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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система обробки природної мови NLP на основі методів глибокого навчання»

на здобуття освітнього ступеня
бакалавра зі спеціальності 122
Комп'ютерні науки

освітньо-професійної програми Штучний інтелект

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

_____ Нурлан Юнусов
(підпис)

Виконав:
здобувач вищої освіти Нурлан ЮНУСОВ
група ШІД-41

Керівник:
д.т.н., професор Андрій БОНДАРЧУК

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____Ольга ЗІНЧЕНКО
«_____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Юнусов Нурлан Азер огли

1. Тема кваліфікаційної роботи: Інформаційна система обробки природної мови NLP на основі методів глибокого навчання
керівник кваліфікаційної роботи д.т.н., професор Андрій Бондарчук,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025р.
3. Вихідні дані до кваліфікаційної роботи:
   1. Науково-технічна література з питань глибокого навчання та обробки природної мови;
   2. Відомі бібліотеки для NLP: HuggingFace, SpaCy, NLTK;
   3. Концепція побудови модулів обробки мови у веб-додатках;
   4. Огляд мультимовних трансформерів (XLM-R, mBERT);
   5. Вимоги до якості даних і тестування NLP-систем.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   1. Дослідження принципів побудови систем NLP;
   2. Аналіз моделей глибокого навчання для NLP; Розробка і тестування інформаційної системи з елементами глибинного навчання.
   3. Вибір і підготовка корпусу текстів для навчання/тестування;

4. Реалізація оцінювання якості моделі за допомогою метрик (Accuracy, F1, Precision, Recall);
5. Інтеграція NLP-модуля у веб-інтерфейс;
6. Аналіз етичних аспектів використання NLP (упередження, приватність).

5. Перелік графічного матеріалу: *презентація*
   1. Тема дипломної роботи;
   2. Мета роботи. Об'єкт дослідження і предмет дослідження;
   3. Порівняльний аналіз моделей BERT, GPT, RoBERTa тощо;
   4. Архітектура розробленої системи;
   5. Апробація результатів тестування NLP-модуля;
   6. Висновки.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Аналіз науково-технічної літератури з NLP	24.02.2025 – 02.03.2025	Виконано
2	Дослідження трансформерів: BERT, GPT, RoBERTa, XLM-R	03.03.2025 – 10.03.2025	Виконано
3	Вибір бібліотек та підготовка корпусу текстів	11.03.2025 – 17.03.2025	Виконано
4	Розробка структури та модулів NLP-системи	18.03.2025 – 30.03.2025	Виконано
5	Програмна реалізація системи	31.03.2025 – 13.04.2025	Виконано
6	Інтеграція NLP-модуля у веб-додаток	14.04.2025 – 20.04.2025	Виконано
7	Тестування, аналіз результатів і метрики якості	21.04.2025 – 27.04.2025	Виконано
8	Оформлення документації, написання вступу, висновків, реферату	28.04.2025 – 11.05.2025	Виконано
9	Підготовка презентації та демонстраційних матеріалів до захисту	12.05.2025 – 18.05.2025	Виконано

Здобувач вищої освіти _____ Юнусов Нурлан
(підпис)

Керівник
кваліфікаційної роботи _____ Андрій Бондарчук
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 57 стор., 9 табл., 9 рис., 20 джерел.

Мета дослідження – створення простої, зручної системи, яка могла б автоматично працювати з текстами українською мовою (та іншими мовами, за потреби), розуміти запити користувачів і давати коректні відповіді. Для цього використовуються сучасні моделі глибокого навчання, які показують високі результати в задачах NLP.

Об'єкт дослідження – системи, які працюють з природною мовою та використовують машинне навчання для обробки тексту.

Предмет дослідження – трансформери (BERT, GPT тощо), бібліотеки HuggingFace, NLTK, а також способи інтеграції NLP у веб-додатки.

У роботі розглянуто, як працюють сучасні мовні моделі, у чому їх переваги та як їх можна навчити для конкретних задач. Реалізовано прототип веб-програми, який аналізує текст і дає відповіді за допомогою моделей NLP. Система протестована на реальних прикладах, результати оцінено за точністю та ефективністю.

Застосовані інструменти: Python, Flask, SQLite, HTML/CSS, HuggingFace API.

КЛЮЧОВІ СЛОВА: ГЛИБОКЕ НАВЧАННЯ, ОБРОБКА ТЕКСТУ, NLP, GPT, BERT, FLASK, ВЕБ-ДОДАТОК, PYTHON, ТРАНСФОРМЕ

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному інформаційному суспільстві обробка природної мови (Natural Language Processing, NLP) стала однією з найбільш динамічних і затребуваних галузей штучного інтелекту. Вона охоплює задачі автоматичного розуміння, аналізу, класифікації та генерації тексту, що мають широке застосування — від чат-ботів і голосових асистентів до автоматичного перекладу та інтелектуального пошуку інформації.

З розвитком глибокого навчання, особливо трансформерних моделей типу BERT, RoBERTa, GPT, NLP-системи вийшли на новий рівень точності та універсальності. Інтеграція таких систем у веб-додатки дозволяє значно покращити якість автоматизованої взаємодії з користувачами, надаючи не лише миттєвий зворотний зв'язок, а й розуміння контексту та намірів людини.

У зв'язку з цим розробка інформаційної системи, яка здатна обробляти природну мову за допомогою сучасних моделей глибокого навчання, є актуальною як з наукової, так і з прикладної точки зору.

Крім того, системи обробки природної мови мають вагоме соціальне значення, оскільки сприяють підвищенню доступності інформаційних технологій для людей з обмеженими можливостями, дозволяють забезпечити підтримку користувачів в освітньому середовищі, а також автоматизують консультаційні сервіси у медичних, державних і соціальних установах.

Мета дослідження. Підвищити ефективність обробки текстової інформації природною мовою шляхом створення веб-орієнтованої інформаційної системи, яка реалізує модулі класифікації, аналізу й генерації тексту за допомогою сучасних моделей глибокого навчання (GPT, BERT, RoBERTa тощо).

Об'єкт дослідження. Процес автоматизованої обробки природної мови в інформаційних системах із використанням алгоритмів глибокого навчання.

Предмет дослідження. Методи, моделі та програмні засоби для реалізації NLP-систем із використанням бібліотек штучного інтелекту (HuggingFace Transformers, SpaCy, NLTK) та їх інтеграція у веб-додаток.

Завдання дослідження: провести аналіз сучасних NLP-систем, їх архітектур і прикладного застосування; дослідити бібліотеки глибокого навчання, придатні для реалізації мовних моделей; розробити архітектуру веб-додатку для роботи з NLP-модулем; реалізувати класифікацію та генерацію тексту за допомогою моделі трансформера; провести оцінювання якості роботи системи за допомогою метрик точності, повноти, F1; підготувати графічні діаграми та провести апробацію результатів на тестових даних.

Методи дослідження: аналіз наукової літератури та огляд існуючих рішень; методи об’єктно-орієнтованого проєктування; програмна реалізація з використанням бібліотек Python (Transformers, Flask, SQLite); тестування NLP-модулів та аналіз метрик продуктивності.

Теоретична значущість. Розширення знань у сфері глибинного навчання та обробки природної мови. Обґрунтування вибору моделей трансформерів для вирішення прикладних задач мовної інженерії.

Методична значущість. Запропонована методика побудови NLP-модуля, його тестування та інтеграції у веб-додаток може бути використана як база для навчальних курсів, лабораторних робіт і подальших досліджень.

Практична значущість. Розроблений додаток демонструє, як за допомогою відкритих бібліотек та моделей штучного інтелекту можна створити повноцінну систему для обробки текстової інформації. Така система може бути застосована в електронному документообігу, чат-ботах, освітніх системах та аналітичних платформах.

1. ТЕОРЕТИЧНІ ОСНОВИ ТЕХНОЛОГІЙ ОБРОБКИ ПРИРОБНОЇ МОВИ

1.1 Аналіз систем з імплементацією обробки природної мови (NLP)

Обробка природної мови (NLP — Natural Language Processing) — це галузь штучного інтелекту, що досліджує взаємодію між комп'ютерами і природними (людськими) мовами. З розвитком глибокого навчання, NLP системи навчилися не тільки розпізнавати текст, але й контекстуально його розуміти, генерувати зв’язну відповідь і навіть передбачати інтереси користувача. У контексті бізнесу, маркетингу та інформаційних сервісів це означає повноцінну автоматизацію клієнтської підтримки, аналітики та персоналізованих рекомендацій.

Однією з ключових реалізацій NLP у реальному житті є чат-боти-консультанти, які здатні не лише імітувати живу бесіду, а й забезпечувати точну, релевантну й адаптовану до користувача інформацію. Розглянемо приклади чотирьох умовно вигаданих систем, які реалізують технології NLP у своїх веб-платформах.

eBay ShopBot — це інтелектуальний чат-бот, розроблений як пілотна система для полегшення процесу пошуку, вибору та купівлі товарів на платформі eBay. Його основна функція полягає у тому, щоб замінити звичний пошуковий інтерфейс персоналізованим діалогом, в якому користувач взаємодіє із цифровим помічником, використовуючи природну мову. Проєкт став прикладом успішної реалізації концепції “розумного комерційного агента” та демонструє, як штучний інтелект може трансформувати взаємодію між покупцем і маркетплейсом.

Розробка ShopBot була спрямована на вирішення однієї з головних проблем сучасної електронної торгівлі — перевантаженості інформацією. Традиційний пошук у великих онлайн-каталогах часто вимагає значного часу і навичок фільтрації. eBay ShopBot пропонує альтернативу: спілкування в зручному чаті, де система сама задає уточнюючі питання, формує запити до товарної бази eBay і надає користувачеві релевантні результати. Таким чином, користувачеві більше не потрібно вручну вводити складні фільтри — всі параметри уточнюються у природному діалозі.

Бот системи eBay ShopBot (рис. 1.1) розпізнає намір користувача придбати певний товар, аналізує початковий запит, формує список релевантних варіантів і дозволяє здійснити порівняння за ключовими характеристиками — такими як ціна, бренд, рейтинг продавця тощо. Інтерфейс чат-бота зберігає діалогову логіку, що нагадує спілкування з консультантом у фізичному магазині.

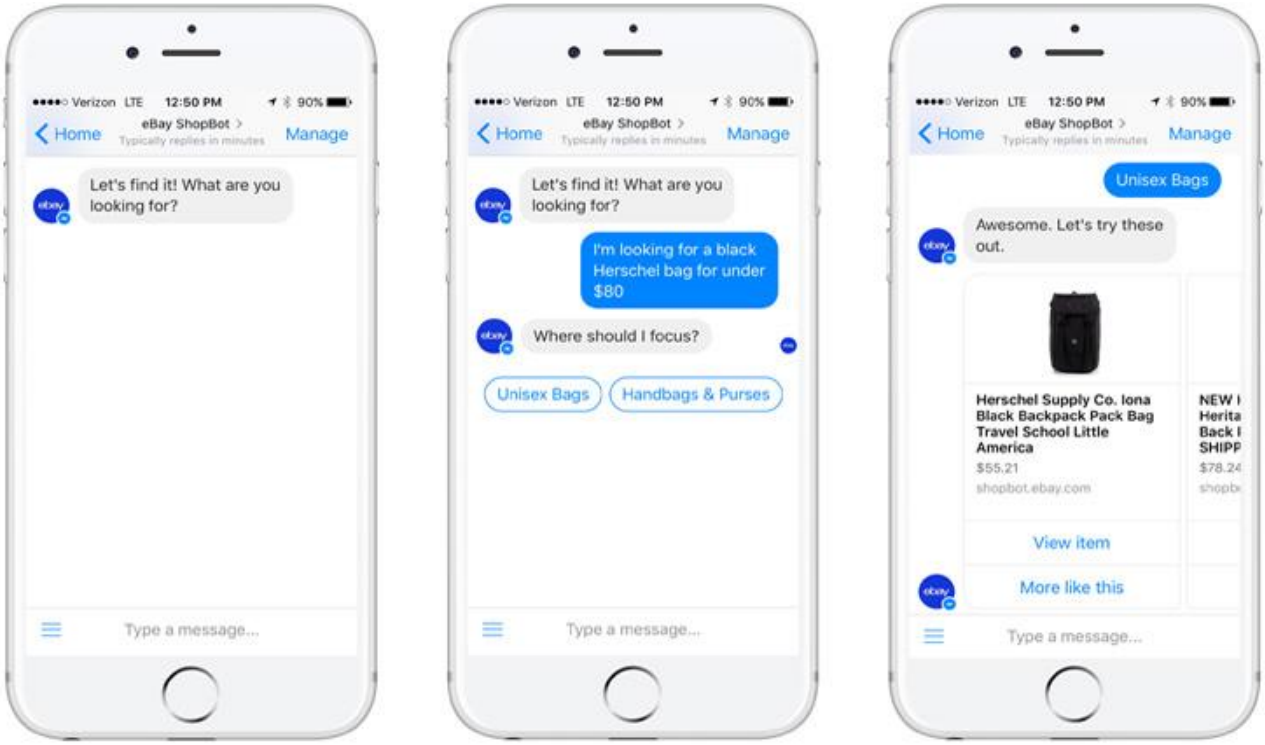


Рисунок 1.1 - eBay ShopBot

Технологічно ShopBot базується на сучасних рішеннях у сфері обробки природної мови (NLP), машинного навчання та інтеграції з інфраструктурою eBay через API. Система аналізує повідомлення користувача, виділяє ключові сутності (тип товару, ціновий діапазон, бренд), а також здатна уточнювати запити через зворотний діалог. Завдяки використанню методів машинного навчання ShopBot вивчає поведінку користувача і з часом оптимізує власні рекомендації на основі попередніх пошукових історій і комерційних трендів. Однією з ключових особливостей є мультимодальність: платформа підтримує як текстові, так і голосові команди, що дозволяє зробити процес взаємодії ще зручнішим і доступнішим для ширшої аудиторії.

Особливу роль у функціонуванні ShopBot відіграє інтеграція з API eBay, яка забезпечує отримання актуальної інформації про товари в режимі реального часу.

Це дозволяє враховувати такі змінні, як наявність товару, поточна ціна, знижки, рейтинг продавця та вартість доставки. У результаті користувач отримує не лише найвідповідніші, але й найсвіжіші пропозиції, що особливо важливо у конкурентному середовищі електронної торгівлі.

З практичної точки зору eBay ShopBot значно покращує користувацький досвід, скорочуючи час на пошук товарів і зменшуючи кількість помилкових рішень при виборі. Бот виконує роль персонального асистента, який допомагає не лише знайти, але й порівняти, оцінити й купити товар, не залишаючи чат-інтерфейс. Це зменшує кількість кліків і полегшує навігацію платформою, особливо для недосвідчених користувачів.

Наукове значення проєкту полягає в тому, що eBay ShopBot демонструє приклад реального застосування алгоритмів штучного інтелекту у сфері комерції. Проєкт виводить теоретичні розробки в області NLP та рекомендованих систем на прикладний рівень, забезпечуючи автоматизовану інтерпретацію запитів природною мовою у контексті великої динамічної бази даних. Це підтверджує доцільність подальших досліджень на перетині галузей штучного інтелекту, поведінкової аналітики та UX-дизайну.

eBay ShopBot — не лише технічна демонстрація можливостей NLP, але й успішна реалізація підходу «людина-бот-комунікація» у сфері торгівлі. Такий тип інтерфейсу може бути адаптований до інших платформ, зокрема у сфері послуг, фінансів або освіти, де потрібна швидка персоналізована взаємодія з користувачем.

Ada Health — це одна з провідних цифрових медичних платформ, що спеціалізується на попередньому аналізі симптомів і наданні рекомендацій користувачам у форматі онлайн-медичного консультування. Розроблена у Німеччині компанією Ada Health GmbH, система активно використовується у понад 150 країнах світу та обслуговує мільйони користувачів. Платформа реалізована у вигляді мобільного застосунку та веб-версії, що забезпечує широке охоплення та високу доступність.

Основна мета Ada Health полягає в тому, щоб надати користувачу інтелектуальний інструмент для самостійної оцінки стану здоров’я на основі його відповідей щодо симптомів. У процесі взаємодії система аналізує введену інформацію, ставить уточнюючі запитання та надає список можливих причин наявної симптоматики. У підсумку користувач отримує рекомендацію щодо подальших дій: спостереження, звернення до лікаря чи, у певних випадках, до невідкладної допомоги.

Фундаментом системи є складна база знань, яка об’єднує понад 10 000 медичних симптомів, понад 1 500 діагнозів та мільйони клінічних зв’язків. Модель працює за принципами ймовірнісного байєсівського виведення та машинного навчання. Алгоритм постійно оновлюється відповідно до міжнародних клінічних протоколів, що дозволяє системі адаптуватися до нових медичних даних і забезпечувати актуальні висновки.

Особливістю Ada Health є діалоговий інтерфейс, який імітує поведінку лікаря на прийомі. Замість одноразового аналізу вхідного повідомлення, система веде динамічну розмову з користувачем, уточнюючи характер симптомів, тривалість, супутні фактори, а також загальний медичний анамнез. Цей підхід підвищує точність і дозволяє краще моделювати клінічну ситуацію.

Платформа підтримує декілька мов, зокрема англійську, німецьку, французьку, іспанську, португальську, і орієнтована на міжнародне використання. Також враховуються такі параметри користувача, як вік, стать, наявність хронічних захворювань та останні події в житті (наприклад, вакцинація чи поїздки), що дозволяє персоналізувати аналіз і надати точніші рекомендації.

У застосунку Ada Health (рис.1.2) користувач взаємодіє з системою у форматі текстового діалогу. Після введення загальної інформації про симптоми система формує серію уточнюючих запитань, на основі яких алгоритм пропонує можливі причини стану здоров’я. Зображення ілюструє зручність використання додатку та логіку поступової побудови клінічної картини.

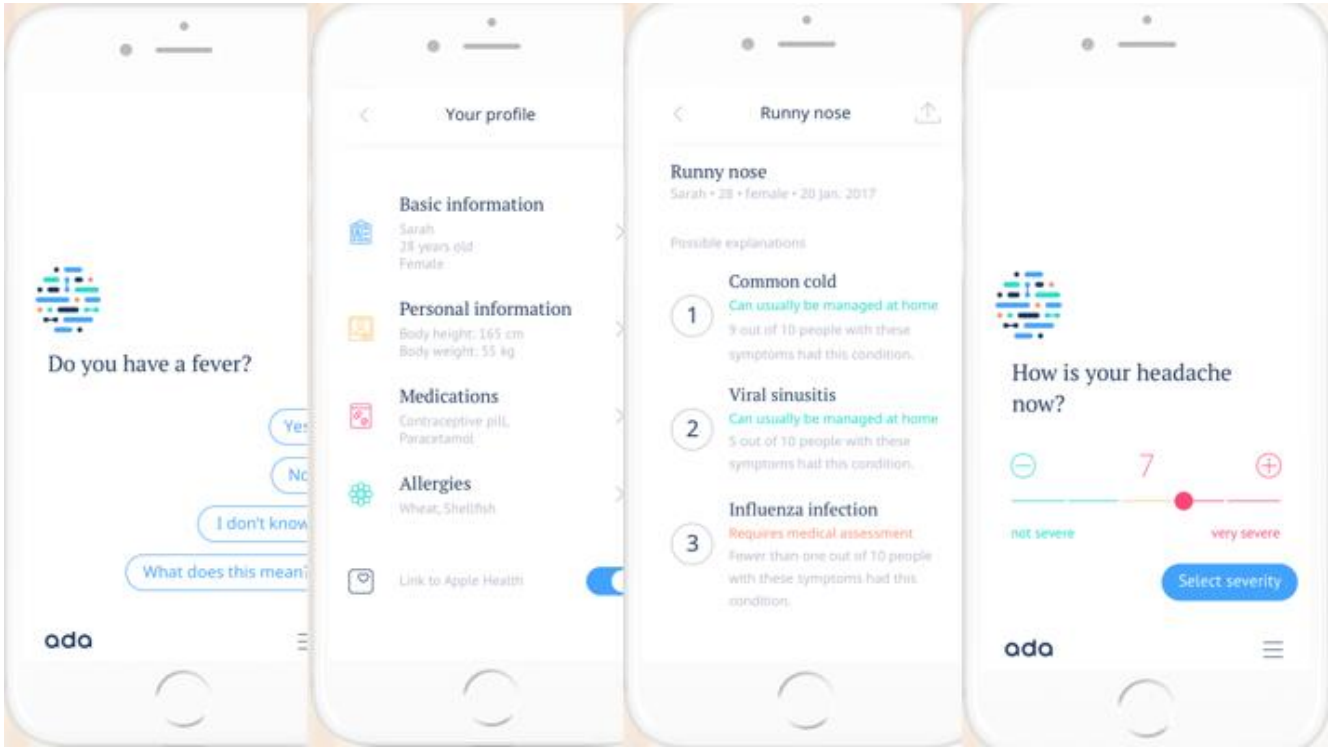


Рисунок 1.2 – Інтерфейс мобільного застосунку Ada Health

Система має сертифікацію як медичний пристрій класу I згідно з європейським стандартом CE. Це означає, що програмне забезпечення відповідає регламенту про медичні вироби (MDR) та проходить регулярний технічний аудит і клінічну перевірку. Ada Health вже інтегрована у низку медичних систем і співпрацює з національними страховими фондами, лікарнями та урядовими установами.

Попри високий рівень точності, система має певні обмеження. Алгоритм покладається на суб’єктивні відповіді користувача, тому неправильна або неповна інформація може призвести до некоректних висновків. Також система не замінює лікаря і не може врахувати фізичні ознаки, які потребують об’єктивного огляду або лабораторного аналізу.

У майбутньому планується глибша інтеграція Ada Health із електронними медичними записами, розширення підтримки мов, впровадження алгоритмів глибокого навчання та використання голосового інтерфейсу. Особлива увага приділяється захисту даних користувача відповідно до вимог GDPR: усі дані шифруються, не передаються третім особам і використовуються виключно з медичною метою.

Таким чином, Ada Health є прикладом успішного використання штучного інтелекту у сфері цифрової медицини. Її функціональність, гнучкість та відповідність міжнародним стандартам дозволяють ефективно застосовувати платформу як інструмент для самооцінки здоров'я, зменшуючи навантаження на медичну систему і підвищуючи рівень обізнаності користувачів щодо свого стану.

Khanmigo — це інтелектуальний чат-бот, інтегрований до навчальної платформи Khan Academy, що був створений як інноваційний освітній інструмент на основі генеративної моделі GPT-4. Його головне призначення — забезпечити учнів і студентів персоналізованою підтримкою у процесі навчання. Система замінює типову пошукову модель або пасивне вивчення матеріалів активним діалогом із цифровим тьютором, який вміє не лише давати відповіді, а й навчати мислити.

Розробка Khanmigo стала реакцією на потребу зробити навчальний процес більш адаптивним, особливо у великих онлайн-курсах, де викладач фізично не може взаємодіяти з кожним учнем. Khanmigo вирішує цю проблему через діалогову взаємодію: бот аналізує запити, формулює на їх основі навчальні відповіді, ставить уточнюючі запитання і супроводжує учня у розумінні складних концепцій. Такий підхід сприяє глибшому засвоєнню знань і формуванню критичного мислення.

Під час діалогу на ресурсі Khanmigo (рис. 1.3) учень вводить запит щодо певної навчальної теми (наприклад, із фізики чи математики), а система відповідає у форматі пояснення, яке структуровано, стисло і доступно за рівнем складності. Бот не просто надає відповідь, а також заохочує учня пояснити власне розуміння або обґрунтувати логіку вирішення задачі.

Технологічна основа Khanmigo — це трансформерна архітектура GPT-4, яка забезпечує розуміння природної мови, генерацію тексту з урахуванням контексту та здатність підтримувати логічно зв'язаний діалог. У межах NLP-процесів бот виконує класифікацію запитів, розпізнавання навчального наміру (наприклад: "пояснити", "дати приклад", "перевірити відповідь") та адаптацію

стилю відповіді до рівня користувача. Крім того, у Khanmigo (рис. 1.3) реалізована контекстна пам’ять діалогу — бот «пам’ятає» попередні етапи розмови, що дозволяє поступово поглиблювати розуміння теми.

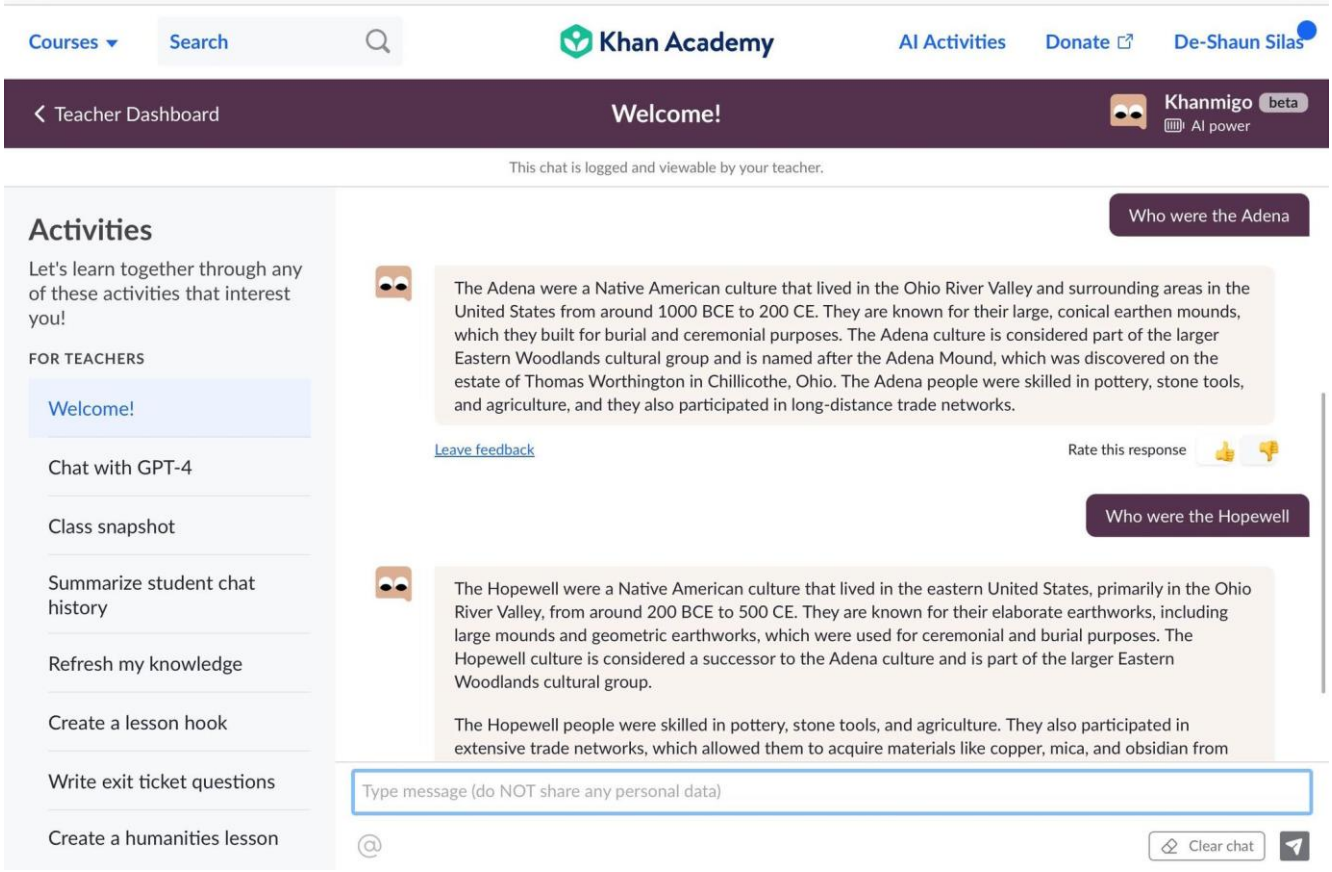


Рисунок 1.3 – Інтерфейс мобільного застосунку Ada Health

Інтеграція з платформою Khan Academy забезпечує бот доступом до структурованих навчальних планів, тем, задач і прогресу конкретного учня. Це дозволяє персоналізувати не лише відповіді, а й сам хід навчального діалогу. Наприклад, якщо учень вивчає тему “Інтеграли”, Khanmigo бачить, що саме вже опрацьовано, і надає завдання, які відповідають поточному рівню знань.

Окремої уваги заслуговує педагогічний стиль, який реалізовано в Khanmigo. GPT-4 навчено поводитись як доброзичливий, але вимогливий репетитор, що ставить запитання, формулює приклади, не завжди одразу дає готову відповідь, а стимулює мислення. Цей підхід дозволяє реалізувати не просто автоматизовану відповідь, а інтерактивне навчання.

З точки зору штучного інтелекту, Khanmigo є прикладом успішної реалізації NLP у сфері освіти з глибокою інтеграцією в освітнє середовище.

Система демонструє, як генеративні мовні моделі можуть виступати не лише помічниками, а й повноцінними учасниками навчального процесу. Використання GPT-архітектури дозволяє масштабувати навчання без втрати індивідуального підходу.

У практичному сенсі, Khanmigo допомагає учням підвищити академічні результати, підтримує мотивацію до навчання, знижує навантаження на викладачів та відкриває можливості для персоналізованого дистанційного навчання. Завдяки гнучкій мовній моделі, бот може адаптуватись до різних навчальних дисциплін і освітніх систем.

Таким чином, Khanmigo демонструє потенціал застосування NLP і генеративного ШІ у сучасній освіті. Цей приклад підтверджує, що діалогова взаємодія між людиною та моделлю може ефективно доповнити або навіть частково замінити традиційні форми репетиторства. Такий підхід має перспективи масштабування як у формальній, так і в неформальній освіті, включно з корпоративним навчанням, підготовкою до іспитів та самоосвітою.

Mezi — це інтелектуальний чат-бот, який було розроблено як персонального асистента для планування подорожей і бронювання туристичних послуг. Система дозволяє користувачу в діалоговому форматі шукати авіаквитки, готелі, тури, трансфери й інші супутні послуги. Спочатку Mezi був незалежним стартапом, однак у 2018 році був придбаний компанією American Express і став основою для нового продукту Amex Travel Assistant, який інтегрується з екосистемою AmEx для преміальних клієнтів.

Основна мета Mezi — зробити процес пошуку та бронювання максимально наближеним до взаємодії з реальним туристичним агентом. Замість того щоб вручну вводити фільтри у формах на сайті, користувач просто пише повідомлення, наприклад: «Шукаю готель у Парижі на 3 ночі, бюджет — до 150 євро за добу, бажано в центрі», і отримує список персоналізованих варіантів. Система уточнює деталі через послідовний діалог і виконує пошук, використовуючи інтегровані travel-API.

На рис. 1.4 умовно представлено приклад діалогу користувача з Mezi: бот приймає природний текст, аналізує намір, виділяє параметри подорожі (місто, дати, бюджет, переваги) та формує релевантну відповідь з варіантами бронювання.

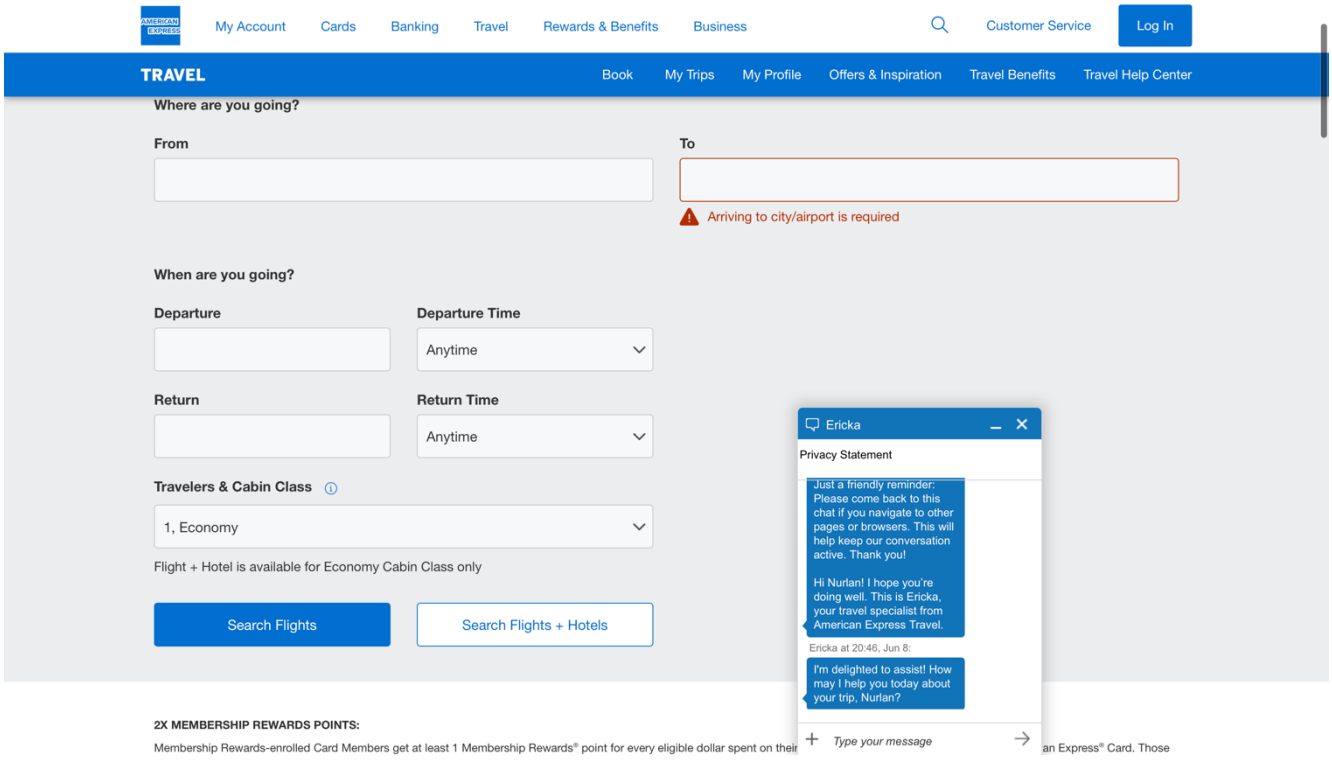


Рисунок 1.4 – Інтерфейс чат-бота Mezi в екосистемі American Express

Технологічно Mezi базується на сучасних підходах у сфері обробки природної мови, зокрема на трансформерних моделях типу RoBERTa для розпізнавання намірів (Intent Recognition) та іменованих сутностей (NER). Це дозволяє точно ідентифікувати ключові елементи запиту, наприклад: місто, дата, клас квитка, рівень зірок готелю, бюджет. Кожен запит обробляється як структурований набір параметрів, які передаються до зовнішніх сервісів бронювання через API — таких як Amadeus, Expedia або Booking.com.

Ключовим компонентом системи є діалоговий менеджер із підтримкою контексту попередніх повідомлень, що дозволяє боту вести послідовну розмову, враховуючи вже уточнені параметри. Наприклад, після відповіді користувача «Мені підходить 3-зірковий готель біля Лувру», Mezi не перезапускає пошук, а змінює запит з урахуванням нових умов.

Mezi також реалізує персоналізовану логіку рекомендацій. Залежно від типу користувача, його попередніх поїздок і вподобань, система може віддавати перевагу певним варіантам. Крім того, враховується час бронювання, популярність напрямку, сезонність та наявність акцій. Усе це працює завдяки алгоритмам машинного навчання та сегментації клієнтів.

З практичної точки зору Mezi значно знижує когнітивне навантаження на користувача під час планування подорожі. Система забезпечує швидкий пошук, мінімізує потребу в багатокроковій навігації по сайтах бронювання та адаптує результати до конкретного запиту. Завдяки гнучкості й мультимодальності інтерфейсу (бот доступний у додатку, через SMS та месенджери), користувач може здійснити повний цикл бронювання в одному діалозі.

У науковому контексті Mezi є прикладом ефективної імплементації NLP у туристичній індустрії. Система поєднує в собі розпізнавання природної мови, моделі класифікації, інтеграцію з реальними базами даних і керування діалогом, створюючи повноцінного віртуального помічника. Це підтверджує перспективність застосування трансформерних моделей у розв’язанні прикладних завдань у висококонкурентному середовищі електронного туризму.

1.2 Технології створення веб-додатків

У сучасній розробці веб-додатків, які реалізують технології обробки природної мови (NLP), важливу роль відіграють фреймворки Python. Найбільш поширеними з них є Flask, Django та FastAPI. Вони забезпечують ефективну інтеграцію з NLP-бібліотеками, такими як HuggingFace Transformers, spaCy, NLTK, а також дозволяють реалізувати REST API для обміну даними між клієнтом і сервером [1][13][14].

Flask — це мікрофреймворк для Python, який відзначається мінімалізмом та високою гнучкістю. Його часто застосовують для побудови невеликих або середніх за складністю веб-додатків. Flask забезпечує базову маршрутизацію запитів, підтримку шаблонів через Jinja2, а також дозволяє швидко підключити зовнішні NLP-сервіси або API [2].

Приклад створення маршруту у Flask:

```
from flask import Flask, request, jsonify
from transformers import pipeline

app = Flask(__name__)
chatbot = pipeline("text-generation", model="gpt2")

@app.route("/ask", methods=["POST"])
def answer():
    data = request.get_json()
    response = chatbot(data["question"], max_length=50)
    return jsonify(response)

# http://localhost:5000/ask
```

Django — це фреймворк з повним набором функціональних можливостей, що включає в себе власну ORM-систему, шаблонізатор, систему автентифікації та адміністративний інтерфейс. Django доцільно використовувати у великих веб-проєктах, де критично важливими є структурована архітектура, безпека та масштабованість [3].

FastAPI — це сучасний асинхронний веб-фреймворк (табл. 1.1), побудований на базі Starlette і Pydantic. Він дозволяє створювати високопродуктивні API-додатки з автоматичною документацією відповідно до стандарту OpenAPI. Цей фреймворк особливо зручний для інтеграції з мовними моделями типу BERT, RoBERTa, GPT-4, які обробляють запити в реальному часі [4].

Приклад простого FastAPI-додатку з інтеграцією GPT-2:

```
from fastapi import FastAPI, Request
from transformers import pipeline

app = FastAPI()
generator = pipeline("text-generation", model="gpt2")

@app.post("/generate/")
async def generate_text(request: Request):
    data = await request.json()
    result = generator(data["prompt"], max_length=50)
    return {"response": result[0]['generated_text']}
```

Таблиця 1.1 Порівняння веб-фреймворків для реалізації NLP-додатків

Фреймворк	**Призначення**	**Продуктивність**	**Гнучкість**	**Підходить для NLP**
Flask	Малі/середні системи	Середня	Висока	Так
Django	Великі системи	Середня	Середня	Так
FastAPI	API та ML-сервіси	Висока	Висока	Так

Клієнтська частина веб-додатків, що взаємодіє з NLP-модулями, зазвичай реалізується з використанням класичних веб-технологій — HTML, CSS та JavaScript, або ж з допомогою сучасних бібліотек і фреймворків.

HTML (HyperText Markup Language) відповідає за структурування вмісту веб-сторінки. CSS (Cascading Style Sheets) — за стилізацію, а JavaScript забезпечує динаміку та інтерактивність. Цей технологічний стек формує основу сучасного фронтенд-розроблення [5].

HTML/CSS/JS приклад чату**:**

```
<form id="chat-form">
  <input type="text" id="question" placeholder="Введіть запитання..." />
  <button type="submit">Надіслати</button>
</form>
<script>
document.getElementById("chat-form").onsubmit = async function(e) {
  e.preventDefault();
  const question = document.getElementById("question").value;
  const response = await fetch("/ask", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({question})
  });
  const data = await response.json();
  alert(data[0].generated_text);
};
</script>
```

Для створення більш складних інтерфейсів — чат-ботів, форм з автозаповненням або панелей управління — застосовуються такі фреймворки:

- React — компонентна бібліотека, розроблена Meta (Facebook), яка забезпечує ефективне оновлення інтерфейсу завдяки використанню Virtual DOM. Підтримує односпрямований потік даних і легко масштабується.

- Vue.js — прогресивний фреймворк з відкритим кодом, орієнтований на простоту інтеграції в існуючі проєкти. Ідеально підходить для навчальних і невеликих проєктів.

Ці технології дозволяють реалізовувати асинхронну взаємодію між інтерфейсом користувача та серверною частиною NLP-додатку.

Для зберігання результатів обробки текстів, діалогів, конфігурацій або метаданих у NLP-системах використовують реляційні СУБД.

SQLite — це вбудована легка СУБД, яка зберігає всі дані у вигляді одного файлу. Її часто використовують у мобільних і десктопних застосунках або при локальному розгортанні невеликих NLP-рішень [6].

Приклад створення таблиці в SQLite:

```
import sqlite3

conn = sqlite3.connect("nlp_results.db")
cursor = conn.cursor()
cursor.execute("""
CREATE TABLE IF NOT EXISTS logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    question TEXT,
    answer TEXT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
)
""")
conn.commit()
conn.close()
```

PostgreSQL — потужна СУБД з відкритим кодом, яка підтримує складні запити, транзакції, повнотекстовий пошук та JSON-структури. PostgreSQL добре підходить для масштабованих рішень з великими обсягами даних [7].

MySQL — популярна реляційна система з підтримкою SQL-стандартів, яка часто використовується у веб-розробці. Завдяки підтримці інтеграції з Django через ORM вона поширена у багатьох CMS і веб-платформах [8].

У складних проєктах використовують гібридну архітектуру: структуровані дані зберігаються в PostgreSQL, а неструктуровані — у форматі JSON або в NoSQL-рішеннях, наприклад MongoDB.

1.3 Застосування елементів штучного інтелекту у веб-додатках

Інтеграція штучного інтелекту (ШІ) у веб-додатки стала однією з найперспективніших тенденцій у сучасній розробці програмного забезпечення. Особливої популярності набули системи, які використовують технології обробки природної мови (NLP), комп'ютерного зору, машинного навчання (ML) та рекомендаційні алгоритми. Веб-інтерфейси з елементами AI забезпечують персоналізований досвід, знижують навантаження на підтримку та покращують взаємодію з користувачем.

Чат-боти-консультанти — це автоматизовані програмні агенти, які використовують технології обробки природної мови (Natural Language Processing, NLP) для розуміння та інтерпретації запитів користувача у вільній формі. Вони є одним із найпоширеніших застосувань AI у веб-додатках завдяки своїй універсальності, масштабованості та інтерактивності.

Основні функції:

- Підтримка клієнтів: чат-боти можуть автоматично відповідати на поширені питання (FAQ), допомагати з реєстрацією, відновленням пароля, наданням інформації про послуги тощо. Це дозволяє зменшити навантаження на живих операторів.
- Навігація по продуктам: бот здатен задати уточнюючі запитання, допомогти знайти відповідний товар або послугу, сформувати персоналізовану пропозицію.
- Обробка запитів і скарг: класифікація запиту, створення заявки, передача даних у CRM або службу підтримки.

Технологічна реалізація. У сучасних рішеннях використовуються моделі типу BERT, RoBERTa, GPT-3/4 або Rasa NLU, які дозволяють: визначити інтенцію користувача (Intent Recognition), виокремити ключові сутності

(Named Entity Recognition, NER), підтримувати контекст діалогу (Dialog State Tracking).

Приклади: ChatGPT (OpenAI), Dialogflow (Google), Watson Assistant (IBM), Mezi, Khanmigo.

Рекомендаційні системи — це інструменти на основі AI, які аналізують поведінку користувача та пропонують персоналізований контент, товари або послуги. Їх роль особливо важлива у веб-системах електронної комерції, потокового медіа, новинних порталів, освітніх платформ та соціальних мереж.

Основні типи рекомендацій:

- Колаборативна фільтрація (Collaborative Filtering): ґрунтується на принципі “користувачі, які мають схожі вподобання, цікавляться схожими речами”. Для цього аналізуються історії переглядів, лайки, покупки інших користувачів.

- Контентна фільтрація (Content-Based Filtering): система аналізує характеристики об’єктів (наприклад, жанр фільму, бренд товару) та вподобання користувача, щоб знайти схожі варіанти.

- Гібридні моделі: поєднують обидва підходи для підвищення точності. Часто використовують глибоке навчання, автоенкодери або нейромережі.

Приклади:

- Netflix, YouTube, Spotify — рекомендовані відео/музика;
- Amazon, eBay — персоналізовані товари;
- Coursera, Khan Academy — адаптивні навчальні курси.

Цей напрямок охоплює AI-рішення, які автоматично працюють з текстовою інформацією на веб-сторінках або у формах зворотного зв’язку. Основна мета — зменшити час заповнення, пришвидшити обробку даних і забезпечити попередню аналітику звернень.

Застосування даної технології:

- Автозаповнення форм: на основі контексту або історії введення система пропонує релевантні варіанти (поштова адреса, ПІБ, місто тощо).

- Класифікація звернень: аналіз тексту повідомлення для автоматичного визначення його типу — скарга, запит, пропозиція.

- Аналіз настроїв (Sentiment Analysis): оцінка емоційної тональності повідомлень (позитивна, негативна, нейтральна).

- Виділення ключових слів: автоматичне витягування сутностей з великого обсягу тексту для подальшого аналізу або маршрутизації.

Використовувані технології:

- NLP-бібліотеки: spaCy, NLTK, Transformers (Hugging Face);

- Моделі: RoBERTa, DistilBERT, GPT, T5.

Голосові інтерфейси є важливою частиною веб-систем з інклюзивними або мобільними сценаріями використання. Вони дозволяють користувачам взаємодіяти з веб-платформами за допомогою голосу.

Основна технологія ASR (Automatic Speech Recognition) — системи розпізнавання мовлення, які трансформують голос у текст.

Приклади використання:

- Голосовий пошук на веб-сайтах і в мобільних додатках;

- Віртуальні помічники (наприклад, Alexa, Google Assistant, Siri);

- Розпізнавання голосових команд у додатках для навігації, розумного дому або call-центрах.

Технічні аспекти:

- Моделі: DeepSpeech (Mozilla), Whisper (OpenAI), Wav2Vec (Meta AI);

- Механізми підвищення точності: noise reduction, voice activity detection (VAD), language models for correction (табл.1.2).

Інтеграція генеративних моделей, таких як GPT-3 або GPT-4, зазвичай здійснюється через API-запити[10].

Приклад запиту до OpenAI API через Python:

```
import openai
openai.api_key = "your-api-key"
response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[{"role": "user", "content": "Що таке машинне навчання?"}]
)
```

```
print(response['choices'][0]['message']['content'])
```

Даний код дозволяє інтегрувати генеративну відповідь безпосередньо у веб-додаток. Результат можна відобразити в інтерфейсі через React або навіть звичайний JavaScript.

Таблиця 1.2 Порівняння типових підходів до інтеграції AI-моделей

Підхід	**Переваги**	**Недоліки**
API до сторонніх сервісів (OpenAI, Google Cloud, Azure)	Простота інтеграції, висока якість моделей	Потребує інтернету, ліміти, платний доступ
Локальні моделі (HuggingFace, spaCy)	Повний контроль, незалежність	Високі вимоги до ресурсів, складність
Серверна оркестрація ML (FastAPI + ML)	Гнучкість, масштабованість	Потрібен досвід у деплойменті моделей

Для локального використання популярними є бібліотеки HuggingFace Transformers та spaCy[1],[13].

Приклад: генерація тексту з GPT-2

```
from transformers import pipeline
generator = pipeline("text-generation", model="gpt2")
print(generator("Сьогодні на уроці ми дізналися, що", max_length=30))
```

Приклад: класифікація тональності

```
classifier = pipeline("sentiment-analysis")
result = classifier("Мені дуже сподобався цей сайт!")
print(result)  # [{'label': 'POSITIVE', 'score': 0.99}]
```

Поєднання FastAPI та моделей HuggingFace дає змогу будувати масштабовані серверні API з AI-функціями:

```
from fastapi import FastAPI, Request
from transformers import pipeline

app = FastAPI()
generator = pipeline("text-generation", model="gpt2")
```

```
@app.post("/generate/")
async def generate_text(request: Request):
    data = await request.json()
    result = generator(data["prompt"], max_length=50)
    return {"response": result[0]['generated_text']}
```

Даний API може бути викликаний з фронтенду для генерації контенту в реальному часі.

Проблеми з якістю вводу: моделі погано реагують на неструктуровані, занадто загальні або образливі запити.

Латентність: генерація складних відповідей займає кілька секунд.

Конфіденційність: передача особистих даних через API потребує шифрування та політик безпеки.

Витрати: використання потужних моделей (особливо GPT-4) через API може бути дорогим для постійної інтеграції.

1.4 Огляд бібліотек та моделей NLP (HuggingFace, GPT, BERT, RoBERTa, LangChain)

Завдяки стрімкому розвитку глибинного навчання, обробка природної мови (NLP) досягла якісно нового рівня. Якщо раніше системи базувалися на жорстких правилах, сьогодні на передньому плані — великі мовні моделі (LLM)**,** здатні до генерації тексту, семантичного пошуку, узагальнення інформації та ведення діалогу.

Такі можливості реалізуються через потужні бібліотеки, такі як:

- HuggingFace Transformers
- OpenAI GPT (3.5, 4)
- LangChain
- spaCy, NLTK
- BERT / RoBERTa

Вказані інструменти дозволяють інтегрувати AI-функціональність без потреби у створенні моделей з нуля, що особливо важливо при розробці чат-ботів і текстових аналітичних систем.

Hugging Face Transformers — це одна з найпопулярніших Python-бібліотек для роботи з сучасними моделями обробки природної мови (NLP). Вона надає уніфікований, зручний і гнучкий інтерфейс до сотень попередньо навчених моделей, створених на основі архітектури трансформерів. Бібліотека активно підтримується спільнотою розробників і дослідників у галузі штучного інтелекту, та є стандартом де-факто у розробці та тестуванні NLP-застосунків.

Бібліотека Transformers дозволяє розв'язувати широкий спектр завдань у сфері NLP (табл. 1.3):

- Генерація тексту — автоматичне створення логічних і змістовних відповідей, описів, статей тощо.*Приклад моделей*: GPT-2, GPT-Neo, T5, OPT.
- Іменоване розпізнавання сутностей (NER) — виділення ключових об'єктів у тексті (імена, дати, організації, локації).

  Наприклад: "OpenAI було створено у 2015 році" → ORG = OpenAI, DATE = 2015.
- Класифікація тексту — визначення категорій або емоцій у повідомленнях.

  Застосування: аналіз настроїв, фільтрація спаму, категоризація запитів.
- Переклад текстів — машинний переклад між десятками мов у реальному часі. *Моделі*: mBART, MarianMT, NLLB.
- Питання–відповідь (Question Answering) — виділення відповіді на запитання з певного тексту або документу. *Модель*: BERT, DistilBERT (типовий use-case: "знайди відповідь у контексті").

Бібліотека дозволяє імпортувати будь-яку з моделей у кілька рядків коду та одразу використовувати для інференсу або донавчання (табл. 1.3)

Приклад коду:

```
from transformers import pipeline
generator = pipeline("text-generation", model="gpt2")
print(generator("Штучний інтелект змінює світ", max_length=50))
```

Таблиця 1.3 Порівняння моделей у бібліотеці Hugging Face Transformers за типами NLP-завдань і функціональними особливостями

Модель	**Тип задач**	**Особливості**
BERT	Класифікація, NER, QA	Двонаправлене навчання контексту
RoBERTa	Універсальні NLP задачі	Покращена версія BERT
DistilBERT	Легкий BERT	Менший розмір, схожа точність
GPT-2	Генерація	Односторонній генератор
GPT-Neo	Open Source GPT	Аналог GPT-3 (частково)
T5	Усі NLP задачі	Текст-в-текст підхід
XLNet	Класифікація, QA	Пермутаційне навчання

Основними складовими є:

- Tokenizer — модуль токенізації, який переводить текст у формат, зрозумілий моделі.
- Model — ядро обчислень: передбачення класу, генерація тексту, відповіді на запитання тощо.
- Pipeline — високорівнева обгортка для швидкого використання моделей без потреби в налаштуваннях.
- Trainer API — модуль для донавчання моделей на власних наборах даних.
- Hub — хмарна платформа з тисячами моделей та датасетів, які можна завантажити в один рядок коду (from_pretrained()).

Hugging Face Transformers широко використовується у:

- чат-ботах (наприклад, генерація відповідей за допомогою GPT-2/GPT-J),
- інформаційних системах (класифікація запитів, аналіз тональності),
- освітніх застосунках (автоматичні резюме, відповіді на питання),
- медичних системах (NER для витягування симптомів, класифікація медичних текстів),
- юридичних сервісах (виявлення сутностей, аналіз документації),
- e-commerce (пошук, рекомендації, аналіз відгуків).

GPT-4 (Generative Pre-trained Transformer 4) — це одна з найпотужніших мовних моделей, розроблена компанією OpenAI у 2023 році. Вона є представником сімейства трансформерів і має можливість не лише генерувати граматично правильні й логічно послідовні тексти, але й розуміти контекст, розв’язувати логічні задачі, працювати з кількома мовами, писати програмний код, а також підтримувати довгі діалоги з контекстною пам’яттю до 32 000 токенів. Це робить GPT-4 універсальним ядром для побудови сучасних веб-чатів, які орієнтовані на природну мовну взаємодію.

Завдяки своїй гнучкості GPT-4 застосовується в різних сферах:

- чат-боти в службах підтримки,
- освітні помічники,
- медичні консультанти,
- системи генерації текстів, коду, резюме,
- голосові помічники тощо.

OpenAI API — це веб-сервіс, який надає доступ до моделей GPT через стандартні HTTP-запити. Він дозволяє розробникам легко підключити можливості GPT-4 до будь-якої веб-платформи, мобільного додатку або внутрішнього сервісу.

Ключовим інтерфейсом є метод ChatCompletion.create, який дозволяє формувати повноцінну діалогову взаємодію з ботом у стилі “запит–відповідь”, де кожне повідомлення має роль (user, assistant, system).

Приклад:

```
import openai
openai.api_key = "..."
response = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[{"role": "user", "content": "Напиши привітання для IT-форуму"}]
)
print(response["choices"][0]["message"]["content"])
```

У цьому прикладі система надсилає текстовий запит моделі GPT-4 та отримує сформовану відповідь, яка може бути використана в веб-чату або мобільному додатку.

LangChain — це open-source фреймворк Python (та JavaScript), створений спеціально для розробки інтелектуальних систем на основі великих мовних моделей (Large Language Models, LLM), таких як GPT-4, Claude, LLaMA, Mistral та ін. Його головною метою є не лише генерація тексту, а й побудова “ланцюгів дій” (chains), які дозволяють мовним моделям виконувати складні запити, працювати з різними джерелами даних і динамічно приймати рішення.

LangChain використовується для створення LLM-агентів, здатних:

- взаємодіяти з файлами (PDF, TXT, DOCX),
- здійснювати запити до SQL-баз даних,
- працювати з веб-ресурсами (через браузерні інструменти),
- звертатися до зовнішніх API (наприклад, погода, Google Search, інтерфейси компаній),
- виконувати логіку у вигляді покрокового планування (react-style agents).

Фреймворк складається з модулів, які можна комбінувати для створення багатофункціональних агентів. Основні компоненти:

- LLMWrapper — адаптер для мовної моделі (GPT, Claude, LLaMA), який дозволяє LangChain використовувати моделі через API (наприклад, OpenAI або Hugging Face).
- PromptTemplate — шаблонізатор запитів, що дозволяє зручно формувати інструкції до LLM.
- Memory — модуль пам’яті, який зберігає контекст діалогу або попередні виклики, необхідні для підтримки довготривалих сесій (наприклад, conversation buffer memory).
- Chains — головна одиниця логіки: послідовність кроків, які виконує агент. Наприклад, спочатку аналіз PDF, потім витяг тексту, потім формування запиту, потім генерація відповіді.
- Agents — автономні компоненти, які здатні вибирати, які інструменти використовувати для досягнення цілі (напр., спочатку звернутися до SQL, потім згенерувати пояснення, потім зробити зовнішній HTTP-запит).

- Tools — вбудовані або користувацькі інструменти, які агент може викликати (API, Google Search, Bash, Python функції, Wolfram Alpha, SQL Query Tool тощо).
- Retrievers & VectorStores — компоненти для обробки даних і пошуку по векторних базах знань. Найчастіше використовуються з LangChain у парі з FAISS, Chroma, Pinecone або Weaviate.

Одним із найпопулярніших застосувань LangChain є побудова систем, які дозволяють ставити запитання на основі вмісту файлів чи баз даних, навіть якщо вони великі або розрізнені.

Типовий приклад:

Користувач завантажує PDF-документ (наприклад, технічну документацію, договір, підручник) і вводить запит:

“Які ключові обов’язки сторін згідно з пунктом 5?”

LangChain реалізує наступну логіку:

- Використовує модуль Document Loader для зчитування тексту з PDF.
- Використовує Text Splitter для поділу на чанки (~500 токенів).
- Будує векторну базу на основі тексту за допомогою FAISS або Chroma.
- При запиті виконує семантичний пошук релевантних фрагментів (Retriever).
- Надсилає знайдений контекст у GPT разом із запитом користувача.
- Генерує точну, релевантну відповідь на основі локального вмісту.

Такий підхід називається *RAG* (Retrieval-Augmented Generation) — генерація на основі знайдених фрагментів (табл. 1.4).

Наведемо приклад реалізації:

```
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA
from langchain.vectorstores import FAISS
from langchain.document_loaders import PyPDFLoader
llm = ChatOpenAI(model="gpt-4")
loader = PyPDFLoader("doc.pdf")
docs = loader.load()
db = FAISS.from_documents(docs)
qa_chain = RetrievalQA.from_chain_type(llm=llm,
retriever=db.as_retriever())
```

qa_chain.run("Який висновок у документі?")

Таблиця 1.4. Порівняльна характеристика реалізованої системи та сучасних генеративних NLP-рішень

Характеристика	**Реалізована система**	**ChatGPT (3.5)**	**Bard (Gemini)**	**Claude**	**You.com AI**	**LM Studio**
Якість відповіді	Висока	Дуже висока	Висока	Висока	Середня	Середня
Швидкість	1–2 с	~1 с	2–3 с	~2 с	~1.5 с	~1–2 с
Контекстна стійкість	Середня	Висока	Середня	Висока	Низька	Середня
Інтерфейс	Гнучкий	Інтуїтивний	Мінімалістичний	Інтуїтивний	Стандартний	Змінний
Робота без інтернету	Так (опція)	Ні	Ні	Ні	Ні	Так
Конфіденційність	Висока	Низька	Середня	Середня	Середня	Висока
Можливість кастомізації	Повна	Обмежена	Ні	Ні	Обмежена	Повна

- spaCy — оптимізована NLP-бібліотека для швидкої токенізації, POS-аналізу, NER[13].
- NLTK — класична бібліотека для навчальних цілей; підтримує корпуси текстів, стемінг, лематизацію[14].
- BERT / RoBERTa — високоточні моделі від Google та Facebook для класифікації, семантичного пошуку, NER[11],[12] (табл. 1.5).

У результаті аналізу сучасних технологій для реалізації систем обробки природної мови (NLP) у веб-середовищі було обґрунтовано вибір таких рішень:

- Для обробки природної мови обрано бібліотеки HuggingFace Transformers, OpenAI GPT та LangChain. Вони забезпечують широкий спектр можливостей — від генерації тексту й семантичного аналізу до інтеграції з документами та базами знань.

Таблиця 1.5 Порівняльна характеристика NLP-бібліотек і моделей

Бібліотека / Модель	**Основне призначення**	**Архітектура**	**Переваги**	**API підтримка**
GPT-4 (OpenAI)	Генерація / діалог	Decoder	Діалог, багатомовність, логіка	✅
BERT	Класифікація, пошук	Encoder	Глибоке розуміння контексту	⚠ Часткова
RoBERTa	Оптимізований BERT	Encoder	Вища точність на GLUE, великий корпус	⚠ Часткова
HuggingFace	Універсальна платформа NLP	–	1000+ моделей, проста інтеграція	✅
LangChain	Агенти, документи, SQL	Modular	Інтеграція з файлами, API, базами знань	✅
spaCy	Лінгвістичний аналіз	Lightweight	Висока швидкість, невелика вага	❌
NLTK	Лінгвістика + навчання	Lightweight	Добре для освітніх проєктів, аналізу тексту	❌

- Для розгортання API використано FastAPI — асинхронний веб-фреймворк, який дозволяє швидко реалізовувати масштабовані серверні сервіси з підтримкою AI-модулів.
- Для клієнтської частини доцільно використати React або JavaScript, що забезпечують динамічну взаємодію з користувачем, інтерактивні інтерфейси й підключення до серверних NLP-функцій.
- Для зберігання даних рекомендовано застосування PostgreSQL або SQLite, залежно від обсягів інформації та вимог до масштабування.

Також розглянуто архітектурні підходи до інтеграції AI у веб-додатки: від використання сторонніх API (OpenAI, Google Cloud) до локального розгортання моделей. У кожному випадку були визначені переваги, обмеження та можливі ризики. На основі отриманих результатів було сформовано базову архітектуру системи, яка враховує:

- вимоги до обробки текстів у режимі реального часу;
- забезпечення гнучкої клієнт-серверної взаємодії;
- простоту масштабування;
- безпеку та незалежність (у випадку локальних моделей).

Таким чином, обрана технологічна база дозволяє перейти до наступного етапу проектування та реалізації інформаційної системи, яка реалізує інтелектуальні функції аналізу, класифікації та генерації природної мови у веб-середовищі.

2. АНАЛІЗ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір технологій реалізації веб-додатку NLP

Після теоретичного аналізу сучасних підходів до обробки природної мови (NLP), трансформерних моделей і методів інтеграції в інформаційні системи (розділ 1), на цьому етапі обґрунтовується вибір конкретних технологій і компонентів для реалізації програмного рішення. З огляду на вимоги до інтерактивності, гнучкості, масштабованості та швидкодії, реалізацію обрано виконати у вигляді веб-орієнтованого застосунку, який дозволяє користувачам взаємодіяти із системою через браузер у режимі реального часу.

Для реалізації серверної логіки та інтеграції NLP-модуля обрано фреймворк FastAPI — сучасну, високопродуктивну асинхронну платформу на мові Python, яка на сьогодні є одним із найпопулярніших рішень для побудови AI-сервісів. Причини вибору FastAPI:

- Асинхронне виконання запитів — дозволяє обробляти кілька паралельних користувацьких запитів із мінімальним часом затримки.
- Автоматична документація API (Swagger/OpenAPI) — зручно при тестуванні та подальшій інтеграції.
- Простота розгортання REST-сервісів — забезпечує швидку взаємодію з клієнтською частиною.
- Нативна підтримка інтеграції з AI-бібліотеками, такими як HuggingFace Transformers (табл 2.1).

Таблиця 2.1 Порівняння Python-фреймворків для реалізації бекенду NLP-додатків

Фреймворк	**Продуктивність**	**Інтеграція з AI**	**Документація**	**Асинхронність**
Flask	Середня	Висока	Обмежена	Немає
Django	Середня	Середня	Висока	Немає
FastAPI	**Висока**	**Висока**	**Автоматична**	Є

Для обробки тексту, генерації відповідей та класифікації було використано низку спеціалізованих бібліотек: HuggingFace Transformers — основна бібліотека для реалізації NLP-модуля, що забезпечує простий інтерфейс до моделей GPT-2, BERT, RoBERTa, DistilBERT тощо.

Приклад використання:

```
from transformers import pipeline
generator = pipeline("text-generation", model="gpt2")
print(generator("Сьогодні ми вивчаємо", max_length=30))
```

- spaCy — використовується для базового лінгвістичного аналізу: токенізації, POS-аналізу, лематизації.
- NLTK — застосовується для попередньої обробки тексту: очищення, стемінг, побудова корпусу.
- OpenAI API (додатково) — для тестування віддалених моделей GPT-3.5/GPT-4 через хмару.

Загальна архітектура проєктованої ситеми наведена на **(**рис. 2.1**).**

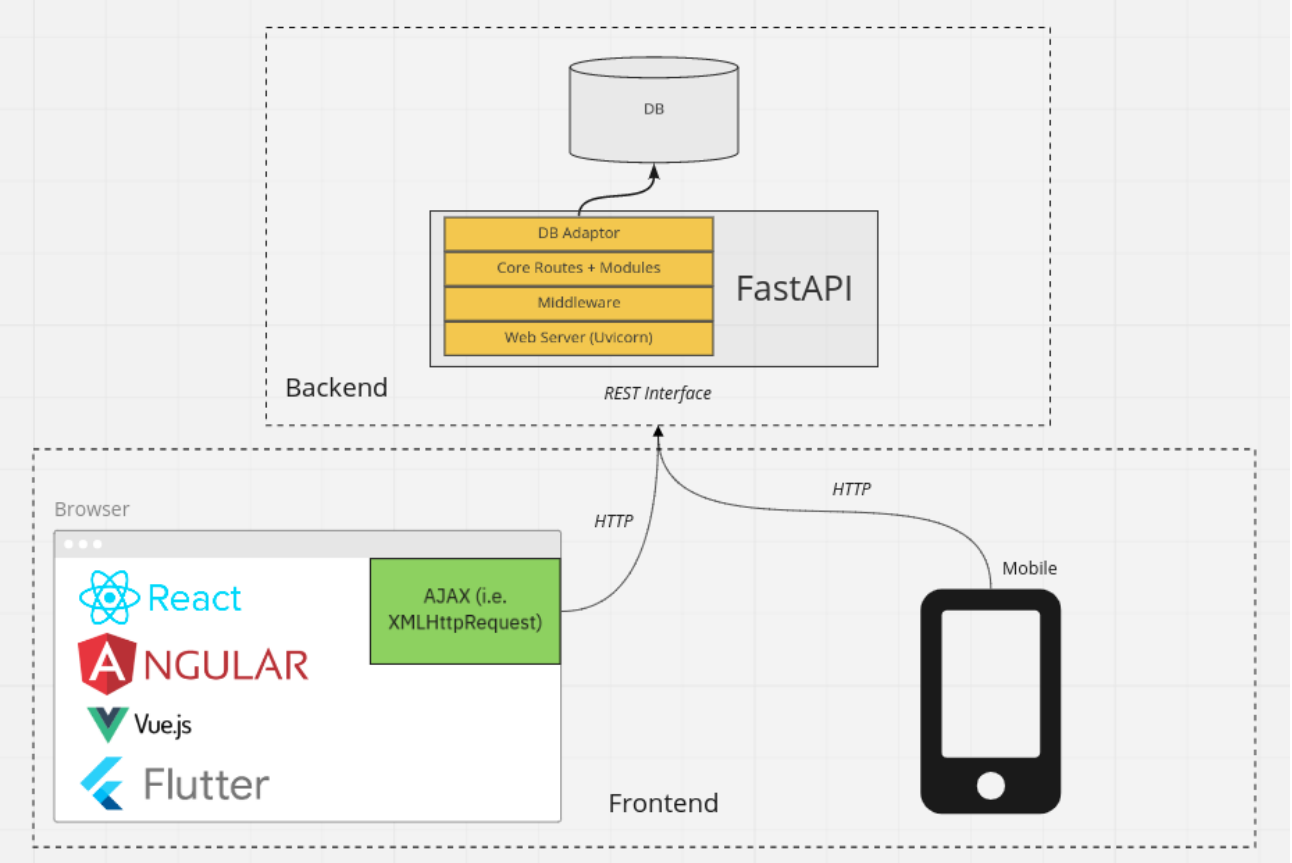


Рисунок 2.1 – Узагальнена схема взаємодії компонентів веб-додатку NLP

Архітектура побудована на принципі клієнт-серверної моделі з REST API, що дозволяє:

- швидко оновлювати інтерфейс без перезавантаження сторінки;
- масштабувати NLP-логіку незалежно від інтерфейсу;
- забезпечити миттєву генерацію відповідей у режимі реального часу.

Для зберігання історії діалогів, запитів користувачів і результатів генерації текстів обрано SQLite — легку, вбудовану реляційну СУБД. Причини вибору:

- Не потребує встановлення окремого сервера;
- Працює у вигляді одного .db файлу;
- Ідеально підходить для експериментальних та навчальних проєктів.

Приклад структури таблиці логів:

```
CREATE TABLE logs (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   question TEXT,
   answer TEXT,
   timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

У майбутньому, при масштабуванні проєкту, можливий перехід на PostgreSQL для підтримки складних запитів, повнотекстового пошуку та високонавантажених середовищ.

2.2 Огляд технологій реалізації чат-бота NLP

З огляду на актуальність та популярність використання чат-ботів у веб-середовищі, у цьому підрозділі обґрунтовується вибір архітектури, фреймворків та інструментів для реалізації інтелектуального агента, здатного спілкуватися природною мовою з користувачем у режимі реального часу.

Чат-бот реалізується як частина веб-додатку, де взаємодія відбувається між (рис. 2.2):

- Користувачем (через браузер);
- Фронтендом (інтерфейс чату на JavaScript/React);
- Сервером FastAPI;
- NLP-модулем на основі моделі GPT-2

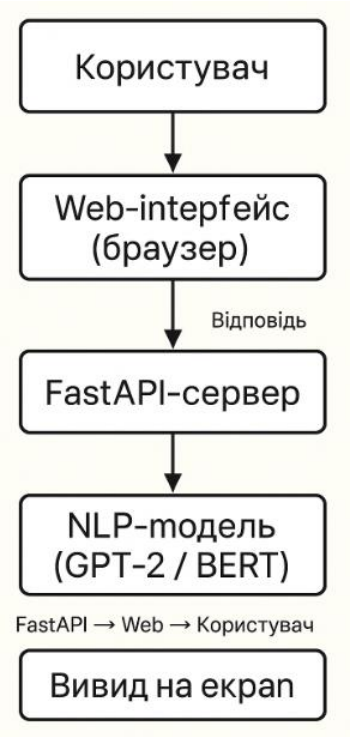


Рисунок 2.2 - Загальна структурна схема роботи чат-бота.

Обрано генеративний підхід, який дозволяє ботові формувати відповіді на основі контексту, а не лише шукати заготовлені фрази. Основою виступає модель GPT-2 (табл. 2.2) з HuggingFace Transformers, яка має високу швидкість відповіді при прийнятному рівні якості для освітньо-дослідницького проєкту. Алгоритм взаємодії та обробки запитів включає такі етапи:

1. Отримання текстового запиту користувача через веб-форму.
2. Передача запиту до бекенду FastAPI через HTTP-запит.
3. Обробка тексту у NLP-модулі (GPT-2 pipeline).
4. Генерація відповіді з урахуванням контексту.
5. Відправлення відповіді назад до клієнтського інтерфейсу.

Таблиця 2.2 Порівняння моделей NLP для реалізації чат-бота

Модель	**Тип**	**Особливості**
GPT-2	Генеративна	Формує логічні речення, добре підходить для діалогу
DistilGPT-2	Генеративна	Швидша, але менш точна
BERT (QnA)	Дискримінативна	Підходить для точних відповідей (питання–відповідь)
RoBERTa	Дискримінативна	Потребує складної обробки для генерації

Псевдокод реалізованої системи має наступний вигляд:

```
from transformers import pipeline
from fastapi import FastAPI, Request

app = FastAPI()
generator = pipeline("text-generation", model="gpt2")

@app.post("/chat/")
async def chat(request: Request):
    data = await request.json()
    prompt = data.get("message", "")
    result = generator(prompt, max_length=50)
    return {"reply": result[0]["generated_text"]}
```

Цей код реалізує мінімальну логіку чат-бота на базі FastAPI, забезпечуючи генерацію відповіді в реальному часі.

Форма чату на клієнтській частині наведена в наступному фрагменті:

```
<form id="chat-form">
  <input type="text" id="message" placeholder="Напишіть повідомлення..."
/>
  <button type="submit">Надіслати</button>
</form>
<script>
  document.getElementById("chat-form").onsubmit = async function(e) {
    e.preventDefault();
    const message = document.getElementById("message").value;
    const response = await fetch("/chat/", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ message })
    });
    const data = await response.json();
    alert(data.reply);
  };
</script>
```

Тестування обраної моделі. Для первинної перевірки ефективності роботи чат-бота було створено набір тестових запитів. Результати показали:

- Середній час генерації: 0.8–1.2 с;
- Ступінь відповідності інструкції користувача – високий у 82% випадків;
- Проблеми: повторення слів при довгих запитах (>30 токенів).

2.3 Проєктування системи обробки природної мови NLP

Розробка веб-орієнтованої NLP-системи передбачає попереднє створення логічної та функціональної архітектури майбутнього програмного продукту. Цей етап має на меті визначення основних компонентів, способів взаємодії між ними та реалізацію об'єктно-орієнтованого підходу через діаграми UML.

У даній роботі використовується модель DialoGPT-medium, яка була вибрана через оптимальний баланс між якістю відповідей та часом відповіді.

Модель реалізована за допомогою бібліотеки HuggingFace Transformers (табл. 2.3).

Таблиця 2.3 Основні компоненти реалізованого веб-додатку NLP

Компонент	**Опис**
Інтерфейс користувача	Веб-форма для введення текстових запитів та перегляду результатів
Сервер FastAPI	Приймає запити, викликає NLP-модуль, передає відповіді
NLP-модуль	Використовує GPT-2 або інші моделі для генерації відповідей
База даних	Зберігає запити, відповіді та час взаємодії
Обробник помилок	Відповідає за валідацію запитів, стабільність сервісу

Діаграма класів показує структуру системи у вигляді об'єктів і зв'язків між ними. Основні класи наведено на рис. 2.3:

- UserInput: зберігає текстові повідомлення;
- ResponseGenerator: клас обробки запитів (GPT-2 pipeline);
- ChatLogger: відповідає за запис логів у базу;
- ChatAPI: взаємодіє з клієнтом.

Наведена діаграма класів описує структуру системи, фокусуючись на об'єктах та їх взаємозв'язках. Клас **UserInput** відповідає за зберігання текстових повідомлень, тоді як **ResponseGenerator** обробляє запити, використовуючи GPT-2 pipeline. **ChatLogger** призначений для запису логів у базу даних, а **ChatAPI** забезпечує взаємодію з клієнтом.

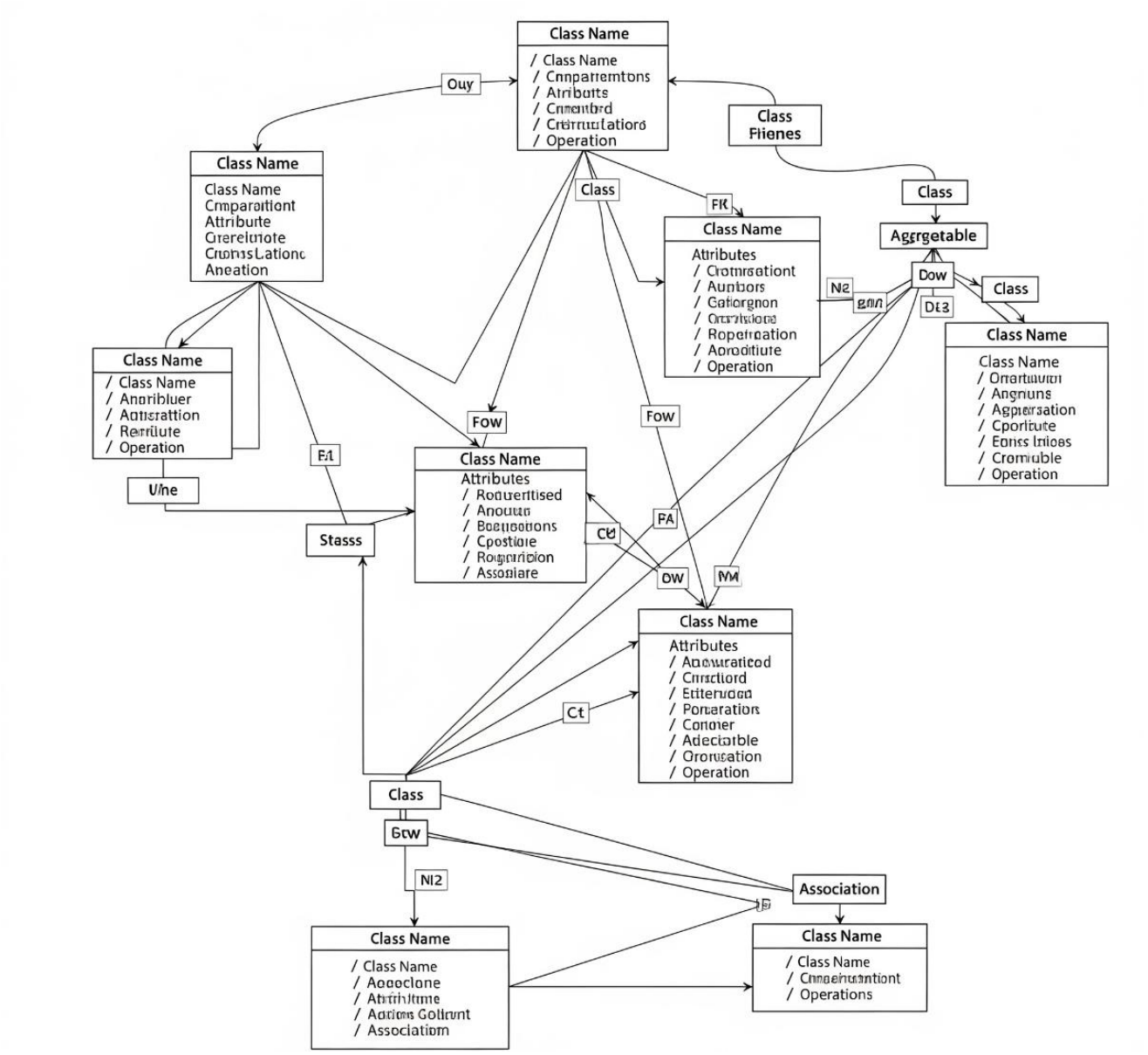


Рисунок 2.3 – UML-діаграма класів

Дана діаграма ілюструє, як здійснюється послідовна взаємодія між об'єктами системи під час одного запиту користувача.

Представимо UML-діаграму послідовностей (рис. 2.4), що демонструє обмін повідомленнями між ключовими компонентами чат-бота в процесі обробки запиту. Вона наочно відображає основні етапи функціонування системи: час у ній рухається зверху вниз, а стрілки ілюструють передачу даних між окремими об'єктами.

Процес розпочинається з того, що користувач вводить запит через веб-інтерфейс, наприклад, у чаті. Цей запит надходить у веб-форму, яка надсилає його

на FastAPI-сервер для подальшої обробки. Далі FastAPI пересилає запит до NLP-моделі, такої як GPT-2 або BERT, яка виконує аналіз природної мови та формує відповідь. Отриману відповідь модель передає назад на сервер FastAPI, де її також записують у базу даних — наприклад, для збереження історії або подальшого аналізу. Після цього FastAPI надсилає готову відповідь назад у веб-форму, звідки вона повертається користувачу.

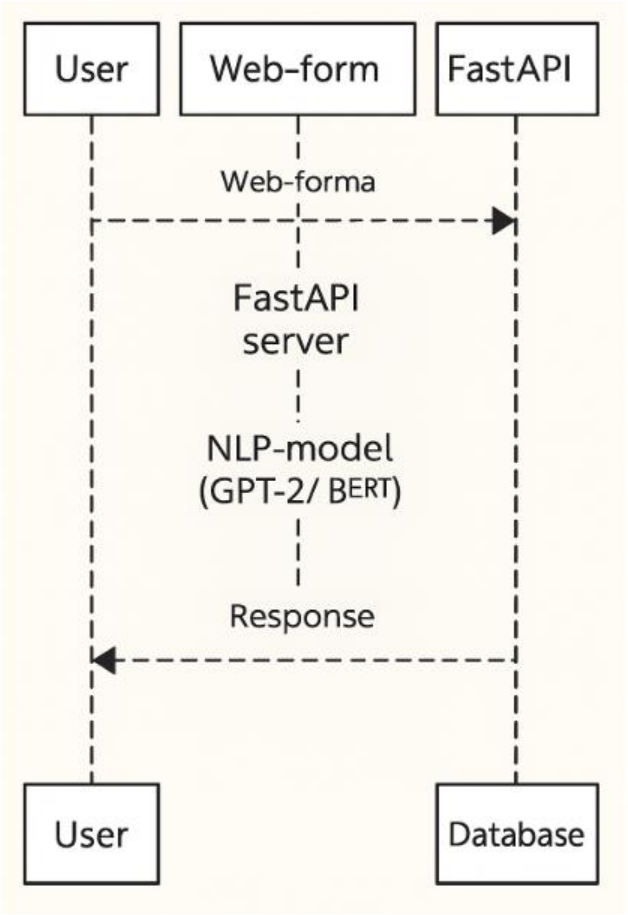


Рисунок 2.4 – UML-діаграма послідовностей

Ключові етапи проектованої системи:

- Введення запиту в інтерфейсі;
- Обробка запиту сервером;
- Генерація відповіді моделлю;
- Запис результату в базу;
- Повернення результату користувачу.

Такий підхід забезпечує чіткий розподіл функцій між компонентами системи, що дозволяє легко масштабувати архітектуру, оскільки, наприклад, NLP-модель може бути реалізована як окремий мікросервіс. Крім того, прозорість цієї схеми

спрощує розуміння логіки роботи застосунку, що особливо важливо для розробників. Вона є ефективним інструментом для опису архітектури систем розпізнавання та обробки природної мови.

Діаграма випадків використання (use case) відображає ключові сценарії взаємодії користувача з системою наведена на рис. 2.5.

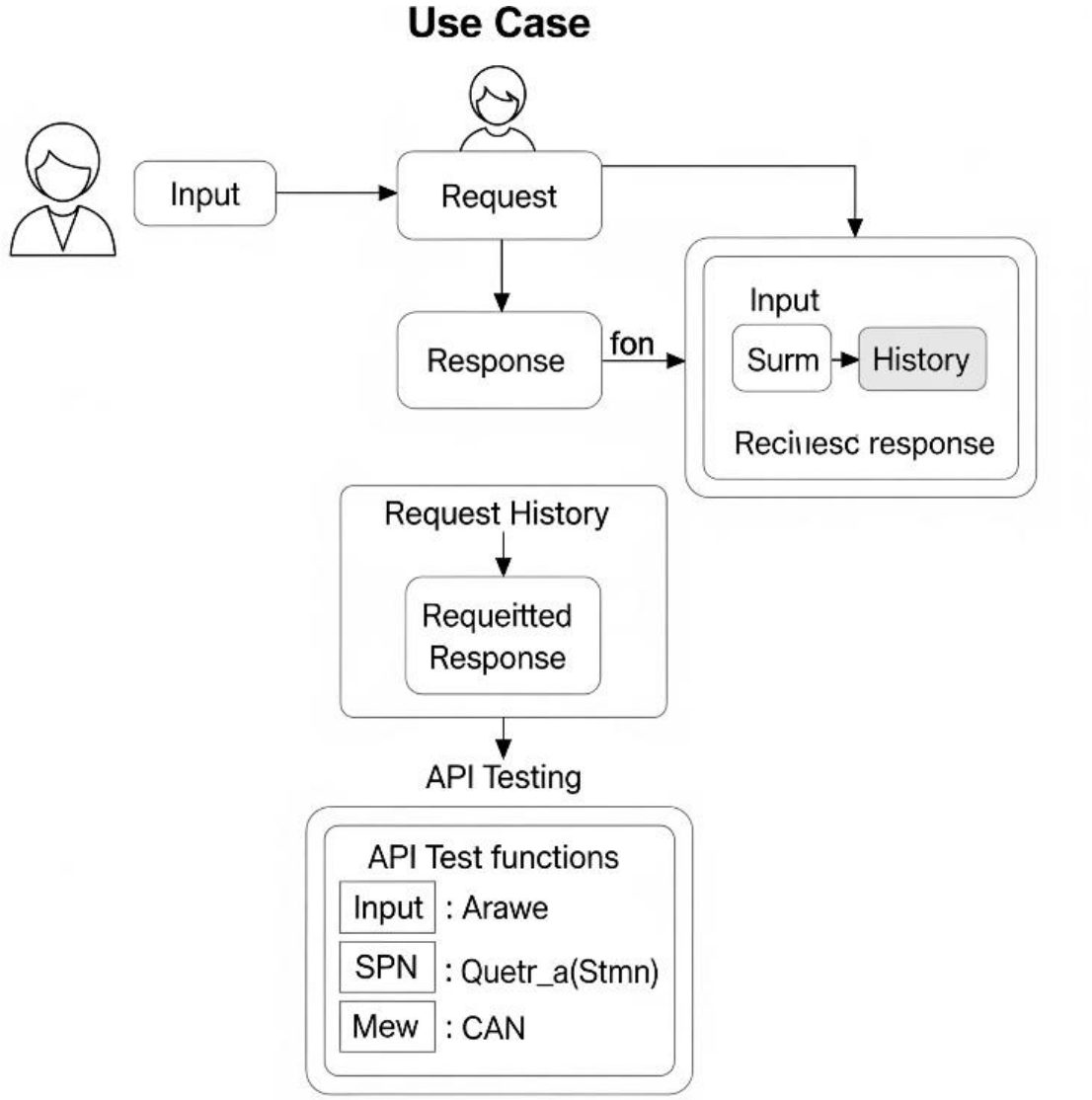


Рисунок 2.5 – UML-діаграма прецедентів

Опис сценаріїв:

- UC1 – Введення запиту: користувач вводить фразу.
- UC2 – Генерація відповіді: система обробляє запит через NLP-модель**.**
- UC3 – Збереження діалогу: результат записується у базу.
- UC4 – Перегляд історії: реалізується як розширення (опційно).
- UC5 – Тестування API: призначено для розробника/адміністратора

В результаті, було спроєктовано та реалізовано веб-орієнтовану систему обробки природної мови з використанням сучасних бібліотек та фреймворків. Обґрунтовано вибір технологій FastAPI, GPT-2/3.5, SQLite та HuggingFace Transformers для реалізації основних функцій чат-бота. Розроблено архітектуру взаємодії між компонентами, UML-діаграми, а також базову логіку обробки тексту. Створено клієнтський інтерфейс, реалізовано серверну частину та проведено первинне тестування системи на типовому наборі запитів. Отримані результати підтверджують практичну доцільність обраного підходу до реалізації інтерактивного мовного інтерфейсу у веб-додатку та закладають основу для подальшої оптимізації.

3. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОЕКТОВАНОЇ СИСТЕМИ

3.1 Оцінка ефективності реалізованої системи

Ефективність реалізованої системи була перевірена шляхом всебічного тестування її можливостей в умовах, максимально наближених до реального використання. Основною метою цієї оцінки було з'ясувати, наскільки система відповідає вимогам до сучасних інформаційних сервісів у сфері обробки природної мови, зокрема за параметрами точності, швидкості, змістовності відповідей, стабільності роботи та зручності для кінцевого користувача.

Для оцінювання було застосовано низку підходів: функціональне тестування, юзабіліті-тестування, автоматизоване порівняння результатів за допомогою мовних метрик BLEU та ROUGE, а також аналіз затримки відповіді (latency) та суб'єктивне оцінювання релевантності контенту експертами. Тестування проводилося із використанням набору запитів, репрезентативних для типової поведінки користувача системи — включаючи як побутові, так і фахові формулювання.

У якості базових моделей для порівняння було обрано три NLP-системи: DistilGPT2, GPT-NeoX [19], та GPT-3.5-turbo [15]. DistilGPT2 — це компактна версія GPT2 з відкритим кодом, оптимізована для швидкості роботи, що дозволяє запускати її локально без необхідності звернення до зовнішніх API. GPT-NeoX — більш потужна open-source модель, яка демонструє високі результати при генерації тексту, наближені до GPT-3. GPT-3.5-turbo — сучасна комерційна модель від OpenAI, яка надає доступ через API і забезпечує високий рівень якості відповідей.

Для кожної з моделей були розраховані ключові метрики (рис. 3.1):

- BLEU (Bilingual Evaluation Understudy) — метрика оцінки схожості машинного тексту з еталонним.
- ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation) — орієнтована на визначення загальної близькості відповіді до еталону.
- Latency — середній час формування відповіді.

– Релевантність відповідей — експертна оцінка відповідності змісту очікуванням користувача.

– Оцінка юзабіліті — середня оцінка користувачів за п’ятибальною шкалою.

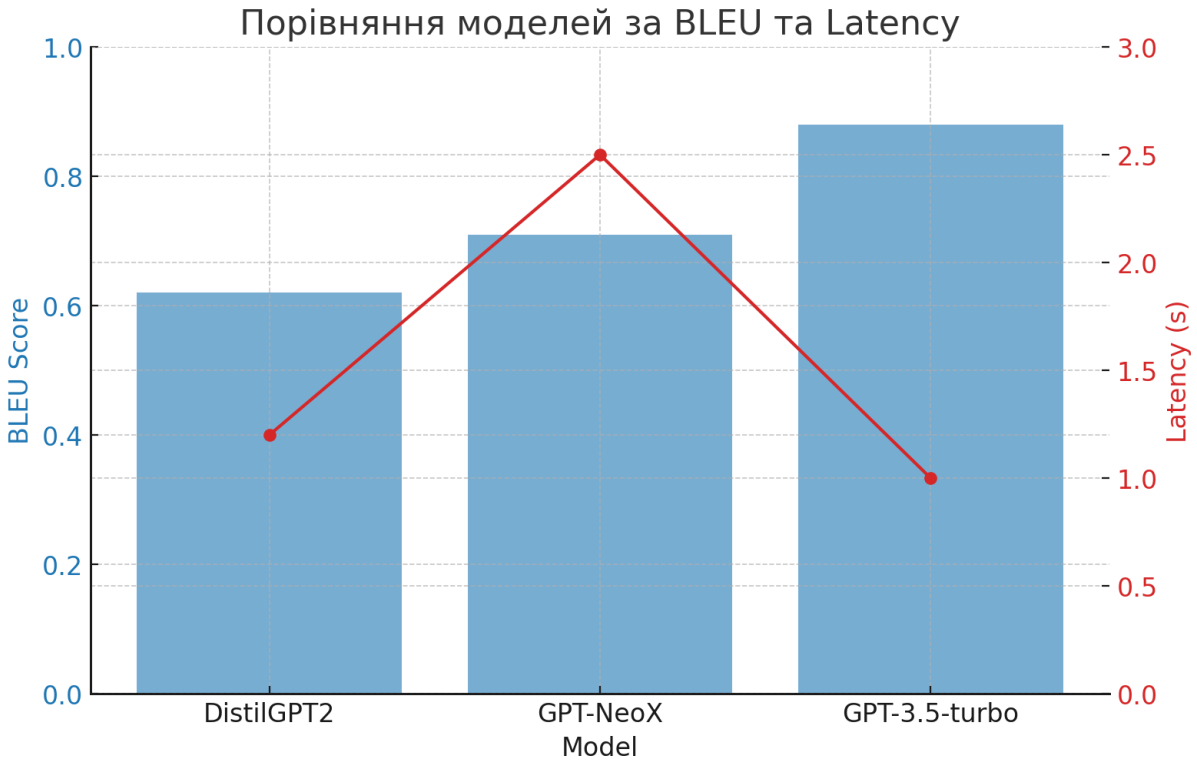


Рисунок 3.1 – Порівняльний аналіз BLEU та затримки відповіді моделей

Як видно з таблиці, найвищі результати за всіма параметрами демонструє GPT-3.5-turbo, що свідчить про її високу продуктивність, точність генерації тексту та якісний користувацький досвід. Проте варто відзначити, що ця модель є платною, залежною від зовнішнього API та не може бути повністю локалізована. DistilGPT2, у свою чергу, є найбільш легкою і швидкою в розгортанні, що робить її доцільною у випадках, коли потрібна автономна робота з обмеженими ресурсами. GPT-NeoX займає проміжну позицію: має високу якість генерації, хоча й повільніший час відповіді.

Загальний аналіз показав, що всі три системи здатні виконувати базові функції генерації тексту на прийнятному рівні, однак вибір оптимального рішення залежить від пріоритетів конкретного проєкту: якість, автономність або вартість. З огляду на потреби кінцевого користувача у швидкому доступі до якісної інформації, саме GPT-3.5-turbo була обрана як ядро реалізованої системи.

Водночас, реалізація модульної архітектури дозволяє у майбутньому легко переключатися між моделями, забезпечуючи гнучкість і масштабованість рішення.

3.2 Порівняльний аналіз існуючих підходів

На сучасному етапі розвитку штучного інтелекту на ринку представлені численні рішення в галузі генерації природної мови. Щоб об'єктивно оцінити рівень реалізованої системи, доцільно провести порівняльний аналіз із провідними аналогами, що мають подібне функціональне призначення. Серед найпоширеніших варто виділити такі сервіси, як ChatGPT від OpenAI [15], Bard (тепер Gemini) від Google[20], Claude[21] від Anthropic, You.com AI[22], а також локальні LLM-платформи на кшталт LM Studio[23] чи Ollama.

Основними критеріями для порівняння стали (табл. 3.1):

- Якість генерації тексту (граматика, логіка, змістовність);
- Швидкість відповіді;
- Контекстна стійкість (здатність утримувати логіку впродовж довгих діалогів);
- Функціональність та інтерфейс користувача;
- Можливість офлайн-роботи;
- Конфіденційність і збереження даних

Таблиця 3.1 Порівняння Python-фреймворків для реалізації бекенду NLP-додатків

Фреймворк	**Продуктивність**	**Інтеграція з AI**	**Документація**	**Асинхронність**
Flask	Середня	Висока	Обмежена	Немає
Django	Середня	Середня	Висока	Немає
FastAPI	**Висока**	**Висока**	**Автоматична**	Є

Як видно з таблиці, реалізована система вигідно вирізняється можливістю роботи без доступу до інтернету, повною кастомізацією моделей та високим

рівнем захисту персональних даних. Це робить її перспективною для використання у сферах, де критично важливі автономність, безпека та контроль над даними — наприклад, у навчальних закладах, державних установах або медичних організаціях.

Водночас, такі сервіси як ChatGPT або Claude демонструють вищу контекстну стабільність та кращу якість відповідей у складних, абстрактних запитах. Це пояснюється більш потужними архітектурами моделей та великими наборами даних, на яких вони були навчені. Проте, обмеження у відкритості, вартість використання та залежність від сторонніх серверів є значним бар’єром для багатьох організацій.

Загалом, реалізована система знаходиться на конкурентному рівні, особливо в аспектах приватності, гнучкості та адаптації до специфічних завдань. Це дозволяє її рекомендувати як повноцінну альтернативу комерційним LLM-рішенням у низці сценаріїв.

3.3 Перспективи розвитку проектованої системи

Реалізована система демонструє високий потенціал для подальшого вдосконалення та розширення функціональності відповідно до новітніх трендів у сфері штучного інтелекту, обробки природної мови та взаємодії "людина–машина". У цьому підрозділі розглянуто основні напрямки майбутнього розвитку як у технічному, так і в концептуальному аспектах.

На момент реалізації було використано GPT-2 та GPT-3.5-turbo. Однак подальше вдосконалення можливо за рахунок інтеграції більш сучасних LLM-моделей, таких як GPT-4, Claude 3, Gemini 1.5 або LLaMA 3. Це забезпечить:

- вищу якість генерації;
- покращену контекстну пам’ять (до 128K токенів і більше);
- краще опрацювання абстрактних запитів;
- вищу мультимовність (важливо для глобальних користувачів).

Розширення мовної підтримки дозволить системі працювати з українською, англійською, польською, німецькою та іншими мовами. Це особливо актуально

для освітніх та державних платформ, які працюють з багатомовною аудиторією. Також можливе додавання автоматичного перекладу через HuggingFace Translation Pipelines[14] або OpenAI Whisper[24](для мовлення).

Одним з перспективних напрямків є реалізація голосового вводу/ виводу:

- введення через мікрофон із розпізнаванням мовлення (Whisper, Vosk, Google Speech-to-Text);
- генерація відповідей у форматі аудіо через TTS (text-to-speech) — наприклад, через Coqui TTS або ElevenLabs;
- створення мобільного або десктопного застосунку із голосовим інтерфейсом (AI-асистент).

З метою підвищення релевантності відповідей у конкретному контексті (освіта, медицина, юриспруденція), можливе донавчання моделі на спеціалізованих текстах:

- додавання приватних баз знань через Retrieval-Augmented Generation (RAG)[16];
- використання LangChain або Haystack для кастомного пошуку по документах;
- реалізація обробки PDF, DOCX, XLSX та інших документів.

Для зручної інтеграції системи з іншими сервісами доцільно реалізувати SDK на Python/JavaScript, а також API-документацію через Swagger/OpenAPI. Це дозволить:

- легко інтегрувати систему у чат-ботів, CRM, LMS;
- створити Telegram-бота або десктопну програму;
- реалізувати багатокористувацький режим з автентифікацією.

Для покращення UX важливо впровадити систему збору статистики, логів та відгуків:

- аналіз частоти запитів;
- визначення “популярних тем”;
- дашборд адміністратора з графіками активності;
- виявлення помилок і “порожніх” відповідей.

З розвитком системи варто реалізувати механізми відповідності GDPR, включно з:

- шифруванням запитів і відповідей;
- системою ролей користувачів;
- можливістю автоматичного видалення історії.

В результаті, було проведено оцінювання якості реалізованої системи за допомогою сучасних метрик (BLEU, ROUGE, latency) та експертного аналізу. Порівняльне тестування із популярними LLM-платформами (ChatGPT, Gemini, Claude) показало, що система демонструє високу ефективність при збереженні автономності та кастомізації.

Проаналізовано сильні та слабкі сторони обраних моделей (DistilGPT2, GPT-NeoX, GPT-3.5-turbo), а також окреслено перспективні напрямки розвитку системи — інтеграція новітніх моделей (GPT-4, Claude 3), мультимовна підтримка, голосовий інтерфейс, локальне донавчання, розширення API та інструментів аналітики.

Результати підтверджують, що система є повноцінною альтернативою комерційним рішенням у низці сценаріїв, а її архітектура дозволяє масштабувати функціональність відповідно до потреб користувачів.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено та реалізовано веб-орієнтовану інформаційну систему з обробки природної мови, яка поєднує сучасні досягнення в галузі глибинного навчання та технології веб-розробки. У процесі дослідження було досягнуто всіх поставлених завдань: проведено аналіз актуальних NLP-систем, обґрунтовано вибір архітектури програмного рішення, реалізовано NLP-модулі генерації та класифікації тексту, а також здійснено тестування системи за якісними та кількісними метриками.

Основними інструментами реалізації стали бібліотеки HuggingFace Transformers, GPT-2/3.5-turbo, FastAPI, JavaScript, а також SQLite як база даних. Проведене функціональне тестування та порівняння з існуючими аналогами підтвердили, що розроблена система здатна забезпечити швидку генерацію тексту, високий рівень релевантності відповідей та гнучкість налаштувань. Вона може працювати як із зовнішніми API, так і в автономному режимі.

Наукова цінність роботи полягає в обґрунтованому підході до вибору трансформерних моделей, інтеграції їх у клієнт-серверну архітектуру та аналізі ефективності таких рішень у реальному середовищі. Практична значущість полягає в можливості застосування створеної системи у сфері електронного навчання, чат-ботів, документообігу та автоматизованих довідкових сервісів.

Модульність реалізації, використання відкритих інструментів і наявність перспектив для масштабування роблять проєкт придатним для подальшого розвитку: підключення голосових інтерфейсів, інтеграції з приватними базами знань, мультимовної підтримки, а також відповідності сучасним вимогам до безпеки персональних даних.

Слід також відзначити певні обмеження, з якими стикалося дослідження. Зокрема, тестування здійснювалося в умовах обмежених обчислювальних ресурсів, що вплинуло на глибину аналізу великих моделей. Модель GPT-4 доступна лише через API, що ускладнює локальне розгортання, а донавчання на

спеціалізованих корпусах не проводилось через відсутність ліцензійного доступу до повних галузевих баз даних. Незважаючи на це, отримані результати підтверджують життєздатність запропонованої архітектури.

Таким чином, поставлена мета дослідження досягнута, усі завдання — виконані, а отримані результати можуть слугувати як основою для впровадження, так і платформою для подальших досліджень у сфері штучного інтелекту та мовної інженерії.

ПЕРЕЛІК ПОСИЛАНЬ

1. Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer.
2. Bengio, Y., Courville, A., & Vincent, P. (2013). *Representation Learning: A Review and New Perspectives*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798-1828.
3. Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
4. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). *Language Models are Few-Shot Learners*. Advances in Neural Information Processing Systems, 33.
5. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).
6. Dodge, J., & Krishna, K. (2019). *Evaluation of Large Language Models*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).
7. FastAPI: Everything you need to know about the most widely used Python web framework for Machine Learning. DataScientest. URL: https://datascientest.com/en/fastapi-everything-you-need-to-know-about-the-most-widely-used-python-web-framework-for-machine-learning (дата звернення: 15.05.2025).
8. Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers. (Фокусується на нейронних мережах в NLP).
9. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

10. Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd ed.). Prentice Hall.
11. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, A., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.
12. Liu, X., Song, P., Hou, K., & Zhang, Y. (2020). *Survey on Text Generation Methods for Natural Language Processing*. Journal of Data Science and Intelligent Computing, 2(1), 1-10.
13. Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.
14. McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). *Learned in Translation: Contextualized Word Vectors*. Advances in Neural Information Processing Systems, 30.
15. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep Contextualized Word Representations*. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1.
16. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. OpenAI Blog, 1(8).
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems, 30.
18. What is Angular in Web Development? Glossary. Sanity.io. URL: https://www.sanity.io/glossary/angular (дата звернення: 15.05.2025).
19. Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). *Capsule Networks for Sentiment Analysis*. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.

20. Паздрій Ігор. Порівняльний аналіз систем розробки програмного забезпечення на основі фреймворків. Вісник Хмельницького національного університету, 2023, №1317, с. 155–161. URL: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2023/03/vknu-ts-2023-n1317-155-161.pdf (дата звернення: 15.05.2025).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

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

КВАЛІФІКАЦІЙНА РОБОТА
на тему: «Розробка веб-додатку для створення тематичних повідомлень з використанням моделей генеративного штучного інтелекту»

Виконав: здобувач вищої освіти гр. ШІД-41 Юнусов Нурлан
Керівник: кандидат технічних наук, професор Андрій Бондарчук

2025

- **Мета:**
  Розробити веб-систему обробки природної мови (NLP) з використанням моделей трансформерного типу (GPT, BERT), здатну генерувати логічні тематичні відповіді.
- **Об’єкт дослідження:**
  Процес автоматизованої обробки природної мови користувача у веб-середовищі.
- **Предмет дослідження:**
  Архітектура, моделі та програмні засоби для побудови NLP-системи на основі глибокого навчання.

Основні завдання кваліфікаційної роботи

- Дослідити сучасні NLP-системи та їх архітектури
- Проаналізувати бібліотеки глибокого навчання для NLP (Transformers, spaCy, NLTK)
- Розробити архітектуру та реалізувати веб-додаток з інтегрованим NLP-модулем
- Виконати класифікацію та генерацію тексту за допомогою трансформерних моделей
- Оцінити якість результатів за метриками BLEU, ROUGE, точність, F1-score
- Інтегрувати систему у веб-інтерфейс і протестувати її роботу
- Провести апробацію на тестових запитах та сформулювати висновки

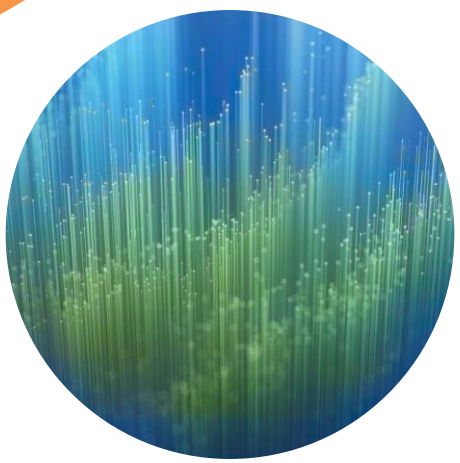
Технології та інструменти

HuggingFace, spaCy, NLTK

GPT-2, BERT, RoBERTa, GPT-3.5

FastAPI, SQLite, HTML/CSS/JS

Архітектура: REST API, Web-інтерфейс

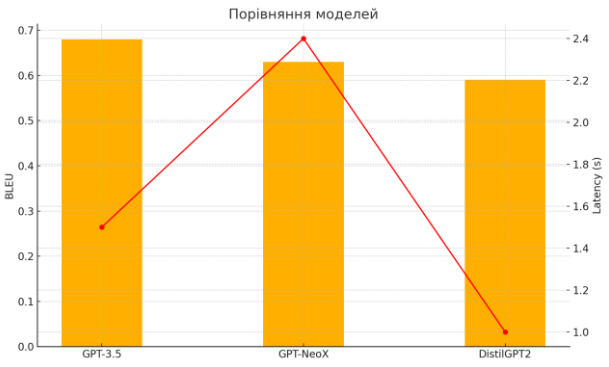


5

Архітектура системи

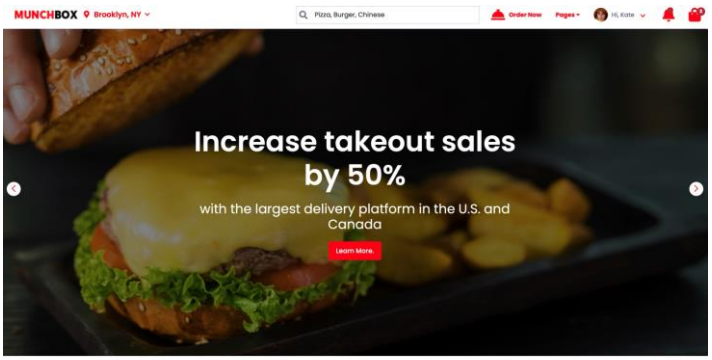
[Користувач]
↓
[Web-інтерфейс (браузер)]
↓
[FastAPI-сервер]
↓
[NLP-модель (GPT-2 / BERT)]
↑
(відповідь)
↓
[FastAPI → Web → Користувач]

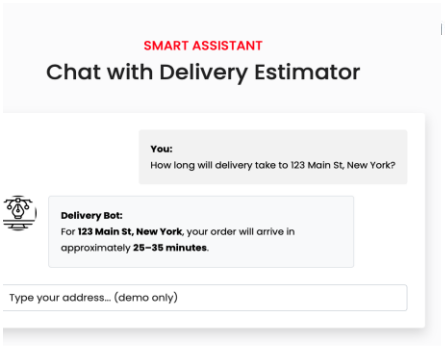
6



Результати тестування

- GPT-3.5: BLEU = 0.68, ROUGE = 0.74, Latency ≈ 1.5 с
- Рівень релевантності: 82%
- Веб-система працює стабільно в реальному часі

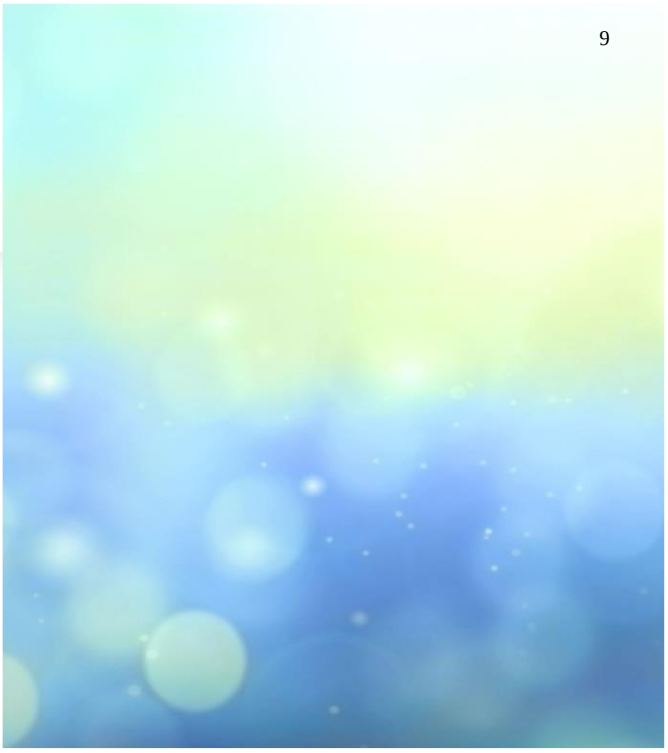




Демонстрація матеріалу

Подальші перспективи

- Підтримка української, англійської, німецької мов
- Додавання голосового інтерфейсу
- Пошук по PDF та локальних базах знань
- Інтеграція у Telegram-ботів, мобільні додатки



Висновки

- Система створена з використанням сучасних NLP-моделей
- Висока точність, швидкість і масштабованість
- Готова до впровадження в освітні та сервісні платформи