ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Автоматизована система генерації текстів на основі мовних моделей OpenAI»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

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

_____ Анастасія КАРАПИШ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконала: Анастасія КАРАПИШ
здобувачка вищої освіти гр.ШІД-41

Керівник: Тетяна КИСІЛЬ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ

Завідувач кафедрою Штучного інтелекту
_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Карапиш Анастасія Олександровна
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Автоматизована система генерації текстів на основі мовних моделей OpenAI»

керівник кваліфікаційної роботи: Тетяна КИСІЛЬ старший викладач
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1.Огляд сучасних технологій для розробки онлайн-консультантів

2. Вибір технологій для реалізації онлайн-консультанта: HTML, CSS, JavaScript, PHP, SQL.

3. Архітектура онлайн-консультанта: клієнтська та серверна частини.

4. Інтеграція з базою даних та зовнішніми сервісам.

5. Забезпечення безпеки та масштабованості системи.

5. Перелік графічного матеріалу: *презентація*

6. Дата видачі завдання «27» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір та аналіз літератури за темою (мовні моделі, OpenAI API, NLP, генерація тексту, архітектура веб-додатків)	27.02-05.11.25	
2	Детальне вивчення теорії мовних моделей OpenAI (архітектура, принципи роботи, можливості та обмеження).	05.11-12.11.25	
3	Обґрунтування вибору конкретної мовної моделі OpenAI для реалізації системи.	13.11-19.11.25	
4	Розробка архітектури системи (компоненти, їх взаємодія, API-інтеграції).	20.11-25.11.25	
5	Проектування структури бази даних (зберігання користувачів, текстів, запитів).	27.11-03.12.25	
6	Тестування модулів веб-ресурсу та генерації сценаріїв.	04.12-10.12.25	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.25	
8	Розробка демонстраційних матеріалів	21.12-31.05.25	

Здобувачка вищої освіти _____ Анастасія КАРАПИШ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ Тетяна КИСІЛЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 83 стор., 12 рис., джерел 21.

Мета роботи: Розробка програмного забезпечення для автоматичної генерації текстів за допомогою мовних моделей OpenAI, орієнтуючись на точність і природність згенерованих матеріалів.

Об'єкт дослідження: Процес генерації тексту за допомогою технологій штучного інтелекту та мовних моделей, таких як GPT, BERT.

Предмет дослідження: Інтелектуальна система, що використовує технології обробки природної мови (NLP) і нейронні мережі для автоматичної генерації текстів в різних контекстах (новини, блоги, доповіді).

Короткий зміст роботи: У роботі проведено огляд сучасних підходів до автоматичної генерації тексту з використанням мовних моделей, особливо моделей GPT. Зокрема, розглянуті методи генерації тексту на основі трансформерів і їх застосування у реальних задачах. Обґрунтовано вибір моделі GPT-4 для створення текстів на основі заданого контексту. Проаналізовано переваги та недоліки існуючих підходів до обробки тексту та детально розглянуто інтеграцію OpenAI API для отримання результатів. У роботі побудована система на базі Python, реалізовано веб-інтерфейс для тестування результатів генерації текстів. Також було проведено тестування згенерованих текстів за допомогою метрик точності, логічності та природності. На основі результатів підтверджено ефективність запропонованої системи для створення текстів, що відповідають заданим вимогам.

КЛЮЧОВІ СЛОВА: ГЕНЕРАЦІЯ ТЕКСТУ, OPENAI, GPT, ТРАНСФОРМЕРИ, PYTHON, API, МАШИННЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

ЗМІСТ

ВСТУП 8
1 ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАЦІЇ ТЕКСТІВ ТА ВЕБ-СПІЛЬНОТ З АІ 12
1.1 Огляд сучасних підходів до генерації текстів 12
1.2 Аналіз мовних моделей OpenAI 16
1.3 Веб-спільноти з використання штучного інтелекту та їх роль в косметології 21
1.4 Аналіз існуючих рішень та аналогів 30
2 ПРОЕКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ГЕНЕРАЦІЇ ТЕКСТІВ ТА ВЕБ-СПІЛЬНОТИ 35
2.1 Формулювання вимог до проектованої системи 35
2.2 Архітектура проектованої системи 40
2.3 Проектування бази даних проектованої системи 46
2.4 Розробка інтерфейсу користувача (Frontend) 49
2.5 Розробка серверної частини (Backend) 55
3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТОВАНОЇ СИСТЕМИ 59
3.1 Середовище розробки та його розгортання 59
3.2 Детальний опис реалізованих функціональних модулів 64
3.3 Тестування проектованого веб-ресурсу 70
3.4 Аналіз тестових результатів проектованої системи 76
ВИСНОВКИ 12
ПЕРЕЛІК ПОСИЛАНЬ 81
ДОДАТКИ 83
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 83

ВСТУП

Сучасний світ стрімко трансформується під впливом технологій штучного інтелекту (ШІ), які активно проникають у найрізноманітніші аспекти життєдіяльності людства — від медичних діагностичних систем і автоматизованих освітніх платформ до правових аналізаторів і креативних інструментів у мистецтві. Зокрема, особливу увагу дослідників і розробників привертає автоматична генерація текстів, яка не лише змінює парадигму створення контенту, а й формує нову культуру взаємодії між людиною і машиною. Такий підхід дозволяє підвищити ефективність у веденні професійної документації, поліпшити якість комунікацій, автоматизувати виробництво навчальних і маркетингових матеріалів, а також підтримати персоналізовану взаємодію у цифрових сервісах. Останніми роками особливу популярність здобули мовні моделі, засновані на трансформерній архітектурі, зокрема GPT (Generative Pre-trained Transformer) від компанії OpenAI. Їх здатність створювати осмислені, логічно пов’язані тексти, адаптовані під конкретний запит користувача, наблизила машинне мовлення до людського рівня. Технології, що ще вчора здавались фантастикою, сьогодні вже доступні через API й інтегруються в різноманітні цифрові рішення.

Актуальність теми зумовлена зростаючою потребою в інструментах для швидкої, якісної та контекстуально релевантної генерації текстів у бізнесі, освіті, журналістиці та інших галузях. Існує значний попит на системи, здатні генерувати тексти автоматично, з урахуванням стилістики, змістовності та мети використання — від створення описів товарів до написання доповідей, звітів, художніх творів чи навчальних матеріалів.

Метою даної кваліфікаційної роботи є розробка автоматизованої системи генерації текстів на основі мовних моделей OpenAI, що забезпечує інтеграцію з API, зручний веб-інтерфейс та високу якість результату.

Об’єктом дослідження є процес генерації текстів за допомогою нейронних мовних моделей.

Предметом дослідження виступає інтелектуальна система, що використовує OpenAI API для генерації текстів заданої тематики.

Для досягнення поставленої мети були сформульовані такі завдання:

- проаналізувати сучасні підходи та інструменти генерації тексту;
- дослідити архітектуру та можливості моделей GPT;
- спроєктувати архітектуру автоматизованої системи генерації текстів;
- реалізувати взаємодію з OpenAI API;
- створити веб-інтерфейс для взаємодії з користувачем;
- протестувати результати генерації за метриками якості.

Методами дослідження стали: аналіз наукових публікацій та огляд літературних джерел, присвячених тематиці генерації тексту та застосуванню великих мовних моделей; вивчення сучасних технологій обробки природної мови та архітектури трансформерів; побудова структурної моделі програмного забезпечення; моделювання алгоритмів взаємодії з OpenAI API; проєктування та реалізація клієнт-серверної архітектури із застосуванням фреймворку Flask; створення зручного веб-інтерфейсу для користувача; тестування ефективності генерації текстів з використанням метрик якості, таких як змістовна релевантність, граматична коректність, семантична повнота й читабельність згенерованих результатів.

Наукова новизна полягає в поєднанні можливостей OpenAI GPT із практичними задачами автоматизованого створення текстів з урахуванням вимог змістовності, логічності та стилістичної відповідності. Практичне значення полягає у створенні працездатного веб-сервісу, який може бути використаний для автоматичної генерації текстового контенту в реальних умовах.

1 ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАЦІЇ ТЕКСТІВ ТА ВЕБ-СПІЛЬНОТ З AI

1.1 Огляд сучасних підходів до генерації текстів

Процеси автоматичної генерації тексту є однією з ключових підгалузей обробки природної мови (NLP), яка стала надзвичайно актуальною у зв’язку з вибуховим зростанням обсягів цифрового контенту, що створюється, аналізується та транслюється через мережу Інтернет. Генерація текстів охоплює широкий спектр завдань: автоматичне написання новин, створення маркетингових описів, сценаріїв для чат-ботів, адаптивних відповідей на звернення користувачів, персоналізованих рекомендацій та навіть художніх творів.

На ранніх етапах розвитку технології текстогенерації мали переважно шаблонний характер, однак з часом почали впроваджуватись статистичні моделі, які базувались на ймовірнісному підборі слів. Проте лише з приходом глибокого навчання та нейронних мереж стало можливим моделювати зв’язні, логічно структуровані та контекстно релевантні тексти. Такі системи вже не просто формують речення — вони навчаються мовним закономірностям, стилістичним особливостям та логічним зв’язкам на рівні цілих абзаців і навіть документів. Сучасна епоха у генерації текстів нерозривно пов’язана з трансформерною архітектурою та великими мовними моделями (LLMs), які продемонстрували безпрецедентні результати у створенні текстів, що іноді не поступаються людському письму. Вони вже зараз змінюють підхід до створення контенту в бізнесі, освіті, журналістиці, наукових дослідженнях і розробці програмного забезпечення, відкриваючи нові горизонти ефективності та масштабування людської праці..

1.1.1 Історичний розвиток та класифікація методів генерації текстів

Історично перші спроби автоматизації створення текстів базувалися на шаблонних методах. Наприклад, система ELIZA (1966) використовувала жорстко запрограмовані правила для генерації фраз у стилі психотерапевта, не маючи жодного розуміння змісту повідомлень.

У 80–90-х роках з розвитком статистичних методів з’явилися підходи, що базувались на n-грамових моделях. Ці моделі прогнозували наступне слово на основі ймовірності послідовності попередніх слів, однак обмежувались вузьким контекстом і не враховували глибоких семантичних зв’язків. Далі були запроваджені методи на основі машинного навчання, які дозволяли навчати системи на корпусах текстів, формуючи статистичні закономірності без необхідності ручного кодування правил. Але справжній прорив стався з приходом глибинного навчання — нейронні мережі відкрили нові можливості для генерації текстів.

Методи класифікують за такими групами:

- Шаблонні (template-based): генерують текст за фіксованими структурами;
- Статистичні (n-gram models): базуються на ймовірностях послідовностей;
- Нейронні (RNN, LSTM, Transformer): здатні моделювати довгі залежності у тексті;
- Гібридні: поєднують кілька підходів (наприклад, GPT з фільтрами або вбудованими логіками).

1.1.2 Нейронні мережі в генерації текстів: від RNN до Трансформерів

Нейронні мережі почали активно використовуватись у генерації текстів у 2010-х роках. Найпершими були рекурентні нейронні мережі (RNN), які дозволяли працювати з послідовностями даних. RNN зберігали стан попереднього входу, що дало змогу враховувати контекст. Проте RNN мали суттєві недоліки: вони не могли ефективно запам’ятовувати довгі залежності через проблему зникнення/вибуху градієнтів. Для вирішення цих проблем були запропоновані архітектури LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit), які забезпечували кращу здатність до збереження контексту на довгих відстанях.

У 2017 році була представлена архітектура Transformer у статті «Attention is All You Need» (Vaswani et al.), яка радикально змінила підхід до обробки послідовностей. Замість послідовного опрацювання, як у RNN, Transformer використовує механізм самоуваги (self-attention), що дозволяє одночасно аналізувати всі елементи вхідної послідовності. Це дало змогу істотно покращити

паралельність обчислень, швидкість навчання моделей і якість згенерованого тексту. На основі цієї архітектури були створені такі моделі, як BERT, GPT, T5, які сьогодні є стандартом де-факто в NLP.

1.1.3 Принципи роботи великих мовних моделей (LLMs)

Великі мовні моделі (Large Language Models, LLMs) — це глибокі нейронні мережі, навчання яких здійснюється на великих корпусах текстів із мільярдами параметрів. Прикладом є GPT-3/4, PaLM, Claude тощо.

Основні принципи роботи LLMs:

Переднавчання (pre-training): модель навчається передбачати наступне слово в тексті, обробляючи мільярди прикладів. Це забезпечує узагальнене мовне представлення.

Донавчання (fine-tuning): після переднавчання модель може бути адаптована до конкретних задач або стилістик (наприклад, діалогові системи, юридичні документи).

Підказки (prompting): користувач взаємодіє з LLM через текстовий запит (prompt), який визначає стиль і зміст відповіді.

Контекстне навчання (in-context learning): модель не потребує перенавчання — вона “вивчає” приклади з тексту prompt’а в реальному часі.

Сучасні LLM, зокрема GPT-4, є надзвичайно потужними інструментами для генерації текстів (рис. 1.1), які здатні імітувати людське мовлення з високим рівнем логічної зв’язності, стилістичної відповідності та семантичної точності. Вони підтримують багатомовність, що дозволяє працювати з текстами на десятках мов, забезпечують можливість виконання таких завдань, як машинний переклад, створення резюме документів, написання програмного коду, ведення діалогів, генерація запитань, побудова відповідей на складні запити та навіть креативне письмо. Моделі здатні до так званого "zero-shot" або "few-shot" навчання, що дозволяє вирішувати нові завдання без потреби в додатковому донавчанні. Вони опановують структури та шаблони мови, розуміють контекст, імітують аргументацію та дедуктивне мислення, що відкриває перед ними перспективи у сфері автоматизації юридичних, освітніх, наукових та медичних процесів.

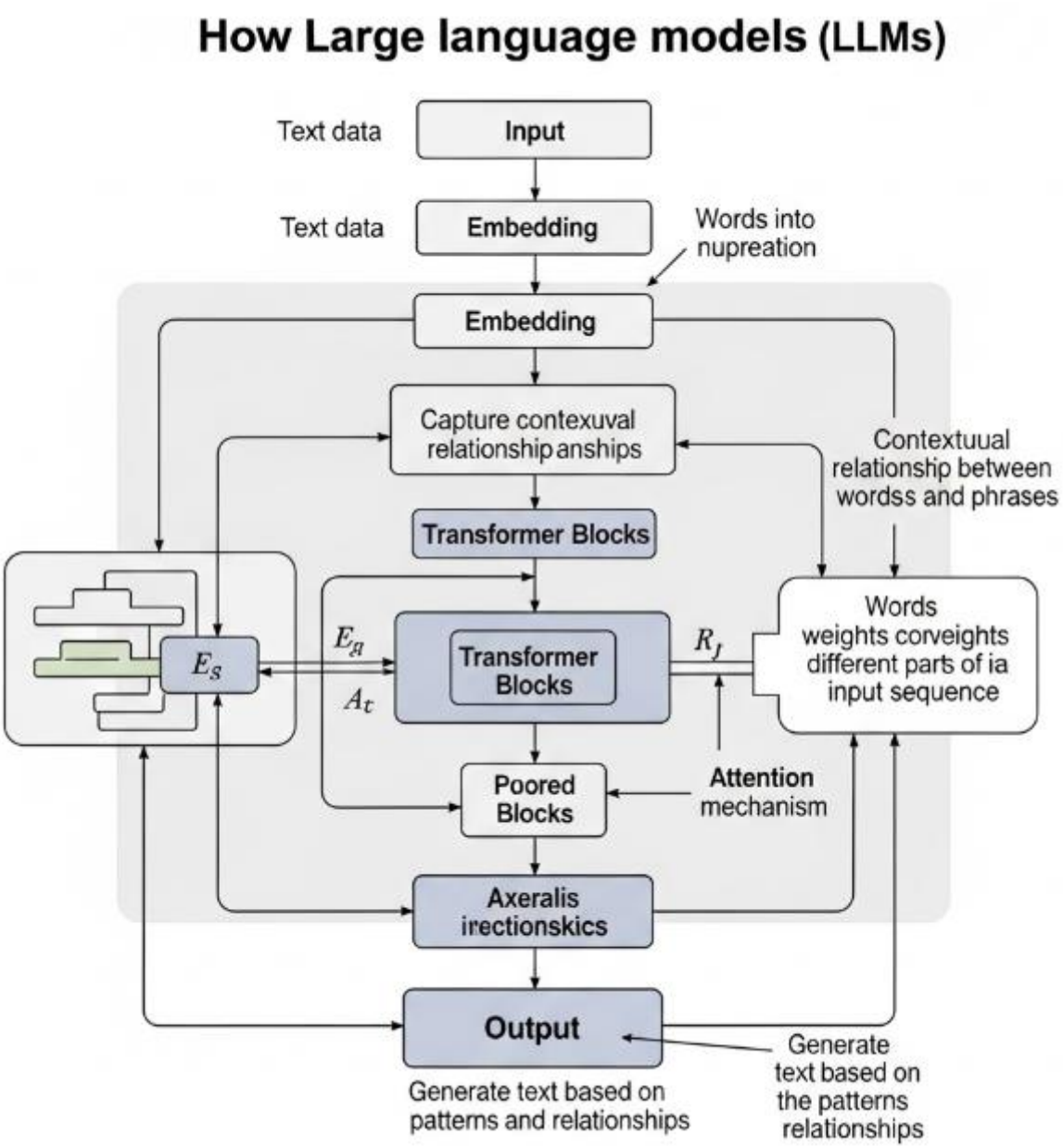


Рисунок 1.1 Структурна схема принципів роботи великих мовних моделей

Попри переваги, великі мовні моделі мають низку важливих обмежень. По-перше, вони можуть генерувати правдоподібні, але вигадані твердження — так звані "галюцинації", які особливо небезпечні у критичних галузях. По-друге, для їх роботи потрібні значні обчислювальні ресурси: використання моделей на кшталт GPT-4 вимагає потужних серверів з графічними процесорами або спеціалізованих AI-чипів, що підвищує вартість експлуатації. По-третє, існує проблема етичної відповідальності — моделі можуть ненавмисно відтворювати упередження, стереотипи або некоректний контент, якщо вони були присутні у навчальних даних. Саме тому важливо забезпечувати фільтрацію виводів, аудит результатів і

контроль над сценаріями застосування таких систем у суспільно важливих контекстах.

Хоча LLM відкривають нові горизонти в автоматизації обробки інформації, їх використання повинно супроводжуватись усвідомленням ризиків і відповідальним підходом до їх проєктування та інтеграції в інформаційні системи.

1.2 Аналіз мовних моделей OpenAI

Мовні моделі OpenAI, зокрема серія GPT (Generative Pre-trained Transformer), є прикладом найуспішнішого застосування трансформерної архітектури у сфері генерації природної мови. Їхня поява ознаменувала нову епоху у розвитку технологій штучного інтелекту, оскільки вони поєднують у собі виняткову здатність до контекстного аналізу, генерації зв'язного тексту та адаптації до різноманітних мовних задач. Моделі GPT навчаються на багатомовних корпусах даних, охоплюючи наукові статті, новини, енциклопедії, форуми, коди програм, що дозволяє їм вільно оперувати знаннями у різних галузях. Ці моделі виявилися настільки універсальними, що стали основою для реалізації цілого спектру сервісів: автоматичних асистентів, інтелектуальних чат-ботів, генераторів маркетингових текстів, систем підтримки клієнтів, інструментів для перекладу, створення коду, написання сценаріїв, рецензій, есе та навіть віршів. Їхні можливості охоплюють як аналітичні функції (наприклад, узагальнення великих текстів, виділення ключових ідей), так і креативні (генерація художніх текстів, ідей для стартапів, слоганів). GPT-моделі здатні підтримувати контекст тривалого діалогу, змінювати стиль відповіді залежно від ситуації, симулювати роль експерта у певній сфері. Наприклад, ChatGPT може виступати як вчитель, юрист, програміст або психолог, імітуючи відповідний стиль мовлення та рівень знань. Це стало можливим завдяки масштабному попередньому навчанню та здатності до in-context learning — тобто розуміння задачі прямо з тексту запиту без додаткового навчання.

Високий рівень генерації тексту, зокрема у GPT-4, дозволяє системам демонструвати не лише мовну грамотність, а й логіку, структурованість, аргументованість, що особливо важливо для задач, де йдеться про створення

пояснень, інструкцій, юридичних висновків, наукових описів. Багато компаній інтегрували API OpenAI у свої сервіси для автоматизації частини інтелектуальної праці, зниження витрат на контент, покращення обслуговування клієнтів та персоналізації взаємодії.

1.2.1 Архітектура та функціональні можливості моделей GPT

Моделі GPT побудовані на основі трансформерної архітектури, запропонованої в роботі "Attention is All You Need" у 2017 році. Ця архітектура докорінно змінила підходи до обробки послідовностей даних, оскільки дозволила відмовитись від рекурентних зв'язків, які були властиві попереднім моделям (RNN, LSTM). Основна інновація трансформера — механізм самоуваги (self-attention), що дає змогу аналізувати залежності між усіма токенами в межах одного блоку обробки одночасно. У випадку моделей GPT використовується модифікована версія трансформера з однонапрямною (каузальною) увагою — causal attention. Це означає, що під час генерації кожне слово (токен) може враховувати лише попередні елементи послідовності, що зберігає причинно-наслідковий зв'язок у побудові тексту. Завдяки цьому GPT моделі ідеально підходять для задач автогенерації — створення тексту на основі заданого контексту.

Модель отримує вхідні токени (наприклад, слова або частини слів), кодує їх у векторне представлення за допомогою ембеддингів, а потім у багатьох шарах самоуваги здійснює трансформацію цих представлень, що дозволяє враховувати як лексичні, так і синтаксичні зв'язки. На виході формується розподіл імовірностей наступного токена, з якого модель вибирає найбільш ймовірний варіант — або випадково (temperature sampling), або за принципом максимізації правдоподібності (greedy decoding).

Однією з переваг архітектури GPT є можливість обробки довгих контекстів (у GPT-4 — до 32 000 токенів і більше), що дозволяє зберігати цілісність змісту навіть у великих текстах. Крім того, трансформери добре паралеляться, що значно пришвидшує навчання і генерацію порівняно з рекурентними мережами. Завдяки цим технічним особливостям GPT здатна створювати граматично правильні,

логічно зв’язні, стилістично витримані тексти з високою релевантністю до запиту користувача (рис. 1.2).

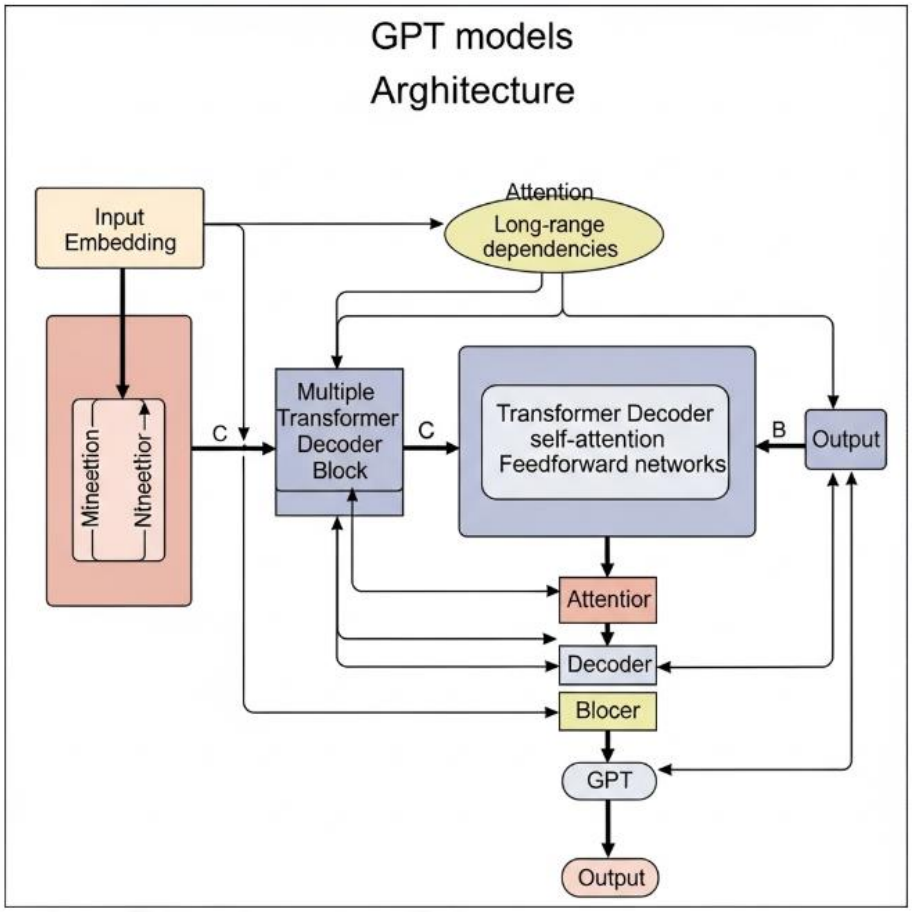


Рисунок 1.2 Архітектура та функціональні можливості моделей GPT

GPT навчаються в два етапи:

1. Pre-training: модель обробляє величезні обсяги тексту (інтернет-корпуси, книги, енциклопедії) для формування узагальненого мовного представлення.

2. Fine-tuning або Reinforcement Learning with Human Feedback (RLHF): модель адаптується до конкретних сценаріїв, де важлива якість, етика і відповідність відповідей.

Функціональні можливості GPT охоплюють:

- Генерацію тексту за заданим початком (prompt);
- Підтримку багатомовності;
- Ведення діалогів у чат-режимі (ChatGPT);
- Автоматичне резюмування великих документів;
- Переклад між мовами;

- Генерацію програмного коду (у спеціалізованих версіях, як-от Codex);
- Пояснення термінів, аргументацію, рецензування текстів;
- Здатність до "інтуїтивного" розуміння завдання за контекстом без перенавчання (in-context learning).

Варто відзначити, що GPT-4 є багатомодальною моделлю, яка, на відміну від своїх попередників, здатна обробляти не лише текстову, але й візуальну інформацію — зображення, схеми, графіки. Це відкриває нові можливості для створення систем, які поєднують текстовий аналіз з візуальним розпізнаванням. Наприклад, GPT-4 може пояснити зображення діаграми, надати опис фото, виявити текст на зображенні або поєднати вхідну графічну та текстову інформацію для складніших сценаріїв (наприклад, створення мультимодальних чат-асистентів). GPT-4 демонструє суттєве покращення в таких аспектах, як точність відповідей, здатність до тривалого утримання контексту, логічність формулювання думок та стилістична узгодженість текстів. Вона краще розпізнає неоднозначні формулювання, вміє будувати умовиводи на основі наданого контексту, а також генерує менш упереджений і більш зважений контент.

Модель здатна ефективно обробляти великі обсяги інформації — зокрема, працювати з контекстами до 32 000 токенів і більше, що дозволяє їй взаємодіяти з великими документами, звітами або навіть повноцінними книгами. Завдяки цьому GPT-4 підходить для задач, де потрібне глибоке опрацювання змісту, збереження структури і логіки викладення, включаючи створення технічної документації, юридичних аналізів, навчальних матеріалів, статей тощо.

GPT-4 поєднує потужність попередніх поколінь GPT з новим рівнем мультимодальної інтеграції, що дозволяє суттєво розширити сфери застосування штучного інтелекту у реальних практичних контекстах.

1.2.2 Застосування та обмеження моделей OpenAI API

API від OpenAI надає можливість інтегрувати мовні моделі GPT у будь-які програмні рішення за допомогою стандартних HTTP-запитів, що робить його доступним для розробників з різним рівнем технічної підготовки. Основний принцип взаємодії полягає у надсиланні текстового запиту (prompt) на сервер

OpenAI, який повертає згенеровану модельню відповідь. Це дозволяє розробникам швидко вбудовувати штучний інтелект у свої додатки, сайти або інформаційні системи без необхідності глибокого знання внутрішньої архітектури LLM або витрат на обчислювальну інфраструктуру. Практичне використання API охоплює безліч галузей. У сфері обслуговування клієнтів GPT дозволяє створювати віртуальних асистентів, які відповідають на запитання користувачів у режимі реального часу. В освітніх додатках — формувати автоматизовані пояснення, приклади, адаптивні тести або супровід навчального матеріалу. У творчих індустріях API активно використовується для генерації маркетингових текстів, слоганів, сценаріїв, креативних описів продуктів або навіть сюжетів до ігор та фільмів. У юридичній практиці моделі GPT допомагають створювати шаблони договорів, резюмувати нормативні акти або здійснювати первинний аналіз справ. Завдяки масштабованій архітектурі API, компанії можуть адаптувати його до конкретних потреб, налаштовуючи параметри довжини відповіді, стилістичної варіативності (temperature), кількості варіантів (top_p), частотного та наявного пенальті (frequency & presence penalty). Це дає змогу тонко керувати поведінкою моделі та адаптувати її під вимоги бізнесу або сфери застосування.

OpenAI постійно розширює можливості API, впроваджуючи функції кодування, зображень, аудіоаналізу, що робить платформу GPT API не лише інструментом текстогенерації, а справжнім багатоцільовим інтерфейсом до великомасштабного штучного інтелекту.

Серед переваг API:

- Простота інтеграції (стандартний REST API);
- Гнучкість налаштувань: довжина відповіді, стиль, рівень творчості (temperature);
- Висока якість тексту та швидкість генерації;
- Доступність через хмарну інфраструктуру.

Однак API має й обмеження:

- Обмеження на кількість запитів та платний доступ — чим більший обсяг, тим вища вартість;

- Нестабільність у випадку складних завдань — модель може "галюцинувати", тобто вигадувати факти;
- Відсутність знань про події після певної дати навчання (наприклад, GPT-4 не має знань про події після жовтня 2023);
- Вимоги до захисту даних — необхідно враховувати передачу конфіденційної інформації через зовнішні сервери.

OpenAI API забезпечує надзвичайно широкі можливості для використання LLM у прикладних задачах та дозволяє інтегрувати інтелектуальні функції у звичайні вебсервіси без необхідності запуску моделі локально..

1.3 Веб-спільноти з використання штучного інтелекту та їх роль в косметології

У XXI столітті веб-спільноти стали невід’ємною частиною цифрової екосистеми. Вони виконують функцію не тільки обміну інформацією, а й побудови соціального капіталу, формування експертних груп, створення продуктів та послуг на основі краудсорсингу. З розвитком веб-технологій та глобальним поширенням інтернету такі спільноти перетворились із простих форумів на багатофункціональні інтерактивні платформи, які впливають на рішення бізнесу, політики, освіти, науки, медицини тощо.

Інформаційний простір сьогодення визначається не лише кількістю доступного контенту, а й здатністю користувачів активно брати участь у його створенні, поширенні та вдосконаленні. Сучасні користувачі не є пасивними споживачами інформації — вони оцінюють, коментують, створюють нові смисли, формують спільноти за інтересами, будують мережі довіри та знань. Цей процес відбувається в умовах постійної еволюції цифрових технологій, які забезпечують інтерактивність, персоналізацію, візуалізацію і миттєву зворотну реакцію. У цьому контексті веб-спільноти постають як складні соціотехнічні системи, де поєднуються технічні інфраструктури (сервери, інтерфейси, платформи) та соціальні механізми (нормативи, модерація, репутація, співпраця). Вони сприяють колективному розумовому процесу — від вирішення повсякденних проблем до

обговорення глобальних тем, від локального обміну досвідом до міжнародних дискусій. Особливу актуальність така форма взаємодії здобула в період пандемії COVID-19, коли більшість комунікацій перемістилися в онлайн-середовище. Веб-спільноти стали не лише майданчиком для обговорень, а й засобом емоційної підтримки, джерелом оперативної інформації, інструментом самонавчання та соціалізації. На тлі глобальної цифровізації та зростання інформаційної мобільності роль таких платформ продовжує стрімко зростати. Вони змінюють моделі комунікації, впливають на формування суспільної думки, надають нові можливості для активної участі громадян у суспільних процесах. Веб-спільноти стають віртуальними аналогами традиційних соціальних інститутів, зокрема освітніх, наукових, професійних і навіть політичних. У найближчому майбутньому вони можуть перетворитися на ключові вузли інформаційної екосистеми людства, де знання буде створюватися колективно, в режимі реального часу, з використанням інструментів штучного інтелекту, автоматизованого аналізу та інтерактивного зворотного зв’язку.

В сучасній косметології штучний інтелект активно використовується для персоналізації догляду за шкірою, діагностики, підбору продуктів та віртуальних примірок. Ось декілька прикладів компаній та технологій, які пропонують такі рішення, і де їх можна знайти:

Платформи та компанії, що спеціалізуються на AI для краси:

1. **Revieve:** Це одна з провідних платформ, яка пропонує AI-рішення та AR (доповнену реальність) для брендів та ритейлерів у сфері краси. Вони мають модулі для:

- **AI Skincare Advisor:** Аналізує стан шкіри за допомогою фотографій і надає персоналізовані рекомендації щодо продуктів.
- **AI Makeup Advisor / AI Makeup Artist:** Допомагає віртуально приміряти макіяж та підібрати продукти.
- **AI Haircare Advisor / AI Hair Color Artist:** Аналогічні рішення для догляду за волоссям та вибору кольору.
- Технологія використовується багатьма відомими брендами (рис. 1.3)

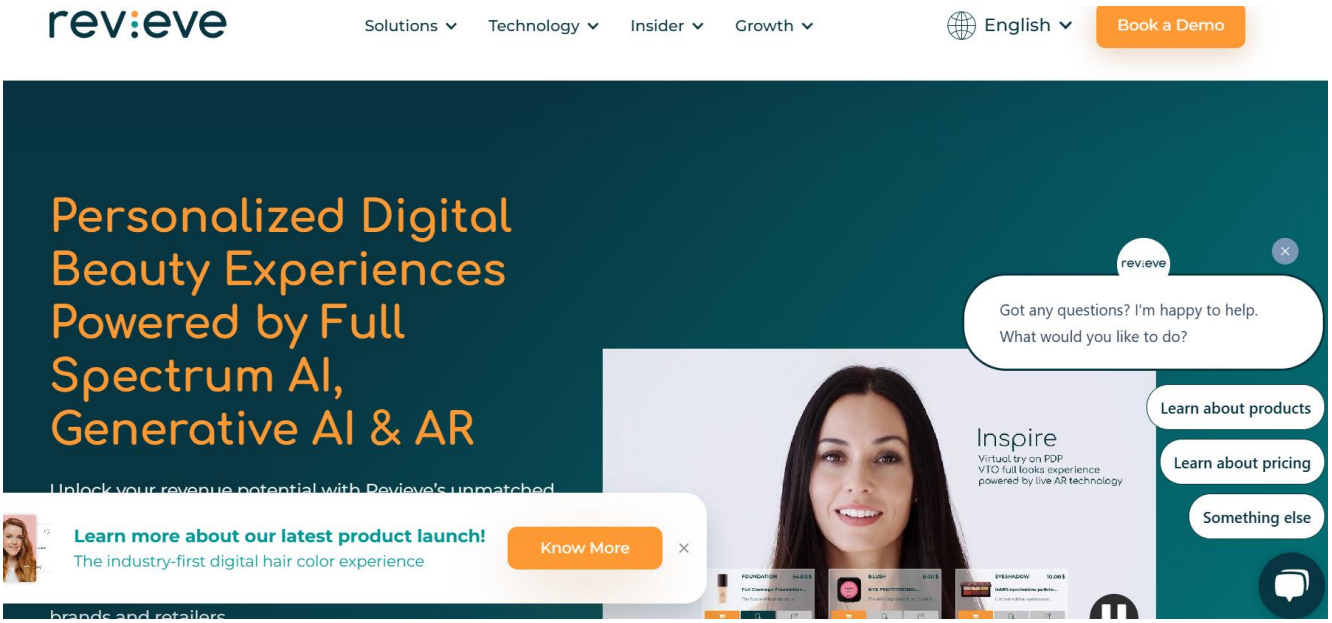


Рисунок 1.3 Зовнішній вигляд веб-ресурсу Revieve

2. **Thea Care:** Пропонує B2B рішення для брендів з AI-аналізом шкіри за допомогою смартфона. Їхня AI-технологія аналізує ознаки почервоніння, запалення, гіперпігментації та пор. Це допомагає брендам пропонувати персоналізовані рішення.

3. **Haut.AI:** Ця компанія пропонує SaaS-рішення для аналізу шкіри на основі AI, яке можна інтегрувати в сайти beauty-брендів. Їхня платформа аналізує понад 150 унікальних біомаркерів обличчя та використовує 94 алгоритми для рекомендації продуктів.

4. **Perfect Corp. (YouCam Makeup, YouCam Perfect):** Хоча це більше додатки, Perfect Corp. є лідером у сфері AR та AI для б'юті-індустрії. Вони співпрацюють з багатьма брендами, надаючи віртуальні примірки макіяжу, діагностику шкіри за допомогою AI. Деякі бренди використовують їхні технології на своїх сайтах для віртуального тестування продуктів. Вони також згадують SENSAI (Skin Vision AI) та Jane Iredale (AI-powered foundation matching) як приклади брендів, що використовують AI.

Українські приклади або згадки про застосування AI:

1. **ALL FACE:** Пропонують діагностику шкіри зі штучним інтелектом у своїх просторах у Києві та Одесі для глибокого аналізу проблем та підбору догляду.

2. **INSTYTUTUM:** Згадують про використання AI для діагностики шкіри обличчя.

3. Деякі українські статті та блоги (наприклад, від Inweb, KITAPP, Cosmedmedia) згадують про загальні тренди використання AI у б'юті-сфері, включаючи чат-боти, віртуальних асистентів, аналіз даних для маркетингу та діагностики.

Загальні напрямки застосування AI в косметології:

1. **Діагностика шкіри:** AI-системи аналізують зображення шкіри (селфі, відео) для виявлення проблем (акне, пігментація, зморшки, почервоніння, пори) та оцінки її стану.

2. **Персоналізовані рекомендації продуктів:** На основі діагностики AI може рекомендувати конкретні продукти та рутини догляду, адаптовані до індивідуальних потреб.

3. **Віртуальна примірка (AR/VR):** Дозволяє "приміряти" макіяж, колір волосся або навіть зміну форми обличчя в реальному часі за допомогою смартфона або комп'ютера.

4. **Чат-боти та віртуальні асистенти:** Надають консультації, відповідають на запитання про продукти, допомагають з вибором.

5. **Розробка нових продуктів:** AI допомагає аналізувати великі обсяги даних про інгредієнти, їх ефективність та взаємодію, що прискорює розробку інноваційних косметичних засобів.

6. **Оптимізація виробництва та ланцюга поставок:** AI може прогнозувати попит, оптимізувати запаси та логістику.

Штучний інтелект продовжує активно інтегруватися в індустрію краси, роблячи її більш персоналізованою, ефективною та доступною для споживачів. Наведемо узагальнену таблиця переваг та недоліків сайтів/платформ з AI у сфері косметології (табл. 1.1).

AI-сайти в косметології пропонують низку значних переваг. Вони дозволяють персоналізувати догляд за шкірою, забезпечуючи індивідуальні рекомендації для кожного користувача. Завдяки віртуальній примірці продуктів

можна спробувати косметику онлайн перед покупкою. Ці платформи також здатні швидко аналізувати стан шкіри, що допомагає вчасно виявити потреби та проблеми. Зрештою, такі рішення забезпечують значну зручність як для кінцевих користувачів, так і для косметичних брендів.

Таблиця 1.1 Аналіз існуючих косметологічних платформ з використанням штучного інтелекту

Сайт / Платформа	**Переваги**	**Недоліки**
Revieve	– Високоточний аналіз шкіри, волосся, макіяжу – Модулі персоналізованих порад – Інтеграція з брендами – AR і AI в одному рішенні	– В основному орієнтований на B2B, не напряму для користувачів – Вартість інтеграції може бути високою
Thea Care	– Аналіз проблем шкіри (почервоніння, пігментація тощо) через смартфон – Проста інтеграція в бізнес-процеси брендів	– Вузька спеціалізація (тільки шкіра) – Обмежена публічна доступність для споживача
Haut.AI	– Потужна платформа з 150+ біомаркерів – SaaS-рішення легко масштабувати – Працює з багатьма брендами	– Складна інтеграція для малих компаній – Немає відкритого демо для індивідуального користувача
Perfect Corp.	– Потужні AR-примірки – Підтримка багатьох брендів – Можливість тестування макіяжу, догляду, кольору волосся – Мобільні додатки	– Не всі функції доступні через сайт – Якість примірки залежить від пристрою користувача
ALL FACE (Україна)	– Локальна діагностика шкіри з AI – Фізичне консультування у Києві та Одесі	– Обмежений географічний доступ – Менш автоматизована взаємодія
INSTYTUTUM (Україна)	– Застосування AI в діагностиці шкіри – Презентація технологічності бренду	– Відсутність детального опису/демо AI-рішень на сайті

Однак існують і певні недоліки. Доступ до AI-рішень у косметології часто обмежений для індивідуальних користувачів, оскільки вони здебільшого

орієнтовані на B2B-сегмент. Для малих брендів впровадження таких технологій може бути доволі витратним. Крім того, якість аналізу сильно залежить від хорошої якості фото- та відеоматеріалів, наданих користувачем. Також існує проблема обмеженої прозорості алгоритмів штучного інтелекту, що може викликати питання щодо їхньої об'єктивності та надійності.

Враховуючи ці аспекти, важливо ретельно оцінювати ринок AI-рішень у сфері краси, щоб знайти оптимальні варіанти як для бізнесу, так і для кінцевого користувача.

1.3.1 Типи та структура веб-спільнот

Веб-спільноти можуть мати різну спрямованість і функціональність залежно від своєї мети, формату та організаційної структури. Деякі орієнтовані на професійний розвиток і обмін знаннями у певній галузі, інші — на розважальний контент, соціальні зв’язки чи навчальні ініціативи. Існують також технічні або краудсорсингові спільноти, де учасники колективно створюють або вдосконалюють продукти — програмне забезпечення, енциклопедії, карти тощо. З формальної точки зору, платформи для спілкування можуть мати різні організаційні форми: форуми, соціальні мережі, тематичні блоги, платформи запитань і відповідей, а також спеціалізовані платформи для розробки або обміну ресурсами. Важливим параметром, який також варто враховувати, є рівень модерації: від абсолютно відкритих спільнот до суворо контрольованих середовищ, де кожне повідомлення проходить перевірку.

Типова веб-спільнота будується навколо принципів зручної комунікації, тематичної організації контенту та прозорої взаємодії між користувачами. Основою структури такої платформи є головна сторінка або інформаційний дашборд, який служить точкою входу до всього функціоналу. На цій сторінці зазвичай розміщуються актуальні новини спільноти, анонси подій, списки популярних або рекомендованих тем, що дозволяє користувачу швидко зорієнтуватися в динаміці обговорень. Інформація в межах спільноти, як правило, структурована за допомогою тематичних категорій або розділів. Це дозволяє учасникам ефективно знаходити матеріали за своїми інтересами — техніка, наука,

культура, освіта, дозвілля тощо. У межах кожного тематичного блоку створюються обговорення, які починаються з ініціативного повідомлення (теми), після чого учасники можуть відповідати, коментувати, обговорювати або доповнювати інформацію. Такий формат підтримує розвиток діалогу та формування колективного знання. Користувачі в межах спільноти мають власні профілі, в яких відображається їх активність, кількість публікацій, коментарів, рівень репутації чи балів за внесок. Система оцінювання контенту — у вигляді лайків, голосувань або карми — виконує регулятивну функцію: виділяє найцінніші коментарі, підвищує авторитет корисних дописувачів та допомагає новачкам зорієнтуватись у великому потоці інформації. Крім того, візуалізація досягнень (наприклад, бейджі, рівні, статуси) створює мотиваційну складову та підвищує залученість користувачів.

Інтеграція інструментів сповіщень (нотифікацій) забезпечує миттєвий зворотний зв’язок — користувач отримує повідомлення про відповіді, згадки або оцінювання своїх дописів. Це стимулює продовження дискусії та підтримує динаміку взаємодії. Веб-спільнота з продуманою архітектурою і функціоналом здатна забезпечити високу якість обміну знаннями, активне залучення учасників, організованість і саморегуляцію інформаційного простору..

1.3.2 Роль штучного інтелекту в покращенні функціоналу веб-спільнот

Штучний інтелект (ШІ) поступово стає фундаментальним інструментом для модернізації веб-спільнот, радикально змінюючи спосіб, у який відбувається взаємодія користувачів із платформою. Зокрема, інтеграція моделей обробки природної мови (Natural Language Processing, NLP), таких як GPT, відкриває нові горизонти для автоматизації, персоналізації та вдосконалення цифрових сервісів. Завдяки цим моделям можливо здійснювати автоматичну модерацію контенту, фільтрувати спам і порушення, надавати миттєві відповіді на запити користувачів, генерувати резюме тривалих обговорень, аналізувати емоційне забарвлення коментарів і забезпечувати інтелектуальний пошук, який орієнтується не лише на ключові слова, а й на сенс запиту. Крім того, GPT-моделі здатні пропонувати індивідуальні рекомендації на основі історії взаємодії користувача, що сприяє його глибшому залученню до спільноти.

Основні напрями використання ШІ:

- Автоматична модерація контенту — виявлення образливих висловлювань, спаму, порушень правил.
- Генерація відповідей / порад — інтеграція ШІ-асистентів, які підказують відповіді на питання.
- Резюмування довгих дискусій — скорочення обговорень до ключових тез.
- Семантичний пошук — замість ключових слів використовується розуміння змісту запиту.
- Аналіз настроїв — розпізнавання тональності коментарів для формування здорової атмосфери.
- Персоналізовані рекомендації — ШІ враховує історію активності користувача, щоб пропонувати релевантний контент.

Такі функціональні можливості штучного інтелекту істотно сприяють оптимізації внутрішніх процесів у межах веб-спільнот. Завдяки автоматичній модерації зменшується навантаження на команду адміністраторів і модераторів, оскільки ШІ здатен оперативно реагувати на порушення правил, відфільтровувати неприйнятний контент та гарантувати безпечний простір для обговорень. Висока швидкість обробки запитів і генерація змістовних відповідей підвищують загальну ефективність комунікації: користувачі отримують підтримку майже миттєво, що позитивно впливає на рівень залученості та довіри до платформи. Крім того, інтегровані ШІ-системи здатні створювати доброзичливе середовище для новачків — вони отримують підказки, персональні рекомендації, пояснення або навіть адаптовану навігацію, що дозволяє їм швидше адаптуватися до функціоналу, знайти однодумців та почати активну участь у спільноті. Ці переваги сприяють зростанню активної аудиторії, посиленню спільнотної культури та загальному покращенню якості цифрової взаємодії.

1.3.3 Вимоги до інтерфейсу та взаємодії користувачів у веб-спільнотах

Якість взаємодії у веб-спільнотах тісно пов’язана з тим, наскільки інтуїтивно зрозумілим, привабливим і функціональним є інтерфейс користувача. Якщо користувач не може швидко зорієнтуватися на сайті або знайти потрібну

інформацію, це призводить до зниження залученості та загального рівня активності. Ідеальний інтерфейс повинен не лише відповідати естетичним очікуванням сучасного користувача, але й бути логічно організованим, інтуїтивно зручним та швидкодійним. Зручна структура розміщення елементів, чітка навігація, доступність функцій, наявність візуального фідбеку (наприклад, спливаючих повідомлень, індикаторів активності, підсвічування елементів при наведенні) — усе це створює позитивний досвід взаємодії. Важливим також є забезпечення адаптивності дизайну під різні пристрої та платформи, оскільки сучасні користувачі очікують однакової якості користування як на комп'ютері, так і на смартфоні чи планшеті. Чим зручніший і доступніший інтерфейс, тим активніше користувачі залучаються до обговорень, коментування, створення контенту, формуючи тим самим живу та ефективну спільноту.

Ключові вимоги

- Адаптивність — інтерфейс повинен коректно відображатись на різних пристроях (ПК, планшет, смартфон).
- UX-навігація — інтуїтивно зрозуміле меню, пошук, категорії, хештеги.
- Швидкість доступу до інформації — мінімізація кліків до цільового контенту.
- Підтримка багатомовності — для міжнародних спільнот.
- Прозора система повідомлень — нотифікації про відповіді, вподобання, згадки.
- Можливості персоналізації — зміна тем оформлення, підбір рекомендованого контенту.
- Інклюзивність — підтримка доступності для людей з інвалідністю.

Усі перелічені аспекти взаємодії користувачів із платформою — інтуїтивно зрозумілий інтерфейс, адаптивний дизайн, багатомовність, персоналізація, підтримка зворотного зв'язку — у комплексі забезпечують сприятливе середовище для ефективного обміну інформацією та колективного розвитку. У такому середовищі користувачі не обмежуються споживанням контенту, а навпаки —

активно залучаються до його створення, аналізу та вдосконалення. Вони не тільки коментують і обговорюють, але й ініціюють нові теми, модерують простір і формують культурні та експертні стандарти спілкування в спільноті. Веб-спільноти виступають не лише як інструмент комунікації, але й як рушійна сила цифрової трансформації суспільства. Вони сприяють розширенню цифрової грамотності, зміцненню соціальних зв'язків, поширенню знань та колективному інтелектуальному розвитку. Їх подальше вдосконалення тісно пов'язане з розвитком штучного інтелекту та інноваційних цифрових технологій, які здатні перетворити онлайн-спілкування на ще більш інклюзивний, ефективний і масштабований процес. Це відкриває нові горизонти для електронної демократії, віддаленої освіти, хмарного колективного виробництва та глобального обміну знаннями.

1.4 Аналіз існуючих рішень та аналогів веб-ресурсів

У цьому підрозділі здійснюється всебічний огляд сучасних програмних рішень, платформ і сервісів, які реалізують генерацію текстів за допомогою мовних моделей, а також прикладів веб-спільнот, у яких успішно інтегровано інструменти штучного інтелекту. Детальний аналіз таких систем дає змогу не лише оцінити їхні функціональні можливості, але й глибше зрозуміти їхню архітектуру, сценарії використання та обмеження. Розглядаються як великі комерційні продукти (наприклад, OpenAI ChatGPT, Jasper.ai, Copy.ai), так і спеціалізовані AI-плагіни до форумів (Discourse, Reddit, Stack Overflow з елементами AI-модерації). Окрему увагу приділено тому, як реалізується взаємодія користувача з AI — чи є вона активною (у вигляді чат-бота, резюмування або генерації тем), чи пасивною (автоматичні пропозиції, модерація, переклад). Такий підхід дозволяє виокремити найбільш ефективні інструменти, які можуть бути адаптовані або вдосконалені у рамках розробки власної системи. Також досліджуються обмеження наявних рішень — від обмежень у контексті безкоштовного використання API до проблем з точністю, етичними ризиками або складністю впровадження. Підсумки цього аналізу дозволяють визначити сильні та слабкі сторони поточних технологій та

сформувати обґрунтовані вимоги до розробки інтегрованої AI-системи у веб-середовище.

1.4.1 Огляд платформ з генерацією тексту

Упродовж останніх років спостерігається стрімке зростання кількості онлайн-платформ, що надають користувачам можливість генерувати тексти за допомогою технологій штучного інтелекту. Цей ринок активно розвивається внаслідок комерційного інтересу до автоматизованого створення контенту для соціальних мереж, блогів, маркетингових кампаній, документації, а також навчальних і технічних матеріалів. Найбільш відомими прикладами таких платформ є сервіси, побудовані на основі моделей GPT (Generative Pre-trained Transformer), зокрема ChatGPT від OpenAI, Jasper AI, Copy.ai, Writesonic, Rytr, Notion AI, Frase.io та інші. Ці платформи забезпечують можливість створення широкого спектра текстових матеріалів: від коротких інформаційних повідомлень, SEO-оптимізованих статей до описів товарів, маркетингових слоганів, сценаріїв для відео та навіть художньої прози. Завдяки вбудованим налаштуванням вони дають змогу персоналізувати стиль тексту, його довжину, тональність, цільову аудиторію, а також мову, що є важливою перевагою для міжнародних користувачів.

Технічно ці системи працюють за допомогою API, які звертаються до мовних моделей великого масштабу (наприклад, GPT-3.5, GPT-4), що навчені на мільярдах фрагментів тексту з відкритих джерел. Завдяки цьому вони можуть створювати контент, який за своєю структурою і мовною побудовою дуже схожий на людський. Крім того, відкритість API дозволяє вбудовувати генеративний функціонал безпосередньо в бізнес-процеси: від CRM-систем і систем управління завданнями до платформ електронної комерції, онлайн-чату, навчальних застосунків і вебінтерфейсів. Однак, попри безперечні переваги, існує низка суттєвих обмежень. По-перше, фінансова вартість використання API великих моделей при масштабному застосуванні може бути значною, особливо для стартапів або малого бізнесу. По-друге, штучний інтелект не завжди гарантує фактологічну достовірність: існує ризик генерації хибної, неперевіреної або упередженої інформації, оскільки моделі не мають справжнього розуміння фактів. Також у

деяких випадках відсутня стилістична узгодженість між окремими частинами довгого тексту або відповідей, особливо якщо генерація відбувається блоками. По-третє, виникають питання авторських прав, добросовісного використання контенту та ризики плагіату.

Платформи з генерацією тексту відкривають нові можливості для автоматизації творчої діяльності, їх ефективне використання вимагає критичного підходу, технічної грамотності та етичної відповідальності.

1.4.2 Огляд веб-спільнот з інтеграцією AI-інструментів

З розвитком генеративного штучного інтелекту (ШІ) спостерігається глибока трансформація цифрових спільнот, які дедалі частіше інтегрують AI-інструменти з метою підвищення ефективності взаємодії, автоматизації процесів та покращення користувацького досвіду. Наприклад, на таких популярних платформах, як Reddit, GitHub, Discord і Stack Overflow, активно впроваджуються різні форми ШІ-підтримки: автоматизовані модератори, віртуальні асистенти для користувачів, боти для підсумовування довгих гілок обговорення, інструменти генерації відповідей на типові запити, а також системи виявлення токсичного контенту. Це дозволяє зменшити навантаження на модераторів, пришвидшити комунікацію та зробити середовище більш інклюзивним. Окрему увагу заслуговує функціональність «інтелектуального перекладу», яка впроваджується у форумах для забезпечення мультикультурної комунікації без мовних бар'єрів. Наприклад, Discourse — популярна система для створення форумів з відкритим кодом — пропонує інтеграцію з OpenAI API через плагіни, що дозволяють не лише перекладати повідомлення в реальному часі, а й автоматично генерувати резюме, пропонувати теми обговорення, виділяти ключові тези з обговорення та формувати відповідні теги.

У спеціалізованих технічних спільнотах, таких як HuggingFace, AI Stack Exchange або навіть внутрішні Slack-групи для інженерів, ШІ-інструменти активно використовуються для покращення технічної підтримки. Наприклад, мовні моделі інтегруються для генерації прикладів коду, автоматичного коментування фрагментів програм, пояснення складних понять з машинного навчання,

комп'ютерного зору, статистики тощо. Деякі спільноти створюють навіть окремі інтерактивні підрозділи на базі чат-ботів, де користувачі можуть отримати швидку консультацію без очікування відповіді від інших учасників. Зростає кількість експериментів із використанням генеративного ШІ для прогнозування тем, які можуть стати популярними у спільнотах, виявлення потенційно конфліктних обговорень ще до їх загострення, або навіть автоматичного формування вітальних повідомлень для нових учасників на основі їх інтересів.

Впровадження AI у веб-спільноти вже не є елементом футуристичного бачення — це активний процес, що змінює принципи взаємодії в цифровому просторі. Інструменти ШІ виступають не просто як допоміжний елемент, а як рушій інновацій, що формує нові стандарти ефективності, зручності та персоналізації в рамках онлайн-комунікації.

1.4.3 Висновки щодо переваг та недоліків існуючих рішень

Проведений аналіз підтверджує, що сучасні рішення у сфері генерації текстів та інтеграції штучного інтелекту у веб-спільноти мають великий потенціал трансформувати цифрове середовище. Вони дозволяють не лише автоматизувати рутинні завдання, але й підвищити якість взаємодії між користувачами, персоналізувати контент і забезпечити більш ефективну модерацію та підтримку. Зокрема, до беззаперечних переваг таких рішень можна віднести:

- Висока швидкість створення контенту.
- Адаптивність до потреб користувача.
- Можливість автоматизації повторюваних завдань.
- Підвищення ефективності модерування.
- Зниження мовних бар’єрів у спільнотах.

До недоліків належать:

- Ризик поширення дезінформації або фактологічних помилок.
- Обмежена контекстуальна обізнаність моделей.
- Можливе дублювання або неоригінальність контенту.
- Висока вартість використання API на масштабах.

- Потреба в регулярному контролі та ручній перевірці.

Попри очевидні досягнення в галузі, повноцінне впровадження штучного інтелекту (ШІ) у веб-спільноти залишається комплексним завданням, яке потребує багаторівневого підходу. Передусім, необхідне стратегічне планування впровадження — з урахуванням специфіки платформи, технічних ресурсів, очікувань користувачів і потенційних сценаріїв використання AI. Важливо також забезпечити постійний технічний супровід, який включає налаштування, тестування, моніторинг якості відповідей, оновлення моделей та реагування на збої. Крім технічного аспекту, не менш значущими є етичні та правові питання. Виникає потреба у створенні чітких правил щодо конфіденційності, використання персональних даних, попередження упередженості у відповідях моделей, боротьби з дезінформацією, а також захисту авторських прав. Необхідна прозорість у тому, як і для чого використовуються дані користувачів, як приймаються автоматизовані рішення та хто несе відповідальність за їх наслідки.

Успішні приклади, такі як GitHub Copilot, Notion AI або Discourse із плагінами на основі GPT, демонструють, що при грамотній інтеграції ШІ не лише оптимізує робочі процеси й покращує комунікацію, але й підвищує загальну цінність цифрової платформи. Наприклад, автоматичне підсумовування дискусій економить час користувачів, персоналізовані відповіді зменшують кількість повторюваних запитів, а мовна адаптація сприяє глобальній інклюзії. Штучний інтелект має всі передумови стати не просто допоміжним інструментом, а ключовим елементом інфраструктури веб-спільнот нового покоління. Його впровадження — це не технічна мода, а невідворотна складова цифрової еволюції соціальної взаємодії.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ГЕНЕРАЦІЇ ТЕКСТІВ ТА ВЕБ-СПІЛЬНОТИ

2.1 Формулювання вимог до проектованої системи

Етап формулювання вимог є критично важливим для забезпечення ефективної реалізації проєкту, оскільки саме він визначає фундаментальні засади функціонування майбутньої системи. На цьому етапі детально фіксуються ключові характеристики, можливості, технічні та організаційні обмеження, а також потреби користувачів і очікування зацікавлених сторін. Чітке визначення вимог дозволяє уникнути двозначностей, запобігає помилкам у комунікації між замовником і розробниками та закладає основу для ефективного управління проєктом протягом усього життєвого циклу. Успішно сформульовані вимоги підвищують передбачуваність результатів, забезпечують цілісність архітектурних рішень і створюють умови для тестування, оцінювання якості та подальшої масштабованості програмного продукту. Саме тому цей етап є одним із найважливіших у процесі створення сучасних інтелектуальних систем.

2.1.1 Функціональні вимоги

Функціональні вимоги описують конкретні дії, які система має виконувати. Для автоматизованої системи генерації текстів з інтегрованою веб-спільнотою ці вимоги охоплюють широкий спектр задач (рис. 2.1):

*Реєстрація та автентифікація користувачів***:** система повинна забезпечувати зручний та безпечний механізм створення облікового запису. Користувачам має надаватися можливість реєстрації за допомогою електронної пошти та пароля, а також через соціальні мережі (наприклад, Google або Facebook), що значно спрощує процес входу. Для підвищення рівня безпеки важливо впровадити двофакторну автентифікацію (2FA) — наприклад, шляхом надсилання одноразового коду на електронну пошту або мобільний телефон. Крім того, слід передбачити можливість відновлення доступу через перевірку особи або зміну пароля. Усі операції мають проходити через захищені протоколи зв’язку з використанням шифрування.

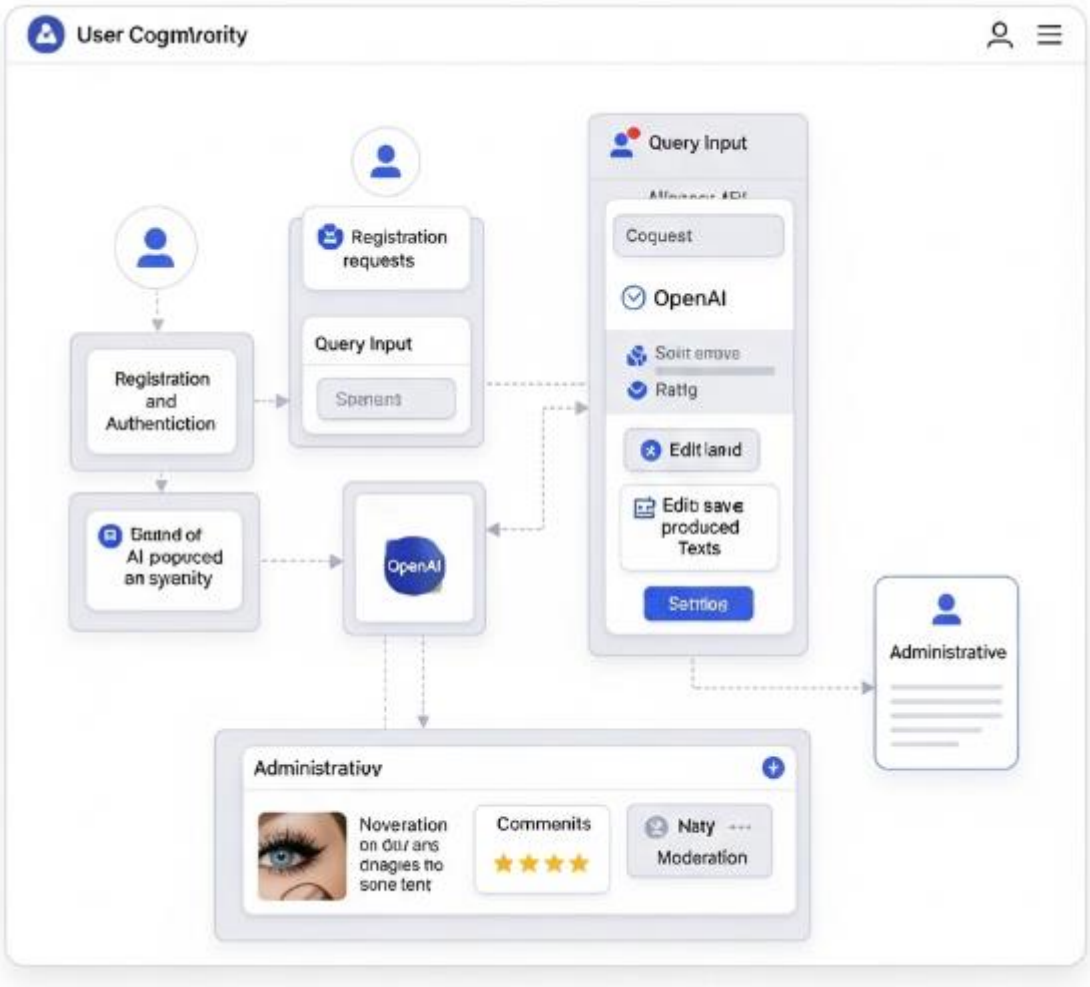


Рисунок 2.1 Зовнішній вигляд макету проектованого веб-ресурсу

Інтерфейс введення запиту. Користувач повинен мати інтуїтивно зрозумілий інтерфейс для введення текстового запиту. Це може бути поле введення, в яке користувач вводить тему, короткий опис або конкретне формулювання задачі. У системі також доцільно реалізувати підказки, автозаповнення або вибір із шаблонів для полегшення взаємодії. Введений запит надсилається до серверної частини, яка виконує обробку за допомогою мовної моделі. Після генерації відповідь відображається у структурованому вигляді, із можливістю копіювання, редагування або повторного використання.

Взаємодія з OpenAI API. Необхідна інтеграція з зовнішнім API для надсилання запитів і отримання відповідей у реальному часі. Це передбачає обробку запитів через серверну частину застосунку з урахуванням обмежень тарифікації та оптимізації кількості запитів. Успішна інтеграція повинна гарантувати стабільну роботу навіть у разі підвищеного навантаження.

Редагування та збереження згенерованих текстів. Користувачі повинні мати можливість переглядати, редагувати, копіювати або завантажувати результати генерації. У системі варто передбачити збереження історії генерацій у профілі користувача, що забезпечує зручний доступ до попередніх запитів та їх результатів.

Система коментування та оцінювання. З метою підтримки взаємодії в спільноті передбачена можливість залишати коментарі, оцінки або позначки до згенерованих текстів. Це дозволяє підвищити залучення користувачів і створити живу екосистему зворотного зв'язку.

Адміністративна панель. Адміністратори повинні мати змогу переглядати статистику використання, керувати користувачами, контролювати генерований контент, а також встановлювати обмеження доступу або накладати санкції у випадках порушення політики спільноти. Панель управління має містити інструменти моніторингу продуктивності системи та інтерфейс для інтеграції з аналітичними службами.

2.1.2 Нефункціональні вимоги (продуктивність, безпека, масштабованість, зручність використання)

Нефункціональні вимоги охоплюють ті аспекти системи, які не визначають конкретну поведінку або функціональність, але критично впливають на ефективність, надійність, масштабованість і зручність використання розробленої платформи. Ці вимоги формують якісні характеристики системи, що визначають загальний рівень задоволеності користувачів, знижують імовірність виникнення критичних помилок у процесі експлуатації та забезпечують стабільність функціонування навіть у складних умовах навантаження. Сюди належать вимоги до швидкодії, часу відгуку, цілісності даних, рівня безпеки, адаптивності інтерфейсу, стійкості до збоїв, а також можливості масштабування та обслуговування системи. Реалізація цих вимог дозволяє створити платформу, здатну не лише відповідати поточним очікуванням, а й адаптуватися до майбутніх змін у вимогах користувачів і зростанні кількості запитів, що надходять.

Продуктивність: система повинна забезпечувати швидкий відгук інтерфейсу (менше 1 сек.) та обробку запиту до OpenAI в межах кількох секунд. Під час пікових навантажень необхідна підтримка черги запитів або розподіленої обробки.

Безпека: усі запити повинні передаватися через захищені канали (HTTPS). Користувацькі дані мають бути зашифровані, а доступ до персональних функцій — обмежений авторизованими сесіями. Важливо передбачити захист від ботів та спам-атак.

Масштабованість: архітектура повинна дозволяти горизонтальне масштабування — підключення додаткових серверів, баз даних чи балансування навантаження без зупинки роботи системи.

Зручність використання (usability): інтерфейс має бути інтуїтивно зрозумілим навіть для користувачів без технічного досвіду. Передбачено адаптивний дизайн для мобільних пристроїв, доступність інтерфейсу для людей з обмеженими можливостями, мінімум кроків до отримання результату.

Поєднання функціональних і нефункціональних вимог створює міцну основу для проєктування інтелектуальної системи генерації текстів із соціальним компонентом у вигляді веб-спільноти. Функціональні вимоги забезпечують реалізацію конкретних сценаріїв користування системою, тоді як нефункціональні формують якісне середовище взаємодії, що відповідає вимогам безпеки, продуктивності, масштабованості та доступності. Завдяки ретельному формулюванню цих вимог можливо не лише досягти високої ефективності на початковому етапі розробки, а й гарантувати адаптивність та стійкість системи в умовах зростаючих навантажень, нових викликів і змінних потреб користувачів. У підсумку, системне підходження до вимог дозволяє створити цілісну, сучасну та конкурентоспроможну платформу, що відповідає як очікуванням кінцевих користувачів, так і технічним стандартам галузі.

2.1.3 Вибір технологічного стеку (backend, frontend, база даних)

У процесі розробки інтелектуальної системи генерації текстів і веб-спільноти критично важливим є підбір ефективного, гнучкого та надійного технологічного

стеку, здатного забезпечити як базовий функціонал, так і потенціал для масштабування. Обрані технології мають відповідати сучасним стандартам веб-розробки, бути сумісними з інтеграцією мовних моделей, а також підтримувати багаторівневу взаємодію з користувачем. На стороні серверної логіки (бекенду) використано мову програмування Python, що поєднує зручність синтаксису з потужним інструментарієм для розробки веб-сервісів. Основою для побудови API і обробки запитів слугує мікрофреймворк Flask. Його гнучкість, легкість у налаштуванні, підтримка REST-архітектури та розширення через численні плагіни дозволяють швидко створювати адаптивні серверні модулі. Flask також забезпечує інтеграцію з WSGI-серверами (наприклад, Gunicorn), підтримку CORS та зручну організацію маршрутизації запитів.

На стороні клієнта (фронтенду) застосовується HTML5 для розмітки, CSS з використанням Tailwind для швидкої адаптивної стилізації та JavaScript для інтерактивності. Tailwind CSS дозволяє реалізовувати UI, що легко масштабується, із підтримкою темної теми, адаптивної верстки, інтерфейсних компонентів (карти, кнопки, поля вводу) без зайвого надмірного коду. JavaScript використовується для обробки подій на стороні клієнта, а також інтеграції з бекендом через AJAX-запити чи WebSockets. Для побудови складніших інтерфейсів із підтримкою стану застосовано React.js — популярну бібліотеку, яка дозволяє створювати динамічні SPA (Single Page Applications). У простіших сценаріях використовується шаблонізатор Jinja2, що забезпечує серверний рендеринг HTML із вставками змінних у шаблони — особливо корисно при генерації PDF або email-шаблонів. У якості СУБД обрано два варіанти: SQLite для локального використання, прототипування або систем із низьким навантаженням, та PostgreSQL — для продуктивного розгортання з високою надійністю, підтримкою транзакцій, індексації, повнотекстового пошуку та масштабування. ORM-бібліотека SQLAlchemy використовується для зв’язку між моделями та базою даних, спрощуючи запити й управління даними.

Для інтеграції мовної моделі OpenAI GPT реалізовано взаємодію через HTTPS-запити. Секретні ключі зберігаються в захищеному середовищі (наприклад,

у .env-файлах), а всі запити до API здійснюються з урахуванням лімітів, повторних спроб у разі помилок та логування відповіді. Це дозволяє гнучко налаштувати логіку генерації текстів, зберігати історію запитів і використовувати параметри, такі як стиль, довжина, мова, тон повідомлення тощо.

Обраний технологічний стек дозволяє ефективно реалізувати всі функціональні та нефункціональні вимоги до системи, підтримує масштабованість, безпеку, адаптивність і забезпечує високу швидкість взаємодії між компонентами інтелектуального застосунку.

2.2 Архітектура проектованої системи

Архітектура інтелектуальної системи генерації текстів та веб-спільноти ґрунтується на сучасних принципах побудови складних програмних систем, зокрема модульності, розширюваності, масштабованості, адаптивності та безпечної інтеграції з зовнішніми сервісами. Модульність передбачає розділення системи на незалежні функціональні блоки (рис. 2.2), кожен з яких виконує конкретне завдання (генерація тексту, керування користувачами, обробка запитів тощо).

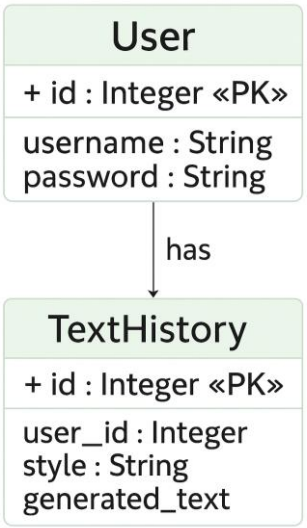


Рисунок 2.2 Структура сформованої бази даних

Такий підхід значно спрощує розробку, тестування, обслуговування та оновлення системи, оскільки зміни в одному модулі не впливають критично на інші. Масштабованість дає змогу збільшувати обчислювальні потужності або

кількість користувачів без втрати продуктивності, що досягається завдяки правильній організації комунікації між компонентами та використанню відповідних інфраструктурних рішень. Інтеграція із зовнішніми сервісами, зокрема OpenAI API, забезпечує розширення функціоналу системи без необхідності розробки складних алгоритмів «з нуля». Одним із центральних рішень у побудові архітектури стало логічне розділення системи на клієнтську, серверну та інтеграційну частини, що дозволяє досягти гнучкості, зменшити навантаження на сервер та забезпечити швидку реакцію інтерфейсу користувача. Такий архітектурний підхід є критично важливим для динамічних систем, що орієнтовані на активну взаємодію з користувачем та використання потужних інструментів штучного інтелекту.

2.2.1 Загальна архітектура системи: клієнт-серверна модель

Загальна архітектура системи реалізована за принципом клієнт-серверної моделі, яка є однією з найпоширеніших і найефективніших архітектурних парадигм у сучасній веб-розробці. У цій моделі клієнт (фронтенд) виконує функцію взаємодії з користувачем, забезпечує зручний і динамічний інтерфейс, обробку подій, валідацію даних на боці клієнта та передачу запитів до сервера. Серверна частина (бекенд) відповідає за опрацювання бізнес-логіки, доступ до бази даних, обробку запитів і відповідей, а також взаємодію з зовнішніми сервісами, зокрема OpenAI API. Ключовим аспектом є комунікація між клієнтом і сервером, яка реалізується через REST API — уніфікований інтерфейс передачі даних у форматі JSON за допомогою HTTP-запитів. Для забезпечення вищого рівня інтерактивності та підтримки реального часу (наприклад, оновлення коментарів або результатів генерації тексту без перезавантаження сторінки) додатково використовується протокол WebSocket. Це дає змогу встановити постійне двостороннє з’єднання між клієнтом і сервером, завдяки чому обмін даними відбувається миттєво і з низькою затримкою.

Таке архітектурне рішення дозволяє досягти не лише високої продуктивності при взаємодії з великою кількістю одночасних користувачів, а й забезпечує масштабованість завдяки можливості розширення інфраструктури за допомогою

горизонтального або вертикального масштабування серверів. Система здатна адаптуватися до різних типів клієнтських пристроїв, включаючи мобільні телефони, планшети та десктопи, що досягається завдяки застосуванню адаптивного дизайну та гнучкої клієнтської логіки. Важливою перевагою є можливість безперебійного розгортання у хмарних середовищах, таких як AWS, Google Cloud або Azure, з підтримкою автоматичного масштабування, балансування навантаження та розподіленого зберігання даних. Крім того, подібна архітектура дозволяє впроваджувати оновлення окремих модулів або компонентів без необхідності зупинки роботи всієї системи, що є критично важливим для стабільності та безперервної доступності сервісу. Усе це робить обрану архітектуру надзвичайно ефективною для реалізації складних, динамічних та масштабованих інтелектуальних вебсервісів.

2.2.2 Архітектура модуля генерації тексту (інтеграція з OpenAI API)

Модуль генерації тексту — це ключовий компонент серверної частини інтелектуальної системи, який виконує завдання перетворення вхідного запиту користувача у сформований промпт, що надсилається до OpenAI API. Архітектура цього модуля побудована з урахуванням гнучкості, розширюваності та безпеки. Основним завданням модуля є не лише ініціація запиту до мовної моделі, а й попередня підготовка тексту, обробка відповіді та контроль якості згенерованого контенту. На першому етапі запит користувача проходить через спеціалізований обробник, який виконує синтаксичний та семантичний аналіз, очищує введений текст від зайвих символів, структурує дані відповідно до шаблону та формує інструкцію у форматі, зручному для GPT-моделі (prompt engineering). Цей підмодуль включає алгоритми класифікації запиту, фільтрації за типом (есе, інструкція, стаття тощо) та інтеграцію з базою знань (наприклад, шаблони для форматування).

Другим важливим підмодулем є авторизація та безпечна взаємодія з OpenAI API. Він містить механізми обробки токенів доступу, обмеження запитів (rate limiting), шифрування чутливих даних та обробку винятків у разі помилок (наприклад, перевищення ліміту або мережеві збої). Після отримання відповіді від

моделі третій підмодуль займається її обробкою: розбивкою на логічні блоки, форматуванням (додавання заголовків, списків, абзаців), фільтрацією небажаного контенту (нецензурна лексика, фейки), а також адаптацією стилю до вибраного користувачем.

Модуль генерації тексту функціонує як інтелектуальний фільтр і посередник між користувачем та мовною моделлю, забезпечуючи точність, безпеку, структурованість і релевантність результату. Його ефективна робота критично впливає на загальне враження користувача від системи.

- обробник запиту (парсинг інструкції та підготовка промпту);
- модуль авторизації та безпечного виклику OpenAI;
- модуль обробки відповіді (нормалізація, форматування, перевірка безпеки контенту).

Цей компонент забезпечує генерацію відповідей у вигляді текстів, порад, коротких статей тощо, з урахуванням тематики і параметрів, заданих користувачем.

2.2.3 Архітектура веб-спільноти: структура бази даних, компоненти інтерфейсу

Компонент веб-спільноти відіграє ключову роль у забезпеченні інтерактивної взаємодії між користувачами, що об'єднані спільними інтересами або тематикою. Цей елемент системи виконує не лише функцію комунікаційної платформи, а й виступає засобом збереження та обміну знаннями, досвідом, а також результатами генерації текстів. У межах веб-спільноти реалізовано сучасний інтерфейс користувача, який дозволяє створювати та брати участь в обговореннях, коментувати, ставити вподобання, переглядати історію активностей, а також керувати власним профілем. Важливою складовою є система автентифікації та авторизації, яка передбачає можливість реєстрації, входу до облікового запису, відновлення доступу та підтвердження електронної пошти. Для підвищення рівня безпеки та гнучкості в управлінні доступом передбачена система ролей: звичайний користувач, модератор та адміністратор. Кожна роль має чітко окреслені повноваження та функціональність, що дозволяє ефективно підтримувати порядок у спільноті. Архітектура компонента передбачає використання бази даних для

зберігання всієї інформації про користувачів, повідомлення, теми, категорії та запити до генеративної моделі. Структура бази даних розроблена відповідно до принципів нормалізації для забезпечення узгодженості та швидкого доступу до інформації. Інтерфейс і логіка веб-спільноти (рис. 2.3) реалізуються з використанням сучасних фреймворків: Flask або Django для серверної частини та React або Vue для клієнтської.

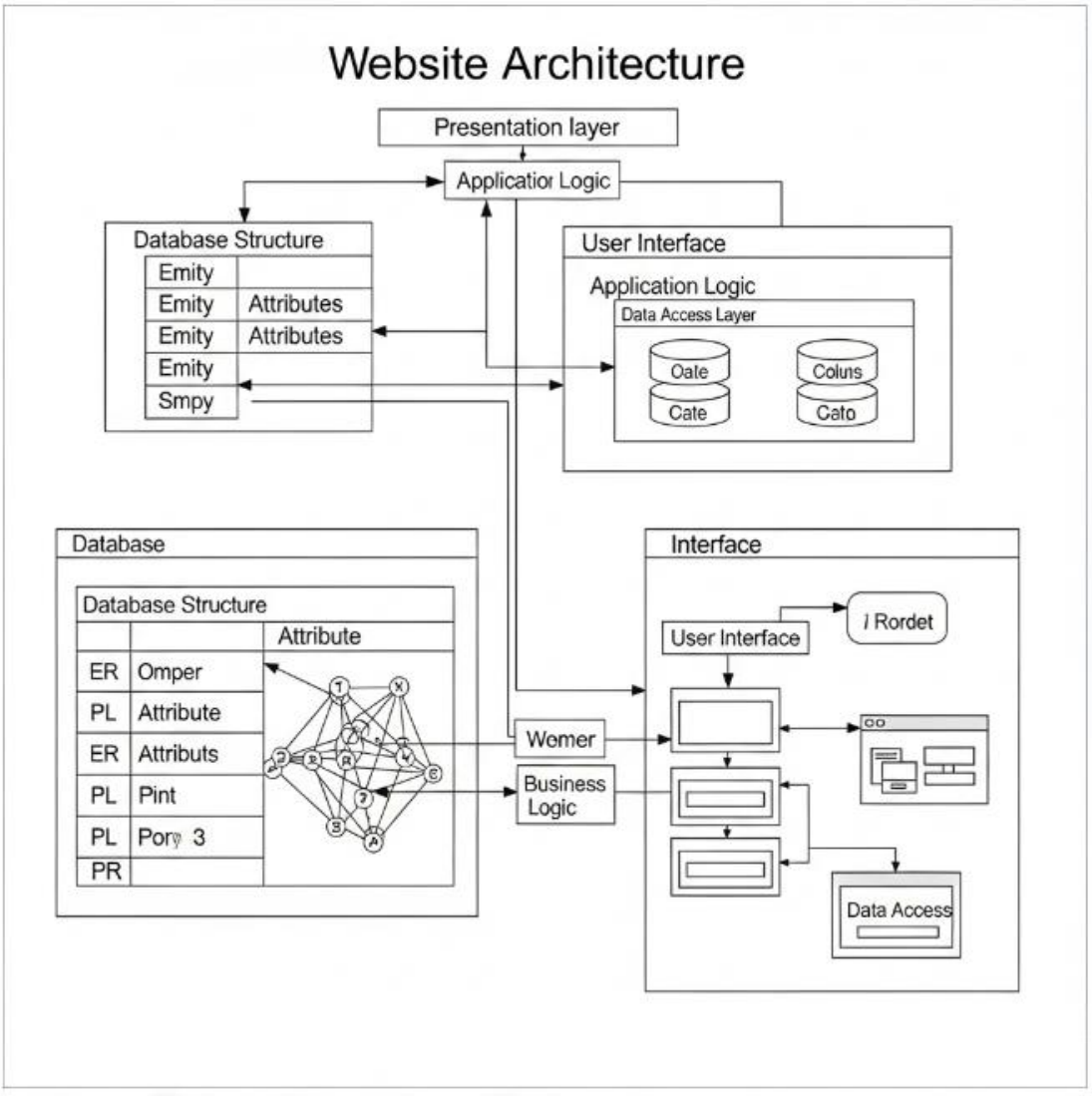


Рисунок 2.3 Архітектура сайту косметологічної веб-спільноти

Завдяки модульності веб-спільнота легко адаптується до нових вимог, підтримує масштабованість і може бути інтегрована з іншими сервісами, такими як системи сповіщень, статистики або інструменти модерації контенту. Таким чином, цей компонент не лише забезпечує взаємодію, але й підвищує якість

обслуговування користувача, створюючи середовище довіри, відкритості та зворотного зв’язку.

- інтерфейс користувача з можливістю створення обговорень, коментарів, перегляду профілю;
- систему автентифікації (реєстрація, вхід, відновлення паролю);
- систему управління ролями (користувач, модератор, адміністратор);
- базу даних з таблицями: користувачі, повідомлення, обговорення, категорії, запити до ШІ.

Для забезпечення масштабованості система будується з використанням фреймворків, таких як Flask або Django для бекенду, і React або Vue для фронтенду.

2.2.4 Взаємодія між компонентами системи

Компоненти системи тісно взаємодіють між собою завдяки чітко визначеним програмним інтерфейсам (API), які забезпечують обмін даними між фронтендом, бекендом та зовнішніми сервісами. Коли користувач вводить тему для генерації тексту, інтерфейс користувача (frontend) обробляє цей запит і передає його через REST або WebSocket до серверної частини (backend). Сервер, у свою чергу, формує промпт, надсилає його до OpenAI API, отримує результат генерації та повертає його на фронтенд для виводу в інтерфейсі. Будь-які дії користувача, пов’язані з активністю в межах спільноти — такі як створення або редагування обговорень, написання коментарів, оцінювання дописів, підписка на теми чи взаємодія з контентом інших користувачів — автоматично фіксуються у системі та зберігаються у відповідних таблицях бази даних. Це дозволяє формувати історію дій кожного користувача, що в подальшому може бути використано для персоналізації досвіду взаємодії, аналітики або модерування. Збережена інформація відображається іншим учасникам відповідно до встановлених прав доступу, що гарантується реалізованим механізмом авторизації, заснованим на ролях користувачів. Таким чином, забезпечується прозорість взаємодії у спільноті, підтримується порядок і посилюється довіра між учасниками платформи.

Для підвищення продуктивності системи реалізовано механізм кешування результатів генерації: у разі повторного запиту на подібну тему система може

повернути вже сформовану відповідь без повторного виклику до зовнішнього API. Це значно знижує затримку, зменшує навантаження на API та скорочує витрати на використання зовнішніх сервісів. Додатково впроваджено журналювання (логування) усіх ключових запитів, що дозволяє адміністраторам проводити моніторинг, виявляти помилки, аналізувати поведінку користувачів і формувати статистичні звіти щодо ефективності системи.

2.3 Проектування бази даних проектованої системи

Проектування бази даних є одним із ключових етапів у створенні надійної та масштабованої системи генерації текстів і функціонування веб-спільноти. Від правильного проєктування структури таблиць і взаємозв'язків між сутностями залежить стабільність, швидкодія, безпечність та ефективність усієї системи. База даних у даному випадку виступає центральним компонентом, що забезпечує збереження всієї інформації — від реєстраційних даних користувачів і згенерованих матеріалів до повідомлень у дискусіях, аналітичних логів та історії активностей. При збільшенні обсягів даних і кількості користувачів зростають вимоги до продуктивності, надійності та цілісності даних. Саме тому проєктування бази повинно враховувати принципи нормалізації, оптимізацію запитів, використання індексів, а також розглядати можливість розподіленого зберігання або шардінгу. У роботі розглянуто різні типи сутностей, побудовані логічні зв'язки між ними, визначено критичні атрибути та типи взаємодій, які потребують оптимізації.

Окрему увагу приділено вибору типу СУБД на різних етапах: для локального прототипування — простих рішень типу SQLite, а для реального розгортання — високопродуктивних систем на кшталт PostgreSQL. У поєднанні з механізмами кешування, журналювання, аудиту, підтримки транзакцій та контролю доступу — це дозволяє сформувати стабільну архітектуру з перспективою подальшого розвитку та інтеграції з іншими AI-компонентами.

2.3.1 Схема даних для користувачів, згенерованих текстів, постів, коментарів, взаємодій

Для забезпечення повноцінної функціональності веб-спільноти, база даних повинна включати декілька ключових сутностей:

1. Користувачі (Users) – таблиця зберігає інформацію про кожного зареєстрованого користувача: унікальний ідентифікатор, ім’я, email, пароль (захешований), роль у системі (користувач, адміністратор), дата реєстрації та інші атрибути.
2. Згенеровані тексти (GeneratedTexts) – ця таблиця містить дані про кожен текст, згенерований за допомогою моделі OpenAI. Поля включають: ідентифікатор тексту, ідентифікатор користувача, тему/запит, згенерований результат, дату створення, тип тексту (есе, інструкція тощо).
3. Пости (Posts) – таблиця, яка зберігає обговорення, створені користувачами на форумі. Вона містить заголовок, текст, автора, дату створення, категорію, статус (активний/архівований).
4. Коментарі (Comments) – таблиця, що зберігає відповіді користувачів на пости. Містить ідентифікатор посту, ідентифікатор автора коментаря, вміст коментаря, дату публікації.
5. Взаємодії (UserActions) – таблиця для фіксації дій користувача у системі: входи, перегляди, оцінки, збереження текстів, генерація запитів, скарги. Це дає змогу здійснювати аналітику поведінки та персоналізувати досвід користувачів.

Для реалізації зв’язків між сутностями в базі даних використовуються зовнішні ключі (foreign keys), які дозволяють встановити логічні та структурні залежності між таблицями. Це не лише забезпечує цілісність даних, але й полегшує побудову складних запитів до бази. Наприклад, таблиця GeneratedTexts має поле user_id, яке є зовнішнім ключем і посилається на primary key таблиці Users. Таким чином, можна легко отримати всі тексти, які були згенеровані певним користувачем, або навпаки — з'ясувати, хто є автором конкретного згенерованого тексту. Таблиця Comments реалізує одразу два зовнішні ключі: post_id, що посилається на таблицю Posts, та user_id — на таблицю Users. Це дозволяє точно встановити, до якого посту належить конкретний коментар і хто його створив. Крім

того, подібні зв'язки дають змогу реалізувати механізми каскадного видалення або оновлення, тобто забезпечити автоматичну підтримку узгодженості даних при зміні або видаленні записів. Усе це критично важливо для стабільної роботи системи, аналітики активності користувачів та організації персоналізованих функцій..

2.3.2 Обґрунтування вибору системи управління базами даних

Для даного проєкту обрано реляційну систему управління базами даних (СУБД) – SQLite або PostgreSQL, залежно від етапу та середовища розгортання. SQLite є оптимальним варіантом для локальної розробки, прототипування, юніт-тестування та легких клієнтських застосунків завдяки своїй простоті, легкості налаштування, відсутності необхідності в окремому сервері та чудовій інтеграції з мовами програмування (особливо Python, PHP, JavaScript через ORM-інструменти). Її файлова структура дозволяє швидко запускати експерименти на одному комп'ютері без складної конфігурації. Натомість PostgreSQL обрано як основну СУБД для продакшн-середовища, враховуючи її надійність, високий рівень захисту, підтримку складних транзакцій, одночасної обробки великої кількості запитів і активних підключень, а також гнучку індексацію. PostgreSQL також підтримує розширення, зокрема для машинного навчання (PostgreSQL ML), JSON-запити, геолокаційні дані (PostGIS), а також реплікацію, горизонтальне масштабування та створення резервних копій. Крім того, вона має активну спільноту, численні плагіни та чудово працює у зв’язці з хмарними сервісами (Heroku, AWS RDS, DigitalOcean). Така комбінація дозволяє створити універсальну архітектуру, де SQLite може слугувати легкою альтернативою на етапах розробки, а PostgreSQL — надійною основою для розгортання в реальному середовищі з великою кількістю користувачів і динамічним навантаженням.

Переваги PostgreSQL включають:

- Підтримку складних запитів та процедур;
- Надійність та відповідність ACID-властивостям;
- Підтримку JSON та інших сучасних типів даних;
- Можливість реплікації та розширення.

Структура бази даних розроблена з урахуванням комплексу вимог до функціональності, захисту даних, продуктивності, масштабованості, гнучкої адаптації до змін бізнес-логіки, а також потреб аналітики. Вона включає логічно нормалізовану модель із чітким поділом на сутності, які відображають користувачів, тексти, взаємодії, обговорення та адміністративні події в системі. Такий підхід дозволяє мінімізувати надлишковість, уникнути аномалій при оновленні даних і забезпечити узгодженість записів. Особлива увага приділена підтримці високої доступності та швидкому масштабуванню — через індексацію полів, кешування часто використовуваних запитів та можливість поділу даних на логічні шардовані області. Оскільки система тісно взаємодіє з API OpenAI, структура бази також адаптована для зберігання метаінформації про запити до моделей, логування відповідей та обробки результатів, що дозволяє вести аналітику точності й продуктивності генерацій. Усі ці компоненти створюють основу для побудови стабільної, надійної та масштабованої платформи, готової до інтеграції як із зовнішніми сервісами, так і з інструментами штучного інтелекту.

2.4 Розробка інтерфейсу користувача (Frontend)

Розробка інтерфейсу користувача (frontend) є критичним аспектом побудови інтелектуальної системи, оскільки саме через візуальну оболонку здійснюється основна взаємодія кінцевого користувача з функціональністю платформи. Основною метою цього етапу є створення зручного, інтуїтивного, привабливого і адаптивного середовища, яке одночасно буде підтримувати інтеграцію із ШІ-сервісами, забезпечувати ефективну навігацію у спільноті, і відповідати очікуванням сучасного цифрового користувача. Особливістю цієї системи є подвійна спрямованість інтерфейсу: з одного боку — інтерактивна панель генерації тексту за допомогою OpenAI API, з іншого — соціальна частина у вигляді динамічного форуму/спільноти з підтримкою дописів, коментарів, профілів, рейтингу, перекладу та модерації.

Під час розробки було враховано принципи Human-Centered Design, Material Design, WCAG-доступності для маломобільних груп, а також сучасні тенденції в

дизайні користувацьких інтерфейсів. Вся логіка реалізована на сучасному стеку технологій — React/Vue на клієнті, REST API для взаємодії з сервером, WebSocket для оновлення повідомлень у реальному часі, i18next для багатомовності, адаптивна верстка із CSS Grid/Flexbox.

Основні блоки інтерфейсу:

- навігаційна панель з іконками та кнопками швидкого доступу;
- головна сторінка генерації текстів з історією запитів;
- стрічка обговорень та постів спільноти;
- модулі перегляду/редагування постів і коментарів;
- розділ налаштувань з опціями персоналізації;
- підтримка світлої та темної теми, адаптація до екранів різної ширини;
- системи сповіщень, модальних вікон, підтверджень дій тощо.

Інтерфейс реалізовано таким чином, щоб навіть користувач без технічної підготовки міг швидко освоїти систему: логічні групування, мінімальна кількість кліків до дії, збереження контексту, використання знайомих шаблонів (іконки, меню, перемикачі, закладки).

Frontend-платформа не просто візуалізує дані, а є інтерактивним інструментом генерації, навігації, комунікації та модерації, що визначає загальний користувацький досвід роботи з інтелектуальною системою нового покоління.

2.4.1 Дизайн-концепція та користувацький досвід (UX/UI)

Інтерфейс користувача — це один із ключових факторів успіху будь-якого сучасного вебзастосунку, особливо у випадку платформи, що поєднує можливості штучного інтелекту з функціональністю веб-спільноти. UX/UI-дизайн був спроєктований таким чином, щоб не лише привабити користувача з перших секунд, але й забезпечити комфортне, ефективне та логічно структуроване використання системи незалежно від рівня технічної підготовки. Дизайн заснований на принципах Human-Centered Design, де головним є досвід і потреби реального користувача. Інтерфейс є інтуїтивно зрозумілим, мінімалістичним, без зайвого перевантаження елементами. Стилістика платформи дотримується сучасних UI-

трендів: flat-дизайн, плавні анімації, soft shadows, скруглені елементи, акцентні кольори з достатнім контрастом для забезпечення високої читабельності. Інтерфейс також адаптивний, тобто коректно масштабується під різні розміри екранів — від смартфонів до великих моніторів.

Ключові елементи інтерфейсу включають:

- Головну панель навігації з логотипом, кнопками переходу до основних розділів (форум, генерація, профіль, історія);
- Стрічку обговорень у вигляді карток з темами, кількістю відповідей, іконками активності;
- Панель генерації тексту з полем для вводу запиту, селектором типу тексту та кнопкою запуску генерації;
- Модальні вікна для створення або редагування постів, перегляду згенерованих відповідей;
- Кабінет користувача з особистою інформацією, історією запитів і можливістю налаштування інтерфейсу.

Інтерфейс демонструє вивід згенерованих текстів на теми догляду та б’юті-трендів. На рис. 2.4 показано приклад генерації тексту за темою «Тренди макіяжу 2025: що в моді?», а на рис.1.2— рекомендації щодо вибору тонера для сухої шкіри. Обидві панелі містять чітку структуру: заголовок, тематичні картки з текстами, інтегровану кнопку запуску «Чат з б’юті-ботом» та брендований заголовок сайту.

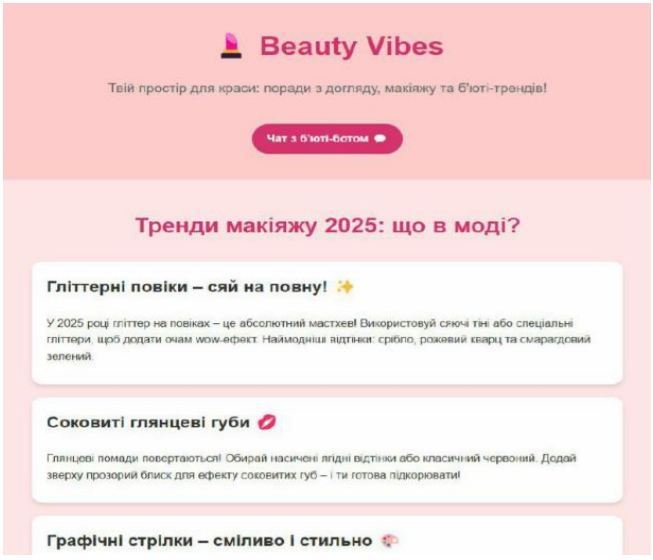


Рисунок 1.1 - Результати роботи сайту-спільноти Beauty Vibes

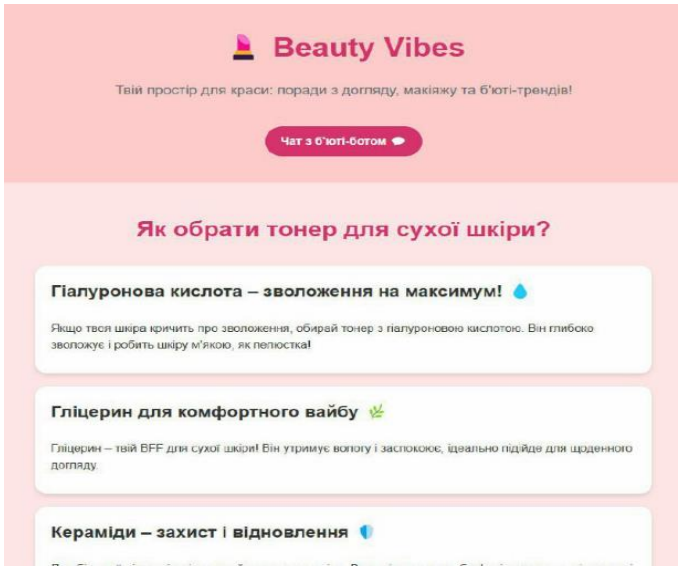


Рисунок 1.2 - Результати роботи сайту-спільноти Beauty Vibes

Колірна палітра та компонування інтерфейсу відповідають принципам зручності та естетичності UX/UI.

2.4.2 Реалізація функціоналу генерації тексту

Функціонал генерації тексту є центральною частиною інтелектуальної системи, і його реалізація відбувається завдяки інтеграції з OpenAI API. На фронтенді передбачено логічно структуровану форму, яка включає поле для введення запиту користувача — це може бути тема, ключові слова або конкретна інструкція щодо стилю/формату тексту. Поруч розміщено селектор типу контенту, який дозволяє обрати між варіантами: есе, порада, опис, маркетинговий текст, повідомлення, оголошення тощо. Нижче розташовано кнопку «Згенерувати», яка ініціює процес створення тексту.

Після активації запиту фронтенд відправляє його на сервер через API-запит, де бекенд готує промпт, звертається до OpenAI API та повертає результат у вигляді відформатованого текстового блоку. Цей блок виводиться в окремому модулі інтерфейсу — у вигляді інтерактивного вікна з можливістю перегляду, редагування чи копіювання результату. Користувач має доступ до історії своїх генерацій у персональному кабінеті, де кожен текст зберігається разом з інформацією про дату, тип запиту та статус обробки. Передбачено можливість відкриття старої версії тексту, її доопрацювання або експорту в PDF-файл із фірмовим шаблоном. Це надзвичайно зручно для тих, хто створює багато різних текстів (копірайтери,

блогери, студенти тощо). Особливу увагу приділено підвищенню зручності використання за допомогою системи підказок (autocomplete). Під час введення запиту система може підказувати поширені теми, ключові фрази або приклади формулювань, що значно полегшує створення ефективних запитів для генерації.

2.4.3 Реалізація функціоналу веб-спільноти (створення постів, коментарів, профілі користувачів)

Функціонал веб-спільноти створено для стимулювання активної участі користувачів у житті платформи та створення середовища взаємодії, підтримки, обміну думками і досвідом. У системі реалізовано повноцінну модель обговорень на базі принципів класичного форуму, доповнену сучасними елементами соціальної платформи. Користувачі мають можливість створювати нові теми для обговорення, розміщувати текстові дописи, додавати до них зображення або посилання, а також використовувати попередньо згенеровані тексти з модуля генерації. Кожен пост містить заголовок, основний вміст, категорію (наприклад, "Догляд", "Макіяж", "Поради від бота"), дату публікації, автора, кількість переглядів і кількість коментарів.

До постів користувачі можуть залишати коментарі, відповідати на них у вигляді вкладених відповідей, редагувати власні повідомлення або видаляти їх. Це реалізує логіку "гілки обговорення" (threaded discussions), яка дозволяє зберігати хронологію та структуру діалогу. Передбачено систему модерації, яка дозволяє видаляти неприйнятний контент, фіксувати порушення правил спільноти та повідомляти модераторів. Окрему увагу приділено функціоналу соціальної взаємодії. Користувачі можуть ставити оцінки дописам (лайки), що формує систему "репутації" або "рейтингу" автора, яка впливає на рівень довіри до нього в межах спільноти. Крім того, впроваджено систему підписок: можна підписатися на обговорення або автора, щоб отримувати сповіщення про нову активність.

У профілі кожного користувача відображається його ім’я, фото (аватар), електронна пошта (або псевдонім, якщо обрано режим анонімності), лічильник постів, лайків, реакцій та активних обговорень. Також виводиться персональна стрічка: список створених тем, коментарів, збережених згенерованих текстів і

історія запитів до OpenAI. Передбачено гнучкі налаштування профілю: зміна паролю, тем оформлення, обмеження видимості (наприклад, приховати активність або email).

Інтерфейс реалізовано у вигляді сучасної панелі з секціями: "Мій профіль", "Мої пости", "Коментарі", "Генерації", "Налаштування". Усе це забезпечує повноцінний, гнучкий і масштабований функціонал соціальної взаємодії в межах інтелектуального форуму.

2.4.4 Забезпечення багатомовності (якщо актуально)

Враховуючи потенційне розширення спільноти до міжнародного масштабу, багатомовність системи є не лише додатковою функціональністю, а стратегічно важливою складовою доступності, інклюзивності та зручності. Інтерфейс користувача реалізовано з урахуванням багатомовної архітектури, де всі текстові компоненти винесено у зовнішні мовні файли — у форматах JSON, YAML або PO/MO залежно від обраної бібліотеки (наприклад, i18next або vue-i18n). Це дозволяє не лише перекладати статичні елементи (меню, кнопки, повідомлення), а й адаптувати інтерфейс під культурні особливості (напр., порядок подання імен, формат дат, напрям письма). Завдяки цьому легко додавати нові мови або редагувати поточні без потреби втручання в кодову базу. Локалізація обирається автоматично при першому вході, базуючись на налаштуваннях мови браузера користувача, але також може бути змінена вручну через меню налаштувань профілю.

Для системного адміністрування передбачено модуль керування мовами, де можна:

- додавати нові мовні файли;
- редагувати переклади онлайн;
- відстежувати відсутні переклади або конфлікти;
- експортувати/імпортувати переклади для роботи в команді локалізаторів.

Крім інтерфейсних компонентів, багатомовність підтримується також у функціоналі генерації тексту. Завдяки інтеграції з OpenAI API користувачі можуть формувати запити будь-якою мовою, а система автоматично генерує відповідь тим

самим мовним кодом. Це забезпечує природну мовну взаємодію без потреби додаткової настройки. При бажанні користувач може окремо обрати мову генерації незалежно від мови інтерфейсу — наприклад, україномовний інтерфейс, але запит англійською мовою. У розділах обговорень або чатах передбачена функція перекладу повідомлень у реальному часі. Біля кожного повідомлення розташована кнопка "перекласти", яка викликає запит до API і відображає переклад без зміни оригіналу. Це особливо корисно для багатонаціональних спільнот, де учасники можуть спілкуватися різними мовами, але залишатися в єдиному інтерфейсі.

Реалізація багатомовності охоплює як статичний інтерфейс, так і динамічний функціонал генерації текстів та спілкування. Такий підхід робить систему гнучкою, інтернаціональною та готовою до масштабування в глобальному середовищі, що є ключовою перевагою для конкурентоспроможності інтелектуальної платформи.

2.5 Розробка серверної частини (Backend)

Серверна частина системи виконує ключову роль у забезпеченні всієї логіки функціонування вебзастосунку. Вона відповідає за прийом і обробку запитів від клієнтів, управління взаємодією з базою даних, обробку запитів до сторонніх сервісів, зокрема OpenAI API для генерації текстів, а також за реалізацію логіки керування користувачами, контентом та правами доступу. Backend-складова діє як «мозок» системи, що забезпечує координацію дій між frontend-інтерфейсом і внутрішніми сервісами. Ретельне проектування цієї частини є необхідним для досягнення високої надійності, продуктивності та масштабованості системи. Саме від стабільності бекенду залежить досвід користувача: швидкість відповіді системи, наявність або відсутність збоїв, правильність обробки помилок, цілісність даних. Побудова сучасного бекенду передбачає впровадження асинхронної обробки запитів, розмежування бізнес-логіки та представлення даних, використання REST або GraphQL API, а також підтримку стандартів безпеки і захисту персональної інформації. Таким чином, серверна частина є критичною опорою всієї архітектури програмного забезпечення й забезпечує ефективну взаємодію між компонентами інтелектуальної системи.

2.5.1 Дизайн-концепція та користувацький досвід (UX/UI)

Хоча бекенд зазвичай не має безпосередньої точки взаємодії з користувачем, його внутрішня архітектура істотно впливає на сприйняття системи в цілому. Всі аспекти, пов’язані з UX — включно з часом відповіді, плавністю взаємодії, коректністю результатів генерації тексту, відсутністю помилок і стабільністю роботи системи — формуються саме завдяки якісному проектуванню серверної частини. Якщо користувач стикається з повільною відповіддю, обірваними сесіями, недоступністю генерації або втратою даних, — навіть при ідеальному інтерфейсі UI це спричиняє негативне враження. Тому UX не можна вважати виключно візуальною складовою. Дизайн бекенду має враховувати кешування запитів, оптимізацію SQL-операцій, ізоляцію запитів, що потребують багато ресурсів, винесення обчислень у фонові черги через Celery, asyncio або task queue з пріоритетами. Крім цього, важливо реалізовувати відкладену обробку великих запитів (наприклад, генерації великих текстів) у паралельному режимі, забезпечуючи своєчасне повернення результатів без блокування користувача. Системи моніторингу (Prometheus, Sentry) дозволяють виявляти «вузькі місця» в продуктивності, а профілювання допомагає локалізувати найвитратніші частини коду. Тобто архітектурне рішення має бути не лише технічно ефективним, а й UX-орієнтованим на рівні швидкодії, безпеки та прогнозованої реакції на дії користувача.

2.5.2 Реалізація функціоналу управління користувачами та контентом спільноти

Функціонал включає модулі:

- реєстрація та вхід користувача;
- редагування профілю, збереження історії генерацій;
- створення, редагування, видалення постів і коментарів;
- модерація вмісту (скарги, приховування, фільтрація);
- рейтингова система (лайки, голосування, активність);
- логування дій користувачів для безпеки й аналітики.

Управління контентом здійснюється через RESTful API-запити, які забезпечують ефективну двосторонню взаємодію між інтерфейсом користувача (frontend) та серверною логікою (backend). Кожен запит проходить через систему маршрутизації, де відбувається перевірка автентичності та авторизації користувача, після чого виконується відповідна дія — збереження нового посту, редагування або видалення. Контент зберігається у структурованому вигляді в базі даних, що дозволяє виконувати складні запити фільтрації, пошуку та сортування. Ключовим елементом є реалізація гнучкої ієрархічної системи ролей (звичайний користувач, модератор, адміністратор), яка визначає рівень доступу до функціоналу: від базового створення і коментування постів до видалення чужого контенту, блокування користувачів і доступу до аналітики. Такий підхід дозволяє реалізувати безпечну багаторівневу систему керування, яка легко масштабується при зростанні аудиторії або впровадженні нових ролей. У майбутньому ця система може бути доповнена механізмами делегування повноважень і аудиту дій адміністраторів..

2.5.3 Механізми аутентифікації та авторизації

Для забезпечення безпеки система реалізує:

- JWT-токени для захисту сесій;
- bcrypt або Argon2 для шифрування паролів;
- механізм відновлення паролю через email;
- двофакторну авторизацію (опційно);
- middleware, що перевіряє права доступу до маршрутів.

Аутентифікація може бути реалізована кількома способами, залежно від потреб користувача та рівня безпеки, що вимагається. Традиційний метод включає введення логіна та пароля, при цьому паролі зберігаються у зашифрованому вигляді (наприклад, за допомогою bcrypt або Argon2). У сучасних системах все частіше застосовуються зовнішні сервіси авторизації на основі протоколу OAuth2, які дозволяють користувачеві входити до системи за допомогою свого акаунту в Google, Facebook або інших авторитетних постачальників ідентифікації. Такий підхід не лише спрощує процес входу, а й підвищує рівень безпеки, знижуючи

ризик крадіжки паролів. Крім того, інтеграція з зовнішніми сервісами забезпечує багатофакторну перевірку автентичності, що є важливим у контексті захисту персональних даних. У системі передбачено вибір способу входу, а також можливість поєднання кількох варіантів — наприклад, використання Google-автентифікації разом із двофакторним підтвердженням через SMS або email..

2.5.4 Валідація даних та обробка помилок

Перед записом у базу всі вхідні дані проходять перевірку:

- типів (число, рядок, email);
- допустимих значень (у рамках словників);
- відсутності SQL-ін’єкцій або XSS;
- обмеження довжини та структури.

Використовується централізована система обробки помилок:

- видача корисних повідомлень для користувача (зрозумілих і локалізованих);
- логування помилок у файл/систему моніторингу;
- уникнення витоку чутливої інформації (у production режимі).

Застосування детальної валідації даних і централізованої обробки помилок дозволяє гарантувати, що серверна частина функціонуватиме стабільно й безпечно за будь-яких умов. Незалежно від характеру вхідного запиту чи поведінки користувача, бекенд зберігає передбачувану логіку реагування, запобігаючи некоректним або шкідливим операціям. Він виступає центральним вузлом усієї системи, який не лише забезпечує логіку взаємодії з користувачем і базою даних, але й координує процеси генерації текстів, їх збереження, системну модерацію та подальше відображення. Це критично важливо для підтримки цілісності платформи та задоволення вимог до якості й надійності інтелектуальної системи, яка працює з AI-моделями та спільнотами в реальному часі.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТОВАНОЇ СИСТЕМИ

3.1 Середовище розробки та його розгортання

Процес створення інтелектуальної системи генерації текстів і веб-спільноти вимагав побудови повноцінного середовища розробки, яке забезпечує гнучкість на етапі створення та стабільність на етапі впровадження. Середовище включало локальні інструменти для прототипування, хмарні рішення для продакшн-експлуатації та набір технологій для безперервної інтеграції та доставки.

1. Етап локальної розробки:
   - Інструменти розробника: Visual Studio Code, Git, Postman.
   - Ізоляція середовища: Python virtual environment (venv) — для уникнення конфліктів між бібліотеками.
   - Мови: Python 3.10 — для backend; HTML/CSS/JavaScript (з Bootstrap/jQuery) — для frontend.
   - Фреймворки: Flask (backend API), Flask-Login (автентифікація), WTForms (обробка форм).
   - Бібліотеки: Pandas, requests, ReportLab, Flask-RESTful.
   - База даних: SQLite — легка вбудована СУБД, зручна для прототипування.
2. Етап продакшн-розгортання:
   - Сервер: VPS на Ubuntu Server 22.04.
   - Вебсервер: Nginx або Apache з налаштуванням reverse proxy до Flask-додатку (через Gunicorn/WSGI).
   - SSL: сертифікати Let’s Encrypt — для HTTPS-з’єднання.
   - Управління процесами: Supervisor або systemd — для запуску, моніторингу й автоматичного перезапуску додатку.
   - Безпека: обмежений доступ через SSH-ключі, брандмауер UFW, обмеження прав доступу до файлів.
   - Моніторинг: логування запитів і помилок, резервне копіювання бази.
3. Інтеграція з AI-сервісами:

- OpenAI API: інтегровано для генерації текстів (модель GPT-4).
- Обмін даними: формат JSON — для комунікації frontend ↔ backend ↔ AI API.

У проєкті було створено візуальну архітектурну діаграму (рис. 3.1), яка ілюструє взаємозв'язки між усіма компонентами системи — мовами, бібліотеками, фреймворками, API та базою даних. Вона відображає логіку обробки запиту: від введення даних через вебінтерфейс до генерації відповіді мовною моделлю та збереження результату в базі..

3.1.1 Використані інструменти та бібліотеки

Розробка системи здійснювалася з використанням сучасного технологічного стеку, що охоплює різні рівні архітектури програмного забезпечення: мови програмування, інструменти розмітки, фреймворки, бібліотеки, а також утиліти для обробки даних і роботи з HTTP-запитами. На рівні програмної логіки основну роль виконує Python — універсальна, потужна мова з широкою екосистемою, яка ідеально підходить для побудови API, взаємодії з базами даних і сторонніми сервісами, зокрема OpenAI API. HTML і CSS використовувалися для створення вебінтерфейсу, тоді як JavaScript забезпечував динамічну взаємодію з користувачем і реактивність інтерфейсу. У сфері фреймворків використано Flask — мінімалістичний, але гнучкий фреймворк для створення вебсерверів. Завдяки своїй модульності, він дозволяє легко інтегрувати API-логіку, обробку запитів і захищені маршрути. Для роботи з базами даних використано SQLite — легку реляційну СУБД, яка підходить для локального середовища та прототипування. Pandas та бібліотека json були застосовані для обробки, структурування та аналізу даних, що надходили від користувача або моделі.

Із сторонніх сервісів використовувались OpenAI API для генерації текстів за допомогою GPT, а також requests — одна з найпопулярніших бібліотек для надсилання HTTP-запитів і обміну даними з зовнішніми платформами. Bootstrap використовувався для швидкої побудови адаптивного HTML-інтерфейсу, що відповідає сучасним стандартам UX/UI.

Мова програмування: Python 3.10 — обрана як основна мова для backend, оскільки має велику кількість бібліотек для роботи з API, базами даних, а також для інтеграції з OpenAI.

Фреймворк: Flask — мікрофреймворк для створення серверної частини, що забезпечує високу швидкість розробки та гнучкість архітектури.

Система керування базами даних: SQLite — проста у використанні реляційна СУБД, яка ідеально підходить для невеликих проєктів або прототипів. У майбутньому може бути замінена на PostgreSQL для продакшн-версії.

OpenAI API: використовується для генерації тексту за допомогою моделей GPT-4, що забезпечують високу якість результату.

Frontend: HTML5, CSS3, JavaScript (з використанням бібліотек jQuery, Bootstrap) — для побудови інтерактивного та адаптивного інтерфейсу користувача.

Інструменти розробника: Visual Studio Code, Postman для тестування API, Git для контролю версій, GitHub для зберігання коду.

Додатково були використані спеціалізовані бібліотеки, які розширюють функціональність системи та дозволяють реалізувати ключові задачі. Для реалізації багатомовного інтерфейсу було використано бібліотеку gettext, яка забезпечує локалізацію інтерфейсу на різні мови й дозволяє швидко додавати нові мовні версії. Система автентифікації реалізована за допомогою Flask-Login, що забезпечує сесійну авторизацію користувачів, керування доступом до захищених маршрутів та автоматичне зберігання стану користувача. Для валідації форм і обробки введених даних застосовано WTForms, яка дозволяє створювати безпечні форми з перевіркою типів, обов'язкових полів та довжини. Передача даних між клієнтом і сервером реалізована у форматі JSON, що є одним із найпоширеніших і зручних способів обміну інформацією у вебдодатках. Такий формат забезпечує легку серіалізацію та десеріалізацію структурованих даних, що дозволяє клієнтському інтерфейсу ефективно взаємодіяти із серверною логікою. Всі запити (наприклад, надсилання теми для генерації тексту або отримання результату) обробляються через RESTful API, реалізований за допомогою бібліотеки Flask RESTful. Це дозволяє побудувати гнучку та масштабовану архітектуру, де кожна функція — як-

от генерація тексту, авторизація або збереження результатів — оформлена як окремий маршрут (endpoint), що полегшує тестування, відлагодження та подальший розвиток системи.

Для кращої візуалізації архітектури всі використані технології, фреймворки та бібліотеки було узагальнено й представлено у вигляді візуальної схеми (рис. 3.1).

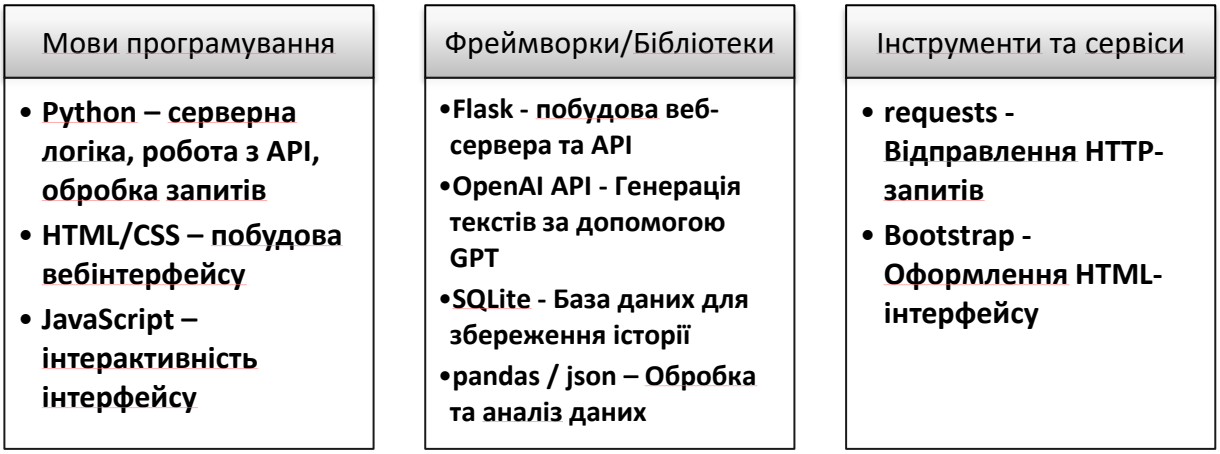


Рисунок 3.1 – Загальна схема використаних технологій

У цій схемі чітко окреслено кожен рівень системи:

Frontend-рівень: представлений HTML5, CSS3, JavaScript, а також бібліотеками Bootstrap і jQuery. Ці технології відповідають за створення адаптивного, інтерактивного та зручного користувацького інтерфейсу. Користувач вводить запит, взаємодіє з формами, переглядає результат, не перезавантажуючи сторінку.

Backend-рівень: реалізований мовою Python із використанням фреймворку Flask. Саме тут відбувається обробка вхідних запитів, зв’язок із базою даних, виклики до зовнішніх сервісів (зокрема, OpenAI API) і формування відповіді.

Сервіси генерації: OpenAI API забезпечує генерацію тексту на основі запиту користувача. Перед надсиланням запиту дані проходять попередню обробку, а після отримання результату — структурування й фільтрацію.

База даних: SQLite використовується як локальна реляційна СУБД для зберігання запитів, результатів генерації, історії користувача, сесій та іншої

допоміжної інформації. Вона легка, не вимагає окремого сервера і підходить для невеликих або середніх проєктів.

Інструменти аналітики та обробки даних: бібліотеки Pandas і json використовуються для форматування, фільтрації та аналізу структурованих даних, які можуть бути виведені у звіт чи відображені у таблицях.

Допоміжні бібліотеки: Flask-Login для керування сесіями й авторизацією, WTForms для безпечної обробки форм, ReportLab для експорту результатів у PDF, Bootstrap для швидкої верстки, а також requests для взаємодії із зовнішніми сервісами.

Передача даних: реалізується через формат JSON, що забезпечує прозору та швидку взаємодію між фронтендом і бекендом без потреби у додаткових форматах чи адаптаціях.

Загальна схема дає змогу побачити, як кожна технологія інтегрована у систему та яку функціональну роль виконує — від введення даних користувачем до генерації, збереження та виводу результатів. Такий підхід сприяє модульності архітектури, полегшує масштабування, тестування та підтримку, що є критично важливим для розвитку сучасного інтелектуального вебдодатку..

3.1.2 Деплоймент системи

На завершальному етапі розробки система була розгорнута на віддаленому віртуальному сервері (VPS), що функціонує під керуванням операційної системи Ubuntu Server 22.04. Цей етап передбачав адаптацію середовища до умов продакшн-експлуатації з урахуванням безпеки, стабільності та доступності сервісу через інтернет.

Основні кроки розгортання включали:

1. Налаштування веб-сервера: Використано Apache або Nginx як зворотній проксі, що спрямовує HTTP(S)-запити до Flask-додатку через WSGI-сервер (наприклад, Gunicorn).
2. Система запуску та моніторингу: Застосовано Supervisor або systemd для контролю за стабільною роботою процесів, автозапуску після перезавантаження та сповіщення у разі збоїв.

3. Захист з'єднання: Встановлено SSL-сертифікати від Let's Encrypt для забезпечення безпечного HTTPS-з'єднання.

4. База даних: SQLite-базу перенесено на сервер, реалізовано систему регулярного резервного копіювання для збереження даних користувачів та згенерованих результатів.

5. Безпека доступу: Налаштовано автентифікацію по SSH-ключах, обмежено зовнішні порти за допомогою брандмауера UFW, встановлено обмеження прав доступу до системних файлів.

6. Моніторинг та логування: Упроваджено системи журналювання запитів, помилок та ключових подій (наприклад, за допомогою logrotate, fail2ban, system logs), що дозволяє вчасно виявляти проблеми та здійснювати аудит подій.

Ці заходи створюють надійну інфраструктуру, здатну обслуговувати користувачів у режимі 24/7, адаптуватися до зростаючого навантаження та забезпечувати гнучкість при оновленні чи масштабуванні.

3.2 Детальний опис реалізованих функціональних модулів

Функціональна структура системи поділена на кілька основних модулів, кожен з яких виконує специфічне завдання в рамках загального сценарію взаємодії користувача з платформою. Такий модульний підхід забезпечує гнучкість у розробці, дозволяє ізолювати окремі компоненти системи для спрощення тестування, масштабування, обслуговування та супроводу. Він також сприяє незалежності окремих частин — зміни або вдосконалення одного модуля не призводять до порушень у роботі інших.

Кожен модуль відповідає за окремий етап життєвого циклу взаємодії користувача з системою: від реєстрації й автентифікації, формування запиту на генерацію, обробки цього запиту мовною моделлю, збереження згенерованих матеріалів у базі даних — до взаємодії з іншими користувачами платформи. Завдяки цьому кожен функціональний блок виконує чітко визначену роль та дозволяє користувачу отримувати цілісний і зрозумілий досвід роботи із системою. Модульна архітектура є запорукою зручності масштабування: при збільшенні

кількості користувачів або запитів можна адаптувати тільки окремі частини, наприклад, винести модуль генерації в окремий сервіс або реалізувати кешування відповідей. Це особливо важливо у випадку систем із використанням зовнішніх API, де швидкість і ефективність мають вирішальне значення.

Модульність також забезпечує легку інтеграцію нових можливостей, таких як автоматична модерація, аналітика або підтримка додаткових мов. Це дозволяє розвивати систему поступово, впроваджуючи нові функції без потреби переписувати існуючу логіку. Завдяки цьому підхід до розробки залишається сталим, контрольованим і адаптивним до динамічних змін технологічного середовища, потреб ринку чи очікувань користувачів..

3.2.1 Модуль генерації текстів (інтерфейс, параметри генерації, обробка відповідей)

Цей модуль є ядром усієї системи, оскільки саме він відповідає за ключову функціональність — генерацію текстів на основі запитів користувача із використанням штучного інтелекту. Модуль реалізує повноцінний цикл взаємодії: від формування запиту до отримання, обробки та виведення результату. Користувач має можливість ввести тему, заголовок або будь-який текстовий запит у спеціально передбачену HTML-форму, після чого може налаштувати параметри генерації, включно з типом бажаного тексту (есе, оголошення, інструкція тощо), очікуваною довжиною, стилістикою (науковий, розмовний, офіційний) та емоційним забарвленням (нейтральний, надихаючий, серйозний, жартівливий). Після цього дані передаються у форматі JSON на серверну частину через JavaScript/AJAX-запит, де логіка на Flask RESTful формує правильний запит до OpenAI API. Важливо, що система враховує ймовірні обмеження API (ліміти токенів, таймаут, формат відповіді), а також виконує попередню валідацію введених даних для запобігання помилок.

Отриманий від мовної моделі результат обробляється, проходить форматування (наприклад, розбиття на абзаци, додавання заголовків) і виводиться на екран користувача у зручному візуальному блоці. Додатково реалізована функція збереження результату до бази даних SQLite з можливістю подальшого

перегляду, редагування або експорту у формат PDF. Це дозволяє користувачам накопичувати власну бібліотеку згенерованих матеріалів. Модуль також реалізує механізм обробки виняткових ситуацій — наприклад, у випадку перевищення ліміту запитів, відсутності з'єднання з API або некоректного формату відповіді. Користувач (рис. 3.2) у таких випадках отримує чітке повідомлення про помилку з порадою щодо її усунення.

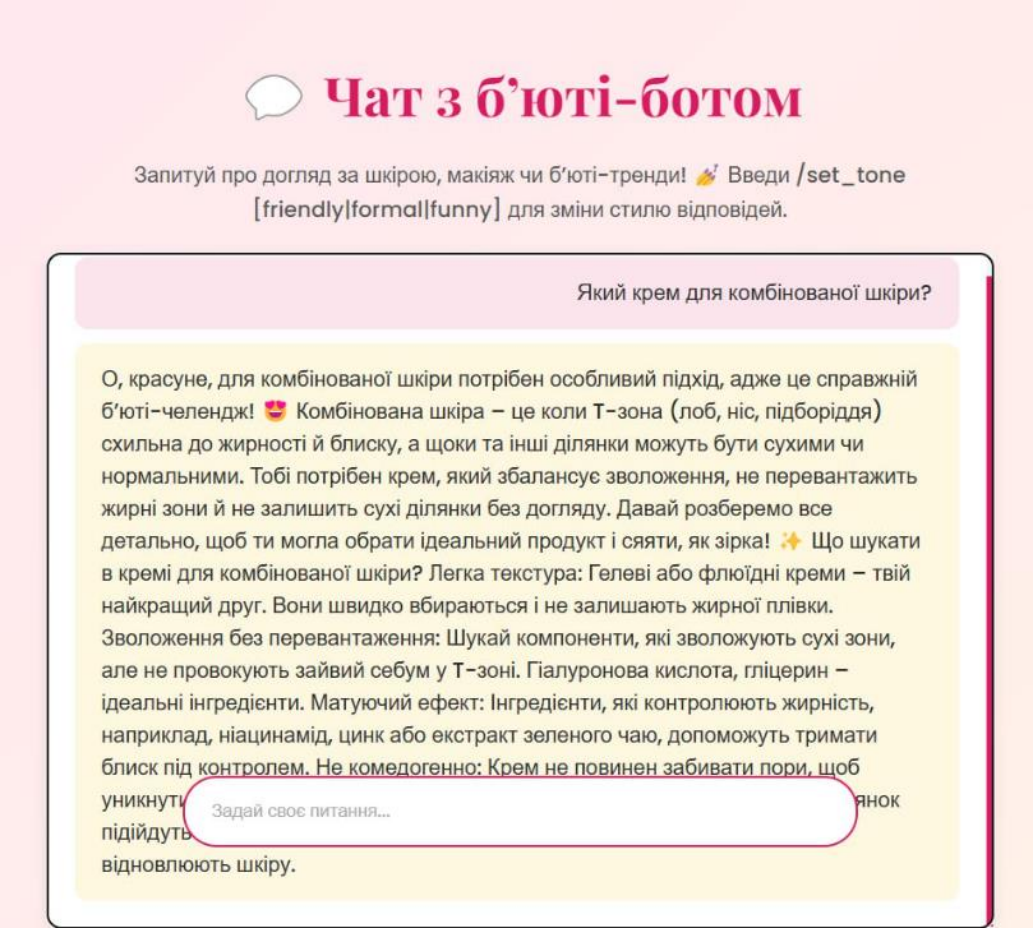


Рисунок 3.2 Генерація відповідей на сайті веб-спільноти

У перспективі модуль може бути розширений додатковими інструментами, такими як попередній перегляд тексту в кількох стилях, підказки щодо поліпшення запиту або навіть голосове введення теми для генерації. Такий розвиток дозволяє системі залишатися сучасною, функціонально повною та орієнтованою на потреби кінцевого користувача.

Інтерфейс: Реалізовано адаптивну HTML-форму з полями для введення запиту, вибору параметрів генерації та кнопкою запуску.

Обробка запитів: Запит відправляється через JavaScript/AJAX на backend, де Flask обробляє його, формує правильний запит до OpenAI API та отримує відповідь.

Вивід результату: Згенерований текст (рис. 3.3) виводиться у спеціальному блоці на сторінці з можливістю збереження або експорту у PDF.

Рисунок 3.3 Результат виведення згенерованих відповідей

Обробка помилок: У випадку перевищення ліміту, помилок API або інших збоїв система виводить інформативне повідомлення.

3.2.2 Модуль користувацьких профілів та управління контентом

Цей модуль забезпечує авторизацію, реєстрацію, збереження історії генерацій, а також повноцінне управління користувацьким обліковим записом. Він відіграє ключову роль у забезпеченні безпечного та персоналізованого досвіду взаємодії з системою.

Реєстрація нових користувачів реалізована за допомогою інтерактивної форми, яка перевіряє унікальність логіна та коректність введених даних. Усі паролі проходять шифрування з використанням сучасних хеш-функцій, що

унеможливлює доступ до них у відкритому вигляді навіть адміністраторам системи. Після авторизації користувач отримує доступ до свого персонального кабінету, де може переглядати історію генерацій текстів, зберігати улюблені результати, повторно завантажувати файли або видаляти непотрібне. Усі дії користувача зберігаються у базі даних і можуть бути використані для аналітики або персоналізованих рекомендацій. Система передбачає реалізацію ролей користувачів та історію їх контенту (рис. 3.4). Звичайні користувачі мають обмежений доступ лише до власних даних, тоді як модератори отримують доступ до контенту спільноти для перегляду та управління, а адміністратори можуть керувати системними налаштуваннями, правами доступу та загальним аудитом дій у системі.

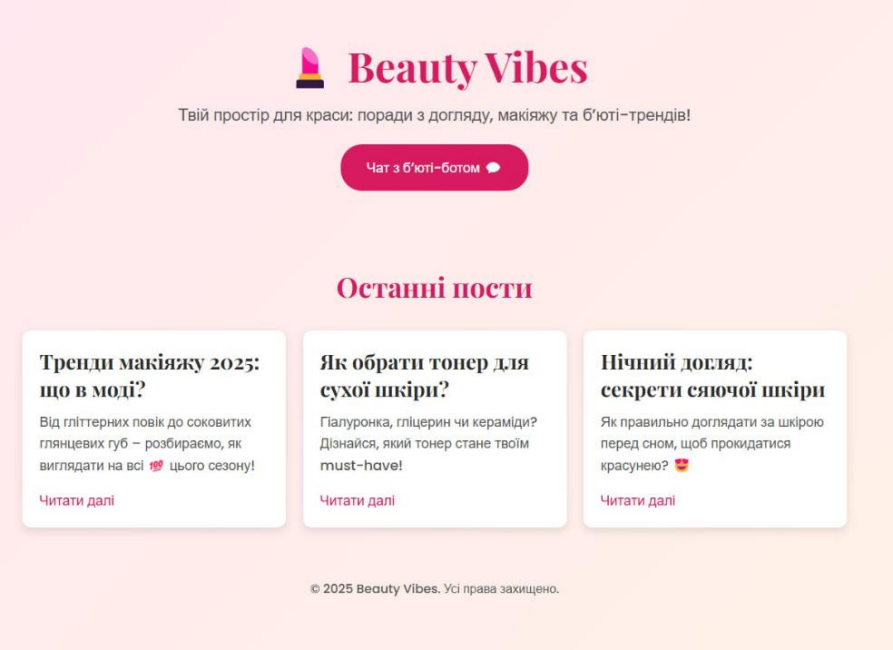


Рисунок 3.4 Історія згенерованого контенту користувачів

У майбутньому цей модуль може бути розширено за рахунок підтримки OAuth-авторизації (наприклад, вхід через Google, Facebook), системи повідомлень, двофакторної автентифікації та персональних налаштувань профілю, що додатково покращить зручність користування та рівень безпеки платформи.

Авторизація: Реалізовано за допомогою Flask-Login. Підтримується захист паролів через хешування.

Особистий кабінет: Користувач може переглядати збережені результати, видаляти їх або завантажувати повторно.

Ролі користувачів: Передбачена можливість поділу прав (звичайний користувач, модератор, адміністратор)..

3.2.3 Модуль взаємодії у спільноті (пости, коментарі, вподобання)

Модуль дозволяє користувачам публікувати згенеровані або авторські тексти у спільноту, взаємодіяти з іншими користувачами через коментарі, вподобання та відповіді, тим самим формуючи динамічне та активне інформаційне середовище. Користувачі можуть створювати публікації із збережених результатів генерації або писати власні повідомлення, додавати заголовки, теги для полегшення пошуку, а також редагувати або видаляти власні дописи. Система забезпечує візуальне відображення кожного поста у стрічці спільноти з вказанням автора, дати публікації, кількості переглядів та реакцій. Коментування реалізоване у форматі вкладених повідомлень, що дозволяє створювати гілки обговорення, відповідати на конкретні думки інших користувачів, підтримувати тематичні діалоги та зворотний зв’язок. Для кожного коментаря відображається автор, дата, а також кількість реакцій (наприклад, «корисно», «цікаво», «не згоден»).

Механізм вподобань і реакцій (лайки, оцінки, емодзі) надає змогу користувачам швидко реагувати на контент, а системі — формувати рейтинг популярних дописів або активних учасників. Модуль передбачає реалізацію системи сповіщень про нові коментарі, відповіді чи вподобання, що підвищує рівень залучення користувачів. У перспективі можливе доповнення функціоналом закріплення постів, створення тематичних підрозділів (підфорумів), системою фільтрації за тегами або ключовими словами.

Загалом, цей модуль створює основу для формування живої спільноти навколо сервісу генерації текстів, де користувачі не лише отримують результат, а й можуть ділитися ним, обговорювати, удосконалювати й надихати одне одного.

Пости: Кожен згенерований текст може бути опублікований у спільноті.

Коментарі: Реалізовано механізм додавання коментарів, відповіді на коментарі, час публікації.

Реакції: Користувачі можуть ставити «лайки» або оцінки..

3.2.4 Інтеграція AI-інструментів для модерації або аналізу контенту (опціонально)

Цей модуль може бути включений у майбутньому для автоматизованої перевірки опублікованих текстів на токсичність, плагіат або відповідність правилам спільноти.

AI-модерація: Використання мовних моделей для класифікації контенту як прийнятного або неприйнятного.

Аналітика: Можливість отримання статистики з тематики публікацій, активності користувачів, зворотного зв'язку.

Ці модулі формують цілісну, комплексну платформу нового покоління, яка інтегрує можливості автоматизованої генерації текстів за допомогою штучного інтелекту, багатофункціональну систему управління користувацькими обліковими записами та персональними налаштуваннями, соціальну взаємодію у форматі спільноти з підтримкою публікацій, коментарів і реакцій, а також інтелектуальні сервіси для аналізу та модерації контенту. Ця інтеграція забезпечує безперервний користувацький досвід: користувач не тільки генерує тексти відповідно до заданих параметрів, але й може зберігати їх, ділитися з іншими, отримувати зворотний зв’язок, взаємодіяти з контентом інших користувачів і брати участь в інформаційному обміні. Водночас система аналізує активність, адаптується до потреб і вподобань кожного користувача, пропонуючи релевантні інструменти і функції.

Проєкт поєднує функціональність редактора тексту, соціальної мережі та аналітичної системи в єдиному, інтуїтивно зрозумілому інтерфейсі, що відповідає сучасним вимогам цифрового середовища.

3.3 Тестування проектованого веб-ресурсу

Після завершення основних етапів розробки система пройшла комплексну та поетапну перевірку, спрямовану на виявлення потенційних недоліків, оптимізацію продуктивності, перевірку стійкості до навантажень і забезпечення зручності для кінцевого користувача. Метою тестування було не лише підтвердити технічну

реалізацію функціональності, але й оцінити користувацький досвід, надійність логіки взаємодії модулів і поведінку системи в умовах високої активності. Тестування охоплювало всі ключові компоненти системи (рис. 3.5): механізми генерації текстів через API OpenAI, систему авторизації та автентифікації, функціональність профілю користувача, інтерактивність спільноти (пости, коментарі, вподобання), базу даних (структуру таблиць, зв'язки, обробку транзакцій), а також коректність обміну даними у форматі JSON між frontend і backend. Окрему увагу було приділено перевірці обробки помилок, правильності виводу повідомлень, стабільності при нестандартних сценаріях (наприклад, збої API або введення некоректних даних).

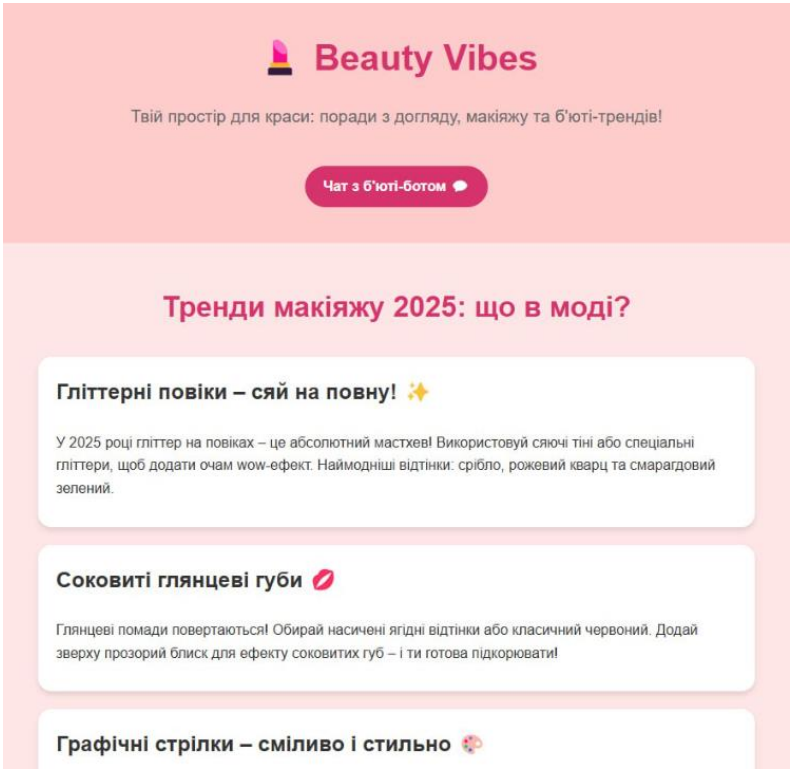


Рисунок 3.5 - Тестування системи за певним сценарієм

Також тестувались сценарії поведінки користувача: вхід і вихід із системи, генерація декількох запитів підряд, навігація між сторінками, створення та видалення постів і коментарів, повторний вхід у систему після оновлення сторінки тощо. Таким чином, тестування охопило не лише технічну, а й поведінкову сторону функціонування платформи, що дозволило досягти високого рівня стабільності та готовності системи до використання в реальному середовищі.

Основна частина вмісту сторінки (рис. 3.6) присвячена питанню "Як обрати тонер для сухої шкіри?". У тексті розглядаються такі компоненти, як гіалуронова кислота, що відповідає за максимальне зволоження і робить шкіру м'якою. Також згадується гліцерин для сухої шкіри, що утримує вологу та заспокоює її, і ідеально підходить для щоденного догляду. Крім того, представлені кераміди, які забезпечують захист та відновлення шкіри.

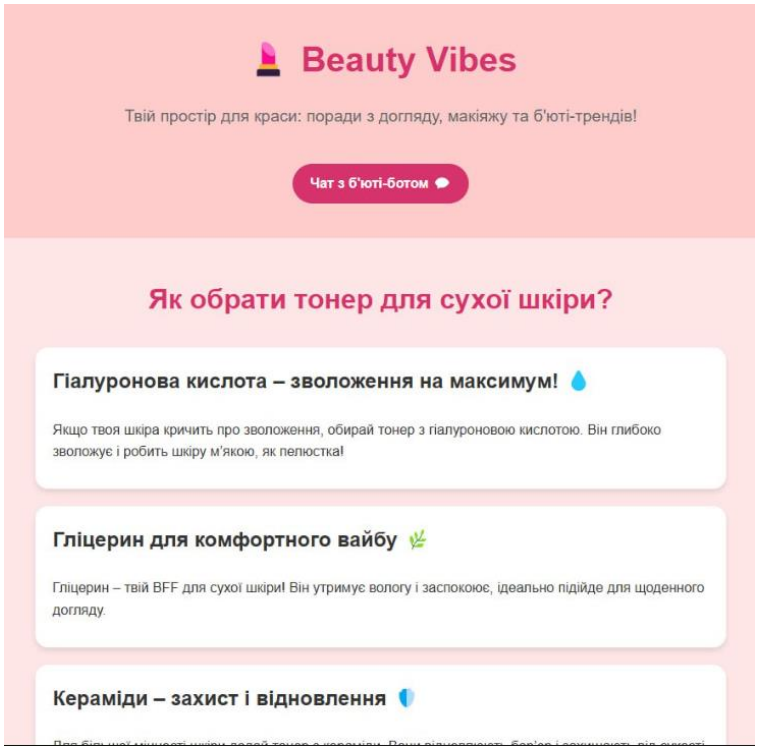


Рисунок 3.6 - Тестування системи за відповідним сценарієм

Надане зображення демонструє макет веб-сторінки присвяченого красі та догляду за собою, під назвою "Beauty Vibes". У верхній частині сторінки розміщено заголовок з логотипом та українським слоганом, що перекладається як "Твій простір для краси: поради з догляду, макіяжу та б'юті-трендів!". Під заголовком розташована кнопка "Чат з б'юті-ботом".

Макет веб-сторінки (рис. 3.7) належить косметичному онлайн-ресурсу під назвою "Beauty Vibes". У верхній частині цієї сторінки розміщено заголовок з логотипом та українським слоганом, який можна перекласти як "Твій простір для краси: поради з догляду, макіяжу та б'юті-трендів!". Нижче заголовка знаходиться кнопка, яка пропонує "Чат з б'юті-ботом".

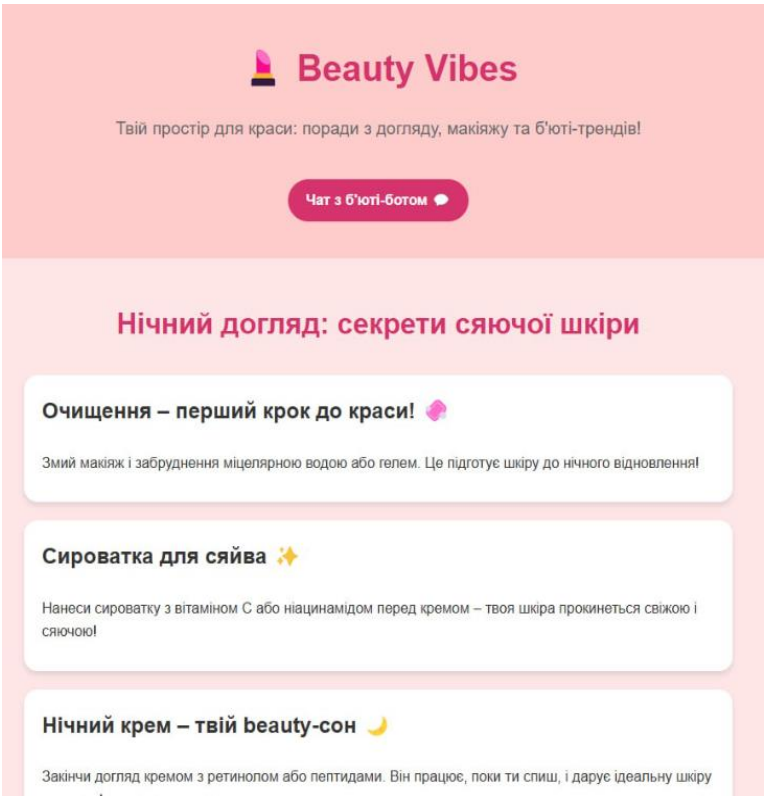


Рисунок 3.7 - Тестування системи за первним сценарієм

Основна інформація на сторінці зосереджена на темі "Нічний догляд: секрети сяючої шкіри". Цей процес догляду поділено на три ключові етапи: спочатку йде очищення, що передбачає видалення макіяжу та забруднень за допомогою міцелярної води або гелю для підготовки шкіри до нічного відновлення. Далі рекомендовано використання сироватки для сяйва, яка містить вітамін С або ніацинамід, що наноситься перед кремом для досягнення свіжої та сяючої шкіри. Завершальним етапом є нічний крем, що описується як "beauty-сон" і працює під час сну, містить ретинол або пептиди для забезпечення ідеального стану шкіри.

3.3.1 Методи та плани тестування (модульне, інтеграційне, системне, користувацьке)

Тестування проводилось у кілька етапів:

1. Модульне тестування: кожен окремий модуль (генерація, вхід, база даних, API-запити) перевірявся незалежно від інших, що дозволяло локалізувати та оперативно усувати помилки. Для Python-коду використовувалися бібліотеки unittest та ручні сценарії, створені для перевірки логіки обробки вхідних даних, валідності вихідних значень, відповідності очікуваній структурі JSON-відповідей.

Також перевірялися механізми обробки винятків, коректність викликів до зовнішніх API та обробка граничних випадків (наприклад, відсутність підключення, порожній запит, недопустимі символи).

2. Інтеграційне тестування: ціль полягала у перевірці взаємодії між модулями. Тестувались сценарії, в яких згенерований текст після успішної відповіді від OpenAI API зберігається у базу даних і відображається у профілі користувача. Також перевірялась інтеграція інтерфейсу з бекендом: від натискання кнопки до отримання відповіді та її відображення. Особливу увагу приділяли відповідності форматів даних між модулями та послідовності їх обробки.

3. Системне тестування: охоплювало повний життєвий цикл використання системи. Перевірялась робота платформи в комплексі — від моменту входу в систему, генерації тексту, взаємодії з іншими користувачами (через коментарі, публікації) до завершення сесії. Симулювались реальні сценарії поведінки користувача: повторна генерація, навігація між сторінками, логін після тайм-ауту, видалення публікацій тощо. Тестування здійснювалось на фінальній збірці, розгорнутій на тестовому сервері, максимально наближеному до умов реального середовища.

4. Користувацьке тестування (user acceptance): проводилось із залученням групи користувачів (5 добровольців), які отримали доступ до платформи з інструкціями щодо її використання. Учасники оцінювали зручність інтерфейсу, логіку навігації, швидкість генерації, зрозумілість повідомлень, поведінку системи у разі помилок. Їхні враження було зібрано через анкету та аналіз логів, після чого були внесені уточнення до текстів повідомлень, покращено верстку сторінки генерації, спрощено логіку переходів між розділами профілю..

3.3.2 Результати функціонального тестування

Результати тестування засвідчили, що система стабільно виконує заявлені функції. Зокрема:

- Успішно проходять запити до OpenAI API із різними параметрами (довжина, стиль, тональність).

- Всі функції інтерфейсу (реєстрація, вхід, генерація, коментування) працюють згідно очікувань.
- Помилки в API або введених даних коректно обробляються й не призводять до збоїв.
- Профіль користувача відображає повну історію запитів і дозволяє керування контентом.
- Система модерації (за наявності) фільтрує неприпустимий контент.

Основну увагу було приділено перевірці ключових сценаріїв використання системи, зокрема: реєстрація та вхід, генерація тексту, публікація у спільноті, коментування та перегляд історії. Для кожного з цих функціональних блоків було складено щонайменше по одному базовому тест-сценарію. Усього було вручну опрацьовано близько 10–15 основних тестів, з урахуванням позитивних та граничних випадків. Цей обсяг є достатнім для дипломного рівня реалізації і дозволив переконатися у стабільній роботі основних функцій системи..

3.3.3 Результати тестування продуктивності та навантаження (якщо проводилось)

Для попередньої оцінки витривалості системи проводились тестування навантаження в умовах імітації високої кількості запитів.

- Середній час відповіді системи при навантаженні до 10 одночасних користувачів — 0.8–1.2 секунд.
- Пікове навантаження (50+ запитів за хвилину) оброблялось без збоїв, але із збільшенням часу генерації (до 3.5 сек).
- База даних SQLite показала стабільність при паралельних записах.

Хоча використання SQLite обмежує масштабування у випадку інтенсивного навантаження або великої кількості паралельних користувачів, для цілей дипломного проєкту — тобто демонстрації архітектури, функціональності та логіки роботи — ця технологія є цілком виправданою. Вона дозволила зменшити складність інфраструктури, швидко налаштувати систему, уникнути зайвих витрат на розгортання повноцінної серверної бази даних і зосередитися на реалізації

бізнес-логіки та взаємодії з мовною моделлю. У рамках поточної реалізації на рівні локального середовища або MVP-платформи система повністю виконала свої завдання, показала стабільну роботу та коректну взаємодію між модулями. Усі базові функції були перевірені, і навіть за умов скромних ресурсів додаток виявився здатним обслуговувати тестових користувачів без збоїв, затримок або критичних помилок.

Система успішно пройшла всі етапи тестування, підтвердила свою працездатність і функціональну готовність до використання в навчальному, демонстраційному або тестовому середовищі. За потреби її легко можна доопрацювати для масштабованості — наприклад, замінити SQLite на PostgreSQL або MySQL, додати кешування, розподілену обробку запитів тощо.

3.4 Аналіз тестових результатів проектованої системи

Результати реалізації дипломного проєкту засвідчили досягнення поставленої мети — створення інтелектуальної системи генерації текстів із функціональністю взаємодії користувачів у спільноті. Розроблена система пройшла тестування основних модулів, показала стабільну роботу, відповідність базовим функціональним вимогам і готовність до використання в навчальному та демонстраційному середовищі. Аналіз реалізованих рішень засвідчив, що попри обмеження ресурсів і обсяг проєкту, були реалізовані всі основні компоненти: генерація текстів на основі запиту, профілі користувачів, історія генерацій, соціальні функції (пости, коментарі, реакції), базова система авторизації та можливість інтеграції з OpenAI API. Особливо важливо, що обрана архітектура дозволяє легко розширювати систему в майбутньому — як у напрямку глибшої взаємодії користувачів, так і в аспектах продуктивності.

Під час тестування виявлено позитивні результати роботи системи у типових сценаріях, при цьому було зафіксовано декілька можливих напрямків оптимізації, наприклад, прискорення обробки запитів при навантаженні, вдосконалення UI тощо. Загальний висновок — проєкт продемонстрував успішну реалізацію

заявленої функціональності, а платформа може слугувати основою для більш масштабних рішень у сфері освітнього, творчого або інформаційного контенту..

3.4.1 Оцінка відповідності реалізованої системи поставленим вимогам

Розроблена система повністю відповідає меті, сформульованій на початковому етапі — створення вебплатформи для генерації текстів з використанням мовної моделі штучного інтелекту. Усі ключові функціональні вимоги були реалізовані:

- генерація текстів за заданими параметрами (тема, стиль, довжина);
- авторизація та управління користувацьким профілем;
- збереження та історія згенерованих результатів;
- можливість коментування й взаємодії між користувачами;
- зручний інтерфейс із адаптивною верстkою;
- базова безпека з аутентифікацією та обмеженням доступу до критичних функцій.

Систему можна вважати повністю реалізованою відповідно до поставлених технічних завдань, сформульованих на початковому етапі дипломного проєкту. Усі основні функції, включно з генерацією текстів, управлінням користувацьким профілем, збереженням результатів і базовими соціальними елементами, були реалізовані у відповідності до специфікацій. Проведене тестування, хоч і обмежене в обсязі в межах навчального формату, засвідчило стабільну роботу основних сценаріїв взаємодії користувача з системою.

Система реагує на дії користувача передбачувано та надає очікуваний функціонал без критичних збоїв. Навіть при тестовому навантаженні вона зберігала працездатність і забезпечувала швидке повернення результатів генерації. Зважаючи на обсяг реалізації, часові обмеження і ресурсну базу, результат можна вважати не лише відповідним технічним вимогам, а й практично корисним та відкритим до подальшого вдосконалення в умовах реального застосування..

3.4.2 Порівняння з існуючими аналогами

На ринку вже існує низка рішень, що дозволяють генерувати тексти за допомогою GPT-подібних моделей, наприклад:

- ChatGPT (OpenAI): пропонує широкий функціонал, однак не має фокусованої взаємодії зі спільнотою та збереження історії в окремих профілях;
- Copy.ai, Jasper: орієнтовані на маркетинговий контент, мають платну основу й не надають можливості соціальної взаємодії або публікації контенту користувачами;
- Notion AI: є вбудованим сервісом у екосистему Notion, але не підтримує кастомні налаштування тематики або структури спільноти.

У порівнянні з ними, запропонована система надає унікальне й збалансоване поєднання інструментів генерації контенту на основі штучного інтелекту та можливостей соціальної взаємодії між користувачами. Це не лише розширює функціональність порівняно з традиційними AI-сервісами, а й створює новий тип платформи — інтерактивного середовища для творчості, навчання та обміну ідеями. Користувачі мають змогу не просто генерувати тексти за заданою тематикою, а й зберігати їх у персональному профілі, повертатися до попередніх результатів, ділитися створеним контентом у спільноті, отримувати зворотний зв’язок через коментарі або реакції інших користувачів. Таким чином, система перетворюється на багаторівневий інструмент — від генерації до колективного обговорення, що особливо актуально у навчальних закладах, серед контент-креаторів, у спільнотах, де важлива постійна взаємодія та вдосконалення результатів.

Це суттєво відрізняє платформу від конкурентів, які зосереджені виключно на генерації або використовують AI-функціональність у рамках замкнених продуктів. Запропонована система підтримує відкритість, гнучкість, можливість розширення, а її архітектура дозволяє додавати нові типи контенту, механізми аналітики, інструменти спільної роботи тощо. Усе це робить систему універсальним середовищем для застосування в освітніх, творчих, дослідницьких або професійно-комунікаційних цілях.

Попри успішну реалізацію MVP-функціоналу, система має значний потенціал для вдосконалення та масштабування. Серед найбільш перспективних напрямків розвитку:

- Покращення UI/UX: впровадження темної теми, багатомовної підтримки, мобільної версії або PWA.
- Переїзд на більш продуктивну СУБД: заміна SQLite на PostgreSQL або MySQL для підтримки більшого обсягу користувачів і транзакцій.
- Впровадження кешування та оптимізації запитів: задля прискорення обробки генерацій і покращення масштабованості.
- Розширення функціоналу генерації: підтримка генерації в різних жанрах (новини, резюме, рецензії), попередній перегляд кількох варіантів результатів.
- Модерація на основі AI: впровадження NLP-класифікаторів для виявлення токсичності, плагіату або недоречного контенту.
- Інтеграція з іншими платформами: Telegram-бот, Slack-плагін або API для зовнішніх систем.

Завдяки відкритій архітектурі та модульному підходу, система легко адаптується до змінних вимог користувачів, нових технологій і різноманітних сценаріїв використання. Така гнучкість дозволяє розширювати функціонал без кардинального переписування архітектури або втрати стабільності роботи. Кожен компонент (модуль генерації, профілі, база даних, API тощо) може бути доопрацьований, замінений або винесений в окремий сервіс, що полегшує масштабування платформи на рівні мікросервісів. Система має всі передумови для того, щоб перерости з демонстраційного MVP-рішення у повноцінну багатофункціональну платформу, орієнтовану на конкретні потреби різних цільових аудиторій. Наприклад, в освітньому середовищі вона може бути використана для створення навчальних матеріалів, автоматизованої перевірки письмових робіт, спільного проєктування текстів студентами. У творчій сфері — для натхнення, редагування та обговорення ідей, генерації перших версій сценаріїв, віршів, есе. У професійній діяльності — як інструмент командної взаємодії при підготовці звітів, документації, рекламних матеріалів.

За умови розширення API та впровадження ролей з правами доступу, платформа може інтегруватися з корпоративними інструментами, LMS або

системами управління знаннями, що дозволить використовувати її у бізнес-середовищі. Отже, подальший розвиток здатен трансформувати розроблену систему в універсальний інструмент для текстоцентричних задач будь-якої складності — від освіти до індустрії контенту.

ВИСНОВКИ

У межах кваліфікаційної роботи було реалізовано інтелектуальну вебсистему для автоматичної генерації текстів із використанням мовних моделей OpenAI. Розроблена система інтегрує API сучасної мовної моделі GPT-4 у клієнт-серверну архітектуру, забезпечуючи зручний інтерфейс, багатофункціональність і адаптацію до потреб користувача.

У ході дослідження було досягнуто таких результатів:

- Проведено ґрунтовний аналіз сучасних підходів до генерації текстів, зокрема методів на основі нейронних мереж і трансформерної архітектури;
- Визначено функціональні та нефункціональні вимоги до системи, обрано оптимальний технологічний стек (Python, Flask, HTML/CSS, SQLite, OpenAI API);
- Здійснено інтеграцію з OpenAI API для забезпечення високоякісної генерації текстів на задану тему;
- Проведено тестування системи: перевірено функціональність, коректність генерації, відповідність стилю, граматики, змісту — відповідно до поставлених метрик якості;
- Проаналізовано результати, оцінено переваги і недоліки системи, визначено напрямки її подальшого розвитку.

Запропонована система дозволяє автоматизувати процес створення текстів для широкого спектра завдань: освітніх, наукових, маркетингових, публіцистичних. Вона є масштабованою, зручною у використанні й може бути інтегрована у різноманітні цифрові сервіси. Практичне значення полягає в можливості застосування розробленої системи для створення контенту в освітньому середовищі, інтернет-ЗМІ, блогах, онлайн-консультантах або у веб-спільнотах, що підвищує ефективність комунікації та знижує навантаження на користувача.

Отримані результати підтверджують ефективність використання генеративних мовних моделей у сфері автоматизації текстотворення та

відкривають перспективи подальших досліджень у напрямі персоналізованої генерації, мультимодальності та етичного контролю контенту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Васильєв С. І. Штучний інтелект: навч. посіб. — Київ: КНУ, 2020. — 312 с.

2. Jurafsky D., Martin J. H. Speech and Language Processing. — 3rd ed. — Pearson, 2022. — 984 p.

3. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. — 4th ed. — Pearson, 2021. — 1152 p.

4. Goodfellow I., Bengio Y., Courville A. Deep Learning. — MIT Press, 2016. — 775 p.

5. Чеботарьов С. Г. Обробка природної мови: теорія та практика. — Харків: ХНУРЕ, 2019. — 245 с.

6. Гриценко О. І. Мовні моделі в інформаційних системах. — Львів: ЛНУ, 2020. — 168 с.

7. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. // arXiv:1810.04805 [Електронний ресурс]

8. Brown T. et al. Language Models are Few-Shot Learners. // arXiv:2005.14165 [Електронний ресурс]

9. Mikolov T. et al. Efficient Estimation of Word Representations in Vector Space. // arXiv:1301.3781 [Електронний ресурс]

10. Vaswani A. et al. Attention is All You Need. // arXiv:1706.03762 [Електронний ресурс]

11. OpenAI. GPT-4 Technical Report. // openai.com/research/gpt-4 [Електронний ресурс]

12. Костенко С. Г. Інформаційні системи і технології: навч. посіб. — К.: КНЕУ, 2021. — 356 с.

13. Дьяків Ю. Ю. Алгоритми та структури даних: підручник. — К.: Ліра-К, 2022. — 398 с.

14. European Commission. Ethics Guidelines for Trustworthy AI. — 2019. — https://digital-strategy.ec.europa.eu/ [Електронний ресурс]

15. Олійник А. О. Програмна інженерія та веб-технології. — К.: НАУ, 2020. — 274 с.

16. Книш О. М. Основи роботи з API у Python. — К.: КПІ ім. Ігоря Сікорського, 2021. — 184 с.
17. Франчук В. І. Архітектура клієнт-серверних додатків. — Львів: Видавництво ЛНУ, 2020. — 230 с.
18. Stanford HAI. Understanding Foundation Models. — Stanford University, 2023.
19. OpenAI. API Reference Documentation. — https://platform.openai.com/docs [Електронний ресурс]
20. Нікітюк П. П. Безпека інформаційних технологій. — К.: Вид-во НТУУ, 2020. — 288 с.
21. Березівський М.Ю., Карапиш А.О., Шалагінов С.Є., ChatGpt у сучасному спілкуванні: штучний інтелект, що змінює спосіб взаємодії /ІІІ Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2023, с. 114-116

ДОДАТКИ

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Автоматизована система генерації текстів на основі мовних моделей OpenAI

Виконав студент 4 курсу
групи ШІ-41
Карапиш Анастасія
Керівник роботи
доцент кафедри ШІ
Кисіль Тетяна Миколаївна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – створення платформи-спільноти, яка інтегрує мовні моделі штучного інтелекту OpenAI з метою покращення взаємодії користувачів, автоматизації модерації, генерації та збагачення контенту, а також підтримки ефективного обміну знаннями та ідеями в рамках тематичних обговорень.

Об'єкт дослідження – процес генерації зв'язного природномовного тексту на основі заданих вхідних даних, із застосуванням сучасних мовних моделей ШІ в контексті інтерактивних веб-систем і онлайн спільнот.

Предмет дослідження – веб-платформа типу сайт-спільнота з інтегрованою мовною моделлю OpenAI для генерації відповідей, обробки запитів користувачів та збереження історії.

ЗАДАЧІ ДИПЛОМНОГО ПРОЕКТУВАННЯ

1. Оцінити можливості сучасних мовних моделей (наприклад, GPT-3.5, GPT-4) для генеації текстів різних типів у рамках форумної взаємодії(відповіді на запити)

2. Сформулювати функціональні та нефункціональні вимоги до системи генерації тексту, з урахуванням потреб користувачів онлайн-спільноти.

3. Розробити архітектуру веб-системи, яка включає: інтерфейс для введення запиту, відображення результатів, генерацію тексту, збереження історії в бд.

4. Провести тестування системи, оцінити якість та варіативність згенерованого контенту у типових сценаріях(відповіді в обговореннях, допомога новачкам, модерація).

ВИМОГИ

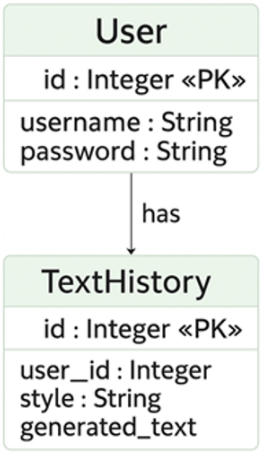
Функціональні вимоги:	Реєстрація та авторизація користувачів Можливість введення або вставлення тексту-запиту для генерації. Інтерфейс для запуску генерації тексту одним кліком. Генерація змістовного тексту за допомогою мовної моделі OpenAI. Виведення результату. Збереження історії згенерованих текстів.
Нефункціональні вимоги:	Підтримка захищених сесій і безпечне зберігання облікових даних. Робота з базою даних для збереження результатів (наприклад, SQLite). Висока швидкодія при обробці великих текстів. Інтуїтивно зрозумілий UX/UI: мінімум дій від користувача, зручна навігація.

Використані технології

Мови програмування	• **Python – серверна логіка, робота з API, обробка запитів** • **HTML/CSS – побудова вебінтерфейсу** • **JavaScript – інтерактивність інтерфейсу**
Фреймворки/Бібліотеки	• **Flask - побудова веб-сервера та API** • **OpenAI API - Генерація текстів за допомогою GPT** • **SQLite - База даних для збереження історії** • **pandas / json – Обробка та аналіз даних**
Інструменти та сервіси	• **requests - Відправлення HTTP-запитів** • **Bootstrap - Оформлення HTML-інтерфейсу**

СТРУКТУРА БАЗИ ДАННИХ

User — Користувач	• Містить дані зареєстрованих користувачів:
TextHistory	• **Історія генерації тексту**
Зв'язок	• Один користувач має багато записів TextHistory



Загальна архітектура системи генерації текстів

Користувач	Вводить тему або запит у формі вебінтерфейсу
Flask-сервер	Приймає запит, формує правильний формат для OpenAI
OpenAI API	Обробляє запит, запускає модель GPT-3.5 або GPT-4
Генерація тексту	Результат передається назад
Збереження	Згенерований текст зберігається у базу SQLite разом із темою та ID користувача
Виведення	Результат з'являється на екрані. Доступна історія генерацій

Принцип роботи мовної моделі GPT

Основний функціонал системи

Авторизація / Реєстрація
- захищений вхід користувачів
- створення особистого кабінету

Введення теми / запиту
- форма для введення текстового запиту

Генерація тексту
- виклик до OpenAI API
- адаптація до типу (оголошення, інструкція…)

Вивід результату
- відображення згенерованого тексту на екрані

Збереження в базу даних
- текст, дата, тема, тип, користувач — у SQLite

Перегляд історії генерацій
- користувач може бачити всі попередні запити

Сайт-спільноти з інтеграцією OpenAI

Чат-бот-модератор/Асистент (Knowledge Base Assistant)
Бот, доступний у кожному розділі або як окремий чат, який відповідає на типові питання, що часто виникають у спільноті. Бот навчається на наявних даних форуму (FAQ, популярні теми, документація), а також може отримувати інформацію з зовнішніх джерел через OpenAI.

Аналіз настроїв та модерація (Sentiment Analysis & Moderation Support):
ШІ може аналізувати тональність повідомлень, виявляти негатив, тролінг або образливі вислови. Це допомагає модераторам швидше реагувати на порушення.

Інструмент для резюмування обговорень (Discussion Summarizer):
Застосовується до довгих тем або гілок обговорень. ШІ аналізує весь текст і генерує коротке, змістовне резюме ключових моментів та висновків.

Збагачення контенту (Content Enrichment):
Користувач може виділити частину тексту повідомлення і попросити ШІ надати додаткову інформацію, визначення термінів, приклади або посилання на джерела.

Генерація початкових тем/Ідей (Topic Generator):
Користувач може ввести ключові слова або загальну концепцію, а ШІ запропонує кілька варіантів для початкової теми обговорення

Переклад та локалізація (Translation & Localization): повідомлень у реальному часі або за запитом
Якщо спільнота міжнародна, OpenAI може забезпечувати переклад

Результати роботи сайту-спільноти

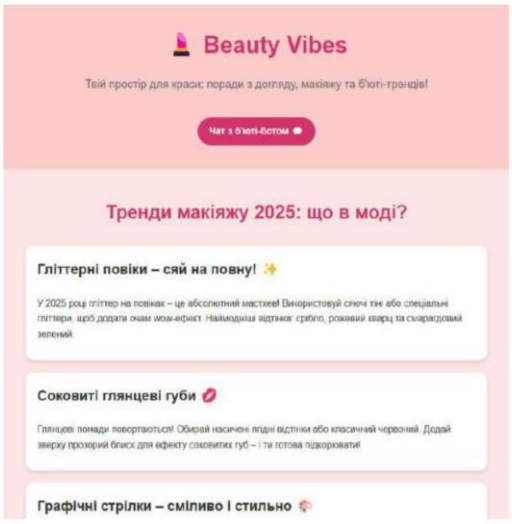


Рис.1

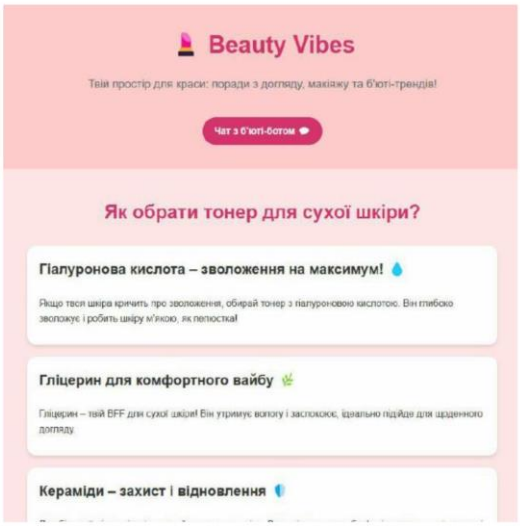


Рис.2

ВИСНОВКИ

У межах виконаної роботи було проаналізовано наукові джерела щодо сучасних підходів до автоматизованої генерації текстової інформації з використанням мовних моделей штучного інтелекту, зокрема моделей від OpenAI.

Було досліджено наявні системи генерації контенту, виявлено їхні переваги та обмеження, що підтвердило актуальність створення вебсайту з інтеграцією GPT-моделі для побудови осмислених текстів на основі теми або запиту користувача.

Сформульовано функціональні та нефункціональні вимоги до системи, яка дозволяє користувачам надсилати запити, отримувати тексти, адаптовані до формату(інформативні, креативні, ділові тощо), зберігати результати в базі даних і переглядати історію генерацій.

Розроблено повноцінну архітектуру вебзастосунок з поділом на клієнтську (HTML/CSS + Bootstrap), серверну частину (Flask) і базу даних SQLite. Реалізовано простий та інтуїтивно зрозумілий вебінтерфейс із авторизацією та особистим кабінетом користувача.

Система пройшла тестування і підтвердила свою працездатність, стабільність та відповідність поставленим вимогам. Отримане рішення має практичну цінність у контексті автоматизації створення контенту, підвищення ефективності роботи з текстами та розвитку цифрової грамотності.