.

Під час інтеграції LLM необхідно врахувати низку технічних обмежень, зокрема продуктивність API, ліміти на кількість запитів, швидкодію та стійкість системи при великій кількості одночасних користувачів. Важливим аспектом є забезпечення конфіденційності та захисту переданих даних, особливо у разі використання персоналізованого контексту або доступу до внутрішніх джерел інформації. Рекомендується впроваджувати механізми контролю якості відповідей, фільтрації небажаного контенту та уникнення галюцинацій моделі. Особливу увагу варто приділити безпечній обробці даних через Azure (використання OAuth 2.0, рольова автентифікація, шифрування) та впровадженню систем моніторингу стабільності інтеграції [4].

Подальші дослідження можуть бути спрямовані на вдосконалення механізмів формування промптів із урахуванням попередньої історії взаємодії користувача, динамічного контексту та адаптивних стратегій генерації відповідей. Перспективним є впровадження гібридних моделей, що поєднують LLM із доменними знаннями або онтологіями для підвищення точності відповідей у спеціалізованих галузях [5].

Реалізація інтеграції LLM у вебдодатки за допомогою Azure OpenAI демонструватиме потенціал створення інтелектуальних систем взаємодії, здатних адаптуватися до потреб користувачів у реальному часі. Подібні рішення формуватимуть основу для нових підходів до автоматизації інформаційних процесів, підтримки комунікації та інтелектуального аналізу запитів, що відповідатиме напрямам розвитку сучасних цифрових технологій.

Література

1. Azure OpenAI in Foundry Models URL: https://azure.microsoft.com/en-us/products/ai-foundry/models/openai. Доступ до ресурсу: 30.11.2025
2. Azure OpenAI: The Smart Choice for AI-Powered Enterprise Solutions URL: https://www.cloudoptimo.com/blog/azure-openai-the-smart-choice-for-ai-powered-enterprise-solutions/ Доступ до ресурсу: 30.11.2025
3. Models Open AI URL: https://platform.openai.com/docs/models_ Доступ до ресурсу: 30.11.2025
4. К.М. Онищенко. Данієль Я.І., Каманєв Р.О. Аналіз методів обробки природної мови.
5. Albert Ziegler. Prompt Engineering for LLMs / Albert Ziegler, 2024 – с. 250.

Тези другої конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

17-18 грудня 2025 року

ТЕРНОПІЛЬ
2025

УДК 004.4

Д. Антонюк
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВПРОВАДЖЕННЯ LLM У ВЕБСЕРВІС ДЛЯ ВІДПОВІДЕЙ ЗА ДОПОМОГОЮ AZURE OPENAI

UDC 004.4

D. Antoniuk

IMPLEMENTATION OF LLM IN A WEB SERVICE FOR RESPONSES USING AZURE OPENAI

Забезпечення ефективної взаємодії користувачів із веб-сервісами потребує застосування інтелектуальних алгоритмів обробки природної мови. У веб-середовищі реалізується інтеграція **моделей великих мов (LLM)** через хмарні сервіси **Azure OpenAI**, що дозволяє автоматично генерувати контекстуально релевантні відповіді на запити користувачів [1].

Архітектура системи передбачає використання веб-фреймворків **Python (Flask або Django)** для організації обробки запитів та управління базою даних **PostgreSQL**, що забезпечує збереження історії взаємодій та структуроване управління даними. Основні механізми обробки інформації включають токенізацію, контекстний аналіз та семантичне зважування ключових елементів тексту, що підвищує точність та релевантність відповідей. Додатково аналіз контексту запитів дозволяє моделі враховувати попередні взаємодії користувача для більш персоналізованих відповідей. Використання сучасних методів обробки природної мови сприяє точнішому розпізнаванню сенсу та намірів у тексті [2]. Застосування семантичного зважування та кластеризації ключових елементів тексту підвищує релевантність відповіді у різних контекстах [3].

Інтеграція LLM дозволяє оптимізувати процес комунікації користувачів із сервісом, підвищити швидкість обробки запитів та ефективність інформаційної системи. Особливу увагу приділено забезпеченню **масштабованості, безпеки та конфіденційності даних**, а також адаптації алгоритмів до змінних навантажень у розподілених середовищах [4].

Використання хмарних платформ та інтелектуальних моделей відкриває можливості для аналітики поведінки користувачів, формалізації запитів і моделювання взаємодії в складних веб-системах. Це сприяє розвитку ефективних інформаційних технологій, здатних адаптуватися до потреб користувачів і забезпечувати високий рівень автоматизації у сучасних веб-сервісах [5].

Література

1. Azure OpenAI in Foundry Models [Електронний ресурс] : https://azure.microsoft.com/en-us/products/ai-foundry/models/openai. Доступ до ресурсу: 09.12.2025.
2. К. М. Онищенко. Аналіз методів обробки природної мови / К.М. Онищенко, Я.І. Данієль, Р.О. Каманєв.
3. Azure OpenAI: The Smart Choice for AI-Powered Enterprise Solutions [Електронний ресурс] : https://www.cloudoptimo.com/blog/azure-openai-the-smart-choice-for-ai-powered-enterprise-solutions/ Доступ до ресурсу: 09.12.2025
4. К.М. Онищенко. Аналіз методів обробки природної мови / К.М. Онищенко, Я.І. Данієль, Р.О. Каманєв.
5. Cathy Tanimura. SQL for Data Analysis. Advanced Techniques for Transforming Data into Insights. 1st Ed. / Cathy Tanimura, 2021 – с. 350.