МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

КОСТЮКЕВИЧ Назар Юрійович

Нейромережева система інтерактивного діалогу з музейними експонатами на основі технологій обробки природної мови / Neural Network System for Interactive Dialogue with Museum Exhibits Based on Natural Language Processing Technologies

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Штучний інтелект

Кваліфікаційна робота

Виконав студент групи КНШІ-41
Н.Ю. Костюкевич

_____

Науковий керівник:
д.т.н., професор, М. П. Комар

_____

Кваліфікаційну роботу допущено
до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Штучний інтелект

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
КОСТЮКЕВИЧУ Назару Юрійовичу
(прізвище, ім'я, по батькові)

1.Тема кваліфікаційної роботи

Нейромережева система інтерактивного діалогу з музейними експонатами на основі технологій обробки природної мови / Neural Network System for Interactive Dialogue with Museum Exhibits Based on Natural Language Processing Technologies
керівник роботи д.т.н., професор М. П. Комар
затверджені наказом по університету від 01 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- здійснити огляд існуючих реалізацій інтерактивних музейних AI-систем та проаналізувати їхні функціональні можливості;
- проаналізувати сучасні підходи до побудови RAG-систем, методи ASR та TTS, визначити їх переваги та обмеження;
- розробити архітектуру, алгоритмічне та інформаційне забезпечення системи і створити загальну структуру голосового RAG-конвеєру;
- реалізувати програмне забезпечення, створити інтерфейс користувача і провести тестування системи.

5. Перелік графічного матеріалу в роботі:
- загальна архітектура системи;
- схема підсистеми розпізнавання та синтезу мовлення;
- схема рівня RAG: підготовка бази знань та семантичний пошук;
- алгоритм функціонування голосового RAG-конвеєру.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ Н. Ю. Костюкевич
підпис

Керівник роботи_____ д.т.н., професор М. П. Комар
підпис

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 53 с., 9 рис., 8 табл., 2 додатки, 42 джерела.

Об’єктом дослідження є процес природномовного голосового діалогу між відвідувачем музею та програмним агентом, що представляє конкретний музейний артефакт.

Метою кваліфікаційної роботи є розробка та програмна реалізація нейромережевої системи інтерактивного голосового діалогу з музейними експонатами на основі технологій обробки природної мови та архітектури Retrieval-Augmented Generation.

Методами дослідження обрано: метод аналізу для вивчення існуючих реалізацій інтерактивних музейних AI-систем, метод синтезу для поєднання компонентів ASR, RAG та TTS в єдиний конвеєр, методи обробки природної мови для аналізу голосових запитів і генерації відповідей від імені персонажа, метод векторного пошуку для семантичної вибірки контексту з бази знань, а також методи тестування програмного забезпечення.

Результатом виконання кваліфікаційної роботи є нейромережева система інтерактивного діалогу з музейними експонатами, яка забезпечує прийом голосового запиту відвідувача, розпізнавання мовлення, семантичний пошук релевантної інформації в базі знань, формування збагаченого контексту, генерацію відповіді від імені музейного персонажа, синтез мовлення та повернення користувачу текстової й аудіовідповіді.

Результати роботи можуть бути використані для розробки інтерактивних музейних AI-систем, інтелектуальних аудіогідів, голосових асистентів для культурно-освітніх установ, вебзастосунків для діалогу з музейними експонатами, а також програмних рішень на основі технологій обробки природної мови, великих мовних моделей та RAG-архітектури.

Ключові слова: ГОЛОСОВИЙ ДІАЛОГ, RAG-АРХІТЕКТУРА, АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ МОВЛЕННЯ, СИНТЕЗ МОВЛЕННЯ, ВЕКТОРНА БАЗА ДАНИХ, ВЕЛИКІ МОВНІ МОДЕЛІ, ШТУЧНИЙ ІНТЕЛЕКТ.

ANNOTATION

The bachelor's thesis report: 53 pages, 9 figures, 8 tables, 2 appendices, 42 sources.

The object of the research is the process of natural-language voice dialogue between a museum visitor and a software agent representing a specific museum artifact.

The purpose of the qualification work is the development and software implementation of a neural network system for interactive voice dialogue with museum exhibits based on natural language processing technologies and the Retrieval-Augmented Generation architecture.

The research methods include: the method of analysis for studying existing implementations of interactive museum AI systems; the method of synthesis for combining ASR, RAG, and TTS components into a single pipeline; natural language processing methods for analyzing voice queries and generating responses on behalf of a character; the vector search method for semantic retrieval of context from the knowledge base; as well as software testing methods.

The result of the qualification work is a neural network system for interactive dialogue with museum exhibits, which provides reception of a visitor's voice query, speech recognition, semantic search for relevant information in the knowledge base, formation of an enriched context, generation of a response on behalf of a museum character, speech synthesis, and returning both text and audio responses to the user.

The results of the work can be used for the development of interactive museum AI systems, intelligent audio guides, voice assistants for cultural and educational institutions, web applications for dialogue with museum exhibits, as well as software solutions based on natural language processing technologies, large language models, and RAG architecture.

Keywords: VOICE DIALOGUE, RAG ARCHITECTURE, AUTOMATIC SPEECH RECOGNITION, SPEECH SYNTHESIS, VECTOR DATABASE, LARGE LANGUAGE MODELS, ARTIFICIAL INTELLIGENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface – програмний інтерфейс застосунку

ASR – Automatic Speech Recognition – автоматичне розпізнавання мовлення

CPU – Central Processing Unit – центральний процесор

GPU – Graphics Processing Unit – графічний процесор

JSON – JavaScript Object Notation – текстовий формат обміну даними

LLM – Large Language Model – велика мовна модель

NLP – Natural Language Processing – обробка природної мови

QR – Quick Response (code) – двовимірний штрих-код швидкого реагування

RAG – Retrieval-Augmented Generation – генерація з доповненням пошуком

REST – Representational State Transfer – архітектурний стиль вебсервісів

SPA – Single Page Application – односторінковий вебзастосунок

TTS – Text-to-Speech – синтез мовлення з тексту

WER – Word Error Rate – відсоток помилково розпізнаних слів

ВСТУП

Стрімкий розвиток інформаційних технологій та систем штучного інтелекту упродовж останніх років суттєво вплинув на підходи до організації культурно-освітньої комунікації. Особливої популярності набули великі мовні моделі (Large Language Models, LLM) та технології обробки природної мови (Natural Language Processing, NLP), які здатні генерувати зв'язні текстові відповіді та підтримувати природномовний діалог з користувачем. Такі технології активно використовуються у сфері освіти, музейної справи, туризму та створення інтелектуальних систем підтримки прийняття рішень.

Водночас із поширенням мовних моделей у музейній галузі виникає проблема забезпечення точності та достовірності відповідей, що надаються відвідувачам. Традиційні формати взаємодії з музейними колекціями – статичні анотації, аудіогіди та групові екскурсії – не забезпечують можливості ведення повноцінного діалогу. Відвідувач не може поставити уточнювальне запитання та отримати відповідь у реальному часі. У зв'язку з цим актуальним завданням є створення нейромережевої системи інтерактивного діалогу, здатної відповідати на запитання відвідувачів від імені конкретного музейного персонажа або експоната, спираючись виключно на верифіковані джерела знань.

Розробка спеціалізованої діалогової системи для музейного середовища потребує поєднання кількох технологій: архітектури Retrieval-Augmented Generation (RAG) для прив'язки генерації до перевірених фактів, механізмів управління контекстом для підтримки багатоходових бесід, а також prompt engineering для відтворення мовленнєвого стилю конкретного персонажа. Такий підхід дозволяє значно підвищити освітню цінність музейного відвідування та зменшити навантаження на екскурсоводів.

Метою кваліфікаційної роботи є розробка та програмна реалізація нейромережевої системи інтерактивного голосового діалогу з музейними експонатами на основі технологій обробки природної мови та архітектури Retrieval-Augmented Generation. У межах виконання роботи передбачається аналіз предметної області, огляд існуючих технологічних рішень, а також створення

програмної системи, здатної автоматизувати процес взаємодії відвідувача з музейним контентом у форматі природномовного голосового діалогу. Для досягнення мети поставлено такі завдання:

– здійснити огляд існуючих реалізацій інтерактивних музейних AI-систем та проаналізувати їхні функціональні можливості;

– проаналізувати сучасні підходи до побудови RAG-систем, методи ASR та TTS, визначити їх переваги та обмеження;

– розробити архітектуру, алгоритмічне та інформаційне забезпечення системи і створити загальну структуру голосового RAG-конвеєру;

– реалізувати програмне забезпечення, створити інтерфейс користувача і провести тестування системи.

Предметом дослідження є моделі, методи та програмні засоби обробки природної мови – автоматичне розпізнавання мовлення, семантичний векторний пошук, генерація відповідей великою мовною моделлю та синтез мовлення, – що застосовуються для побудови нейромережевої системи інтерактивного голосового діалогу з музейними експонатами.

Практичне значення запропонованої системи полягає у можливості її безпосереднього розгортання у музейних експозиціях у вигляді інтерактивних сенсорних стендів або веб-застосунків, доступних через QR-коди поблизу артефактів. Система легко масштабується на нові експонати виключно через підготовку текстових документів і одноразове індексування без змін у програмному коді. Розроблене рішення може слугувати основою для аналогічних застосунків в інших предметних галузях – освіті, архівній справі, туристичному сервісі.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах IV Всеукраїнської науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі» (ІКСМ), м. Тернопіль, ЗУНУ, 20 травня 2026 р. (додаток А).

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметною областю даної кваліфікаційної роботи є інтелектуальні діалогові системи для музейного середовища, що поєднують технології обробки природної мови, семантичного пошуку та генеративних великих мовних моделей. Нижче розглянуто структуру цієї предметної області, її ключові поняття, наявні проблеми та технологічні передумови їх вирішення.

У сучасних умовах стрімкого розвитку інформаційних технологій важливе місце займають інтелектуальні діалогові системи, які дозволяють автоматизувати процеси взаємодії з користувачем у форматі природномовного спілкування [22, 35]. Одним із найбільш перспективних напрямів є створення систем, що здатні сприймати голосові запити, вести зв'язну бесіду та надавати фактично точні відповіді на основі верифікованої зовнішньої бази знань [20]. Такі системи активно впроваджуються у сферах освіти, туристичного сервісу, музейної справи та інтерактивного навчання [19, 26].

Музейна інформатика є міждисциплінарною галуззю, що досліджує застосування інформаційних технологій у роботі музеїв та культурних установ. Упродовж останніх двох десятиліть у цій галузі відбулася значна цифрова трансформація: від простого оцифрування фондів та створення онлайн-каталогів до впровадження доповненої реальності та тривимірного сканування артефактів. Попри цей прогрес, ключовою незаповненою нішею залишається інтерактивний двосторонній діалог між відвідувачем і музейним контентом у реальному часі.

Класичний аудіогід функціонує за принципом відтворення заздалегідь записаного тексту та є принципово односпрямованим: відвідувач є пасивним споживачем інформації. Якщо його цікавить деталь поза межами стандартної анотації, він змушений шукати відповідь самостійно. Навіть найсучасніші мультимедійні стенди, що використовують сенсорні екрани та анімацію, залишаються по суті системами доставки фіксованого контенту, а не повноцінними діалоговими агентами [21].

Концепція «персонажа-розповідача» у музейному контексті передбачає

створення програмного агента, здатного відповідати від першої особи від імені конкретного артефакту або постаті. Наприклад, золота скіфська пектораль IV ст. до н.е. може розповідати про своє виготовлення, художній стиль та культурне значення. Такий підхід значно підвищує емоційне залучення відвідувача та освітній ефект завдяки механізмам наративного мислення [31]. Реалізація даної концепції потребує не лише генерації тематично правильного тексту, але й утримання рольового стилю персонажа впродовж усього діалогу [37].

З технічного боку задача голосового діалогу з музейним персонажем охоплює кілька взаємопов'язаних підзадач. Насамперед, необхідне автоматичне розпізнавання мовлення (Automatic Speech Recognition, ASR) для перетворення голосового введення відвідувача на текст. Далі виконується семантичний пошук у базі знань персонажа для знаходження релевантних фактів. Наступним кроком є генерація природномовної відповіді великою мовною моделлю на основі знайдених фрагментів. Завершальним етапом є синтез мовлення (Text-to-Speech, TTS) для озвучення відповіді природним голосом.

Ключовою технічною проблемою при використанні великих мовних моделей є явище «галюцинацій» – генерація правдоподібних, але фактично хибних відповідей. Для культурно-освітніх сценаріїв, де відвідувач довіряє системі як офіційному представнику музею, це є абсолютно неприйнятним ризиком. Для вирішення цієї проблеми застосовується архітектура Retrieval-Augmented Generation (RAG) [5], яка примусово прив'язує генерацію відповіді до верифікованих фрагментів бази знань, усуваючи необхідність покладатися на «пам'ять» моделі.

Поява ефективних моделей sentence-embeddings, що кодують семантичний зміст тексту у числовий вектор у висококонвергентному просторі, зробила можливим пошук за смислом, а не лише за ключовими словами. Це означає, що запит «звідки походить цей виріб?» знаходить фрагмент «артефакт виготовлений у степовому Причорномор'ї» навіть попри повністю різне формулювання. Разом із нейромережевими методами ASR та TTS, що забезпечують якісне розпізнавання та синтез українського мовлення, це дозволяє реалізувати повністю природний голосовий інтерфейс без клавіатурного введення.

1.2 Огляд і аналіз існуючих рішень

Серед сучасних реалізацій інтерактивних музейних AI-систем найбільш близьким до досліджуваної тематики є проєкт Museum of Zoology University of Cambridge у співпраці з Nature Perspectives [27, 41]. У межах цього проєкту відвідувачі можуть сканувати QR-коди біля експонатів і спілкуватися з ними у форматі текстового або голосового діалогу. Система моделює відповіді від першої особи, що створює ефект «оживлення» музейного об'єкта та підвищує емоційне залучення відвідувача. Іншим близьким прикладом є Artistic Chatbot, розроблений для виставки Warsaw Academy of Fine Arts, де реалізовано voice-to-voice QA-систему з відповідями, що ґрунтуються на попередньо підготовленому корпусі матеріалів [36].

У таблиці 1.1 наведено відомі реалізації та близькі аналоги.

Таблиця 1.1 – Відомі реалізації та близькі аналоги

№	Назва / установа	Коротка характеристика	Чим корисна для порівняння
1	2	3	4
1	Museum of Zoology, University of Cambridge + Nature Perspectives [27, 41]	Відвідувачі сканують QR-код біля експоната і спілкуються з 13 музейними зразками (додо, кит, червона панда). Діалог від першої особи, мобільний доступ.	Дуже близько до теми: «оживлення» експонатів, персонажний діалог, first-person simulation, мобільний доступ через QR-код.
2	Artistic Chatbot, Warsaw Academy of Fine Arts [36]	Voice-to-voice QA-чатбот для виставки Faculty of Media Art. Відповідає на усні запитання про факультет, митців, виставки.	Найближчий технологічний аналог: voice-to-voice, доменна база знань, відповіді прив'язані до підготовленого корпусу.

Продовження таблиці 1.1

1	2	3	4
3	Centre Pompidou chatbot / Ask Mona [28]	AI-асистент, доступний французькою та англійською. Розпізнає об'єкт за фото, надає інтерактивну розповідь.	Приклад інтеграції AI з музейною базою знань і кураторським контролем точності.
4	Cooper Hewitt Museum – Object Phone [21]	Ранній музейний чатбот через SMS/телефон. Еволюціонував у повноцінну діалогову підтримку.	Показує еволюцію від простого SMS-сервісу до діалогу з музейними об'єктами.
5	Case Museo di Milano – Chat Game [19]	Чатбот-квест через Facebook Messenger і Telegram для підлітків.	Приклад поєднання чатбота, сторітелінгу та гейміфікації.
6	Museo de Arte Moderno de Buenos Aires – Dialog with the Artwork [21]	Діалог з твором мистецтва через чатбот. Першим «співрозмовником» стала скульптура Bio Cosmos.	Відповідає ідеї «діалогу з експонатом» без сучасного LLM/RAG.
7	Martmuseumbot, Museo di arte moderna e contemporanea di Trento e Rovereto [19]	Інтерактивний бот для супроводу відвідувачів у Telegram та Facebook Messenger.	Приклад музейного чатбота як цифрового гіда для колекцій.
8	Anne Frank House Bot [42]	Чатбот для одного з найчутливіших історичних музеїв. Надавав інформацію про біографію Анни Франк.	Важливий приклад для теми достовірності й етичності AI у культурно-історичному контексті.
9	Amuseapp [26]	Комерційна платформа AI-аудіогідів для музеїв із AI-generated multimedia.	Готова комерційна реалізація AI-аудіогіда для музеїв.
10	ZKM – «Talk to me! Chatbots in Museums» [42]	Систематизує приклади музейних чатботів з 2018 року, власний чатбот для сервісних запитів.	Оглядовий приклад розвитку музейних чатботів.

Найбільш релевантними для цієї роботи є три реалізації:

1. Museum of Zoology Cambridge + Nature Perspectives [27, 41] – майже прямий аналог концепції «експонат відповідає від першої особи». Система працює через

QR-код, підтримує спілкування з конкретними експонатами та орієнтована на емоційне залучення відвідувача.

2. Artistic Chatbot, Warsaw Academy of Fine Arts [36] – найближчий технологічний аналог, оскільки реалізовано voice-to-voice запитання-відповідь, доменну базу знань і відповіді, прив'язані до підготовленого корпусу.

3. Centre Pompidou / Ask Mona [28] – сильний приклад практичного музейного AI-асистента, де використано генеративний AI, музейний контент, кураторські матеріали та контроль відповідності тону й науковим стандартам музею.

Пряме використання великих мовних моделей без прив'язки до зовнішньої бази знань є технологічно найпростішим підходом: запитання передається до LLM із системним промптом, що описує персонажа. Сучасні моделі, зокрема GPT-4o від OpenAI [1], Gemini 1.5 Pro від Google [2] та Llama 3.3 від Meta [18, 39], здатні генерувати якісні, зв'язні відповіді у стилі будь-якого заданого персонажа. Однак без прив'язки до верифікованого джерела знань моделі регулярно формують правдоподібні, але хибні факти [25]. Для музейного контексту, де відвідувач довіряє системі як офіційному джерелу інформації, це є принципово неприйнятним ризиком.

Retrieval-Augmented Generation (RAG) є підходом, запропонованим Lewis et al. у роботі 2020 року [5]. RAG поєднує переваги семантичного пошуку у зовнішній базі знань із генеративними можливостями LLM. На першому кроці система знаходить у векторній базі найбільш релевантні до запиту фрагменти верифікованих документів. На другому кроці LLM генерує відповідь, що ґрунтується виключно на цих фрагментах як контексті. Такий підхід суттєво знижує ризик галюцинацій, забезпечує простежуваність відповіді до першоджерела та дозволяє оновлювати базу знань без перенавчання моделі. Для музейних діалогових систем RAG є сьогодні галузевим стандартом де-факто.

Окремої уваги заслуговує вибір технологій для підсистем ASR та TTS. Модель Whisper від OpenAI, навчена на 680 000 годинах багатомовного аудіо, підтримує понад 90 мов включно з українською та демонструє конкурентоспроможну точність. Її оптимізована реалізація faster-whisper на основі фреймворку CTranslate2 із квантизацією ваг дозволяє виконувати розпізнавання

мовлення на звичайному CPU без дорогого GPU-обладнання [6], що є принциповою перевагою для розгортання на музейних стендах. Для синтезу мовлення бібліотека Edge TTS надає доступ до нейромережевих голосових двигунів Microsoft Azure через відкритий протокол без необхідності платної підписки, підтримуючи якісні українськомовні голоси [7].

Серед готових фреймворків для побудови RAG-систем виділяють LangChain [8] та LlamaIndex. Обидва надають готові абстракції для завантаження документів, їх розбиття на фрагменти, генерації ембедингів та організації векторного пошуку. Як векторне сховище для локального розгортання використовується ChromaDB [11] – вбудована база даних, що не потребує запуску окремого сервісу та нативно інтегрується з LangChain. Як LLM для генерації відповідей обрано модель Llama 3.3 (70B) [18], доступну через Groq API, що забезпечує затримку генерації 0.3–0.8 секунди завдяки спеціалізованим LPU-чіпам [9].

1.3 Постановка задачі дослідження

Аналіз підходів та відомих реалізацій, наведений у підрозділі 1.2, дозволяє сформулювати ключові технічні вимоги до розробленої системи. Найближчі аналоги – Museum of Zoology Cambridge та Artistic Chatbot Варшавської академії мистецтв – демонструють попит на голосовий інтерфейс і персонажний діалог від першої особи. Разом із тим вони реалізовані на закритих або комерційних платформах без доступної технічної документації. Це підтверджує потребу у відкритому відтворюваному рішенні, побудованому на сучасному стеку NLP-технологій.

Ключовим завданням при використанні діалогових систем у музейному середовищі є забезпечення фактичної точності відповідей при збереженні природності та доступності взаємодії. Оскільки великі мовні моделі генерують текст на основі статистичних закономірностей, без додаткових механізмів контролю вони можуть формувати відповіді, що виглядають переконливо, але містять вигадані факти. Тому архітектура системи повинна примусово прив'язувати

генерацію до верифікованої бази знань, як це реалізовано у найбільш технологічно зрілих аналогах (Centre Pompidou / Ask Mona, Warsaw Artistic Chatbot).

Суттєвою вимогою, підтвердженою досвідом Cambridge та Warsaw проєктів, є підтримка голосового введення. Відвідувач музею, особливо на інтерактивному стенді, не повинен вводити текст із клавіатури – він натискає кнопку мікрофону і звертається до системи природним мовленням. Голосове повідомлення у форматі webm або wav передається на сервер, де розпізнавальна модель перетворює його на текст для подальшої обробки RAG-конвеєром. Рівносильно важливою є функція синтезу мовлення: згенерована текстова відповідь персонажа озвучується та надсилається клієнту у вигляді аудіофайлу mp3 для автоматичного відтворення.

Таким чином, задачею дослідження є розробка нейромережевої системи інтерактивного голосового діалогу з музейними експонатами, яка на основі архітектури RAG та великої мовної моделі забезпечує: розпізнавання українського мовлення відвідувача на CPU-обладнанні без GPU; ведення природномовної бесіди від імені обраного музейного персонажа (аналогічно до Cambridge-проєкту, але з відкритою архітектурою); прив'язку відповідей до верифікованої бази знань персонажа; озвучення відповіді синтезованим голосом; коректну відмову від відповіді на запитання поза межами бази знань без звернення до LLM.

Вхідними даними системи є аудіозапис запитання відвідувача у форматах webm або wav та ідентифікатор обраного музейного персонажа. Вихідними даними є текстова відповідь і аудіофайл mp3, синтезований від імені персонажа на основі релевантних фрагментів бази знань. Функціональний ланцюжок обробки: транскрипція аудіо → семантичний пошук у базі знань → формування збагаченого промпту → генерація відповіді LLM → синтез мовлення → повернення клієнту. Система повинна також реалізовувати механізм порогового відсікання: якщо жоден фрагмент бази знань не є достатньо семантично близьким до запитання, система повертає ввічливу відмову без звернення до LLM, що запобігає галюцинаціям.

Нефункціональні вимоги включають: загальний час відгуку pipeline до 6 секунд на CPU-обладнанні без GPU; портативність через Docker-контейнеризацію; безпеку API-ключів через механізм змінних оточення без їх збереження у репозиторії; масштабованість – можливість додавання нових персонажів виключно

через підготовку текстових документів і одноразове індексування без зміни програмного коду; адаптивний веб-інтерфейс, придатний для використання на сенсорних стендах та мобільних пристроях відвідувачів. Зазначений перелік вимог відрізняє дану систему від аналогів відкритою архітектурою та повністю локальним виконанням ASR-компонента.

Сформульовані функціональні та нефункціональні вимоги безпосередньо узгоджуються з метою кваліфікаційної роботи — розробкою та програмною реалізацією нейромережевої системи інтерактивного голосового діалогу з музейними експонатами. Для досягнення поставленої мети необхідно виконати такі завдання:

– здійснити огляд існуючих реалізацій інтерактивних музейних AI-систем та проаналізувати їхні функціональні можливості;

– проаналізувати сучасні підходи до побудови RAG-систем, методи ASR та TTS, визначити їх переваги та обмеження;

– розробити архітектуру, алгоритмічне та інформаційне забезпечення системи і створити загальну структуру голосового RAG-конвеєру;

– реалізувати програмне забезпечення, створити інтерфейс користувача і провести тестування системи.

Отже, у даному розділі проаналізовано предметну область інтерактивних музейних діалогових систем та визначено актуальність використання технологій обробки природної мови для взаємодії відвідувачів з експонатами. Розглянуто відомі AI-рішення для музейного середовища, зокрема системи діалогу з експонатами, голосові асистенти та чатботи. Встановлено, що найбільш доцільним підходом для забезпечення достовірності відповідей є використання RAG-архітектури, яка поєднує семантичний пошук у базі знань із генеративними можливостями великої мовної моделі. На основі проведеного аналізу сформульовано постановку задачі дослідження.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ НЕЙРОМЕРЕЖЕВОЇ СИСТЕМИ ІНТЕРАКТИВНОГО ДІАЛОГУ З МУЗЕЙНИМИ ЕКСПОНАТАМИ

2.1 Загальна структура нейромережевої системи

Проєктування архітектури нейромережевої системи інтерактивного голосового діалогу вимагає застосування сучасних підходів до побудови мультимодальних інформаційних систем. Обрано дворівневу клієнт-серверну архітектуру: серверний рівень (backend) реалізує весь обчислювальний пайплайн, клієнтський рівень (frontend) забезпечує веб-інтерфейс і медіаввід.

Серверний рівень (Backend) є вертикальним стеком обробки запиту, через який послідовно проходить кожне голосове звернення відвідувача: прийом аудіо → ASR-транскрипція → RAG-пошук → генерація LLM → TTS-синтез → повернення результату. Клієнтський рівень реалізований як тонкий SPA на React, основна роль якого – захоплення голосового введення через MediaRecorder API, відображення відповіді та відтворення аудіо.

Технологічний стек системи визначено у таблиці 2.1.

Таблиця 2.1 – Технологічний стек нейромережевої системи голосового діалогу

Рівень системи	Технологія / Інструмент	Призначення у проєкті
1	2	3
Веб-фреймворк	Python / FastAPI	Асинхронний REST API, валідація Pydantic, Swagger UI
Розпізнавання мовлення	faster-whisper (small, int8)	Транскрипція аудіо на CPU без GPU, підтримка UK
Синтез мовлення	Edge TTS (uk-UA-PolinaNeural)	Безкоштовне озвучення відповіді персонажа
Векторна база	ChromaDB	Локальне сховище ембедингів, нативна LangChain-інтеграція
Ембединги	sentence-transformers	Семантичне кодування запитів і фрагментів БЗ

Продовження таблиці 2.1

1	2	3
RAG-фреймворк	LangChain	Pipeline: document loader → splitter → retriever
Велика мовна модель	Llama 3.3 70B / Groq API	Генерація від імені персонажа, затримка 0.3–0.8 с
Фронтенд	React 18 + Vite + Tailwind CSS	Адаптивний SPA, голосовий запис, відтворення аудіо
Контейнеризація	Docker + Docker Compose	Два сервіси, том chroma_data, .env-конфігурація

Схему загальної архітектури системи наведено на рисунку 2.1.

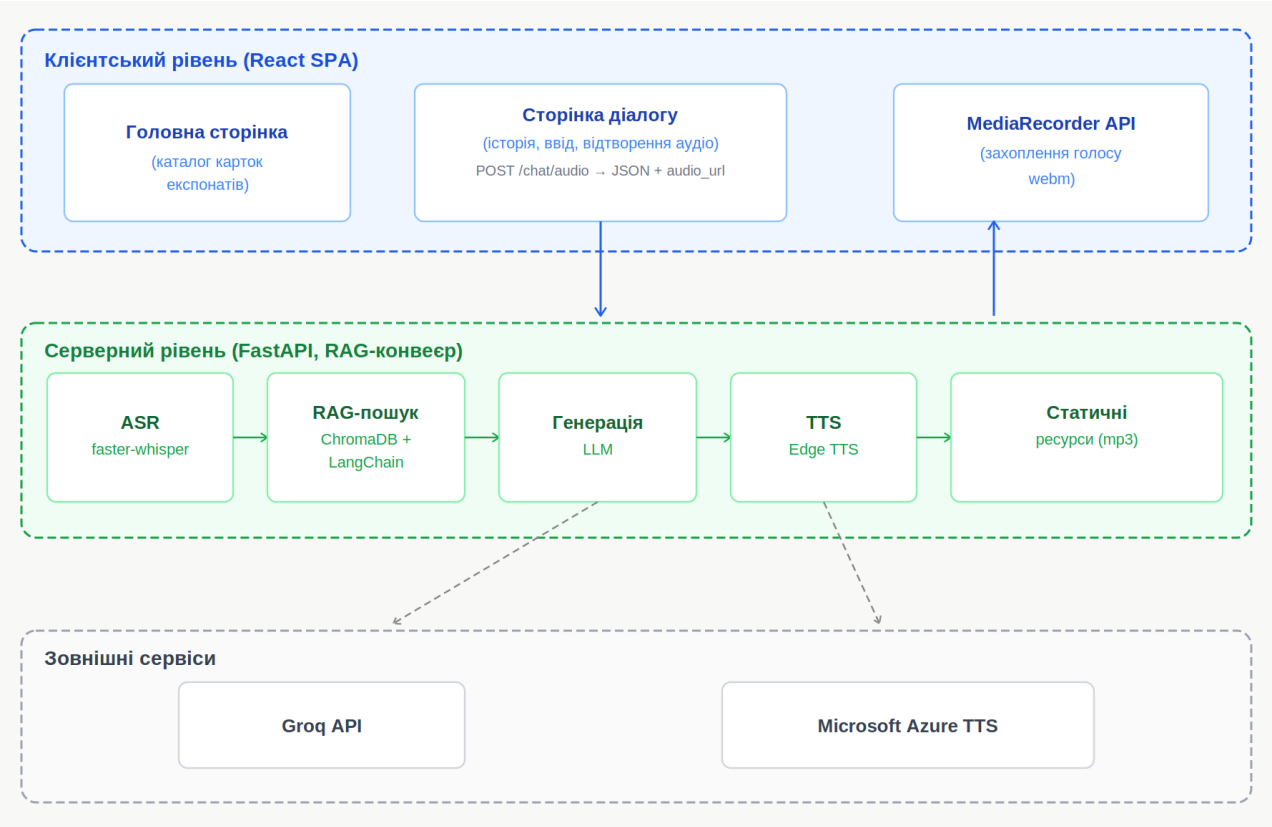

Рисунок 2.1 – Загальна архітектура системи

2.2 Алгоритмічне забезпечення

Узагальнений алгоритм функціонування системи описує логіку послідовної обробки голосового звернення відвідувача від моменту його надходження до повернення озвученої відповіді. Алгоритм передбачає всі ситуації, що можуть

виникнути під час обробки, включно з випадком відсутності релевантного контексту. Послідовність кроків роботи голосового RAG-конвеєра є такою.

1. Прийом аудіофайлу запитання відвідувача у форматі multipart/form-data разом з ідентифікатором обраного експоната та збереження його у тимчасовій директорії з унікальним іменем.

2. Перетворення аудіо у вихідний формат із частотою дискретизації 16 кГц та автоматичне розпізнавання мовлення підсистемою ASR, що повертає текстову транскрипцію запитання.

3. Семантичне кодування тексту запитання у вектор та пошук у векторній базі найрелевантніших фрагментів бази знань обраного персонажа за косинусною подібністю.

4. Перевірка порогу релевантності: якщо жоден фрагмент не є достатньо близьким до запиту, формується ввічлива відмова без звернення до мовної моделі (захист від галюцинацій).

5. Формування збагаченого промпту шляхом підстановки знайдених фрагментів та запитання у шаблон системної підказки персонажа.

6. Генерація природномовної відповіді великою мовною моделлю на основі збагаченого промпту.

7. Синтез згенерованого тексту у аудіофайл формату mp3 підсистемою TTS.

8. Повернення клієнту відповіді у вигляді JSON-об’єкта, що містить транскрипцію запиту, текст відповіді та URL-адресу аудіофайлу.

Графічне представлення наведеного алгоритму у вигляді блок-схеми подано на рисунку 2.2

Мультимодальний характер системи реалізується двома взаємодоповнювальними підсистемами обробки мовлення, що є критично важливими для забезпечення природної голосової взаємодії відвідувача з цифровим персонажем.

Підсистема автоматичного розпізнавання мовлення побудована на бібліотеці faster-whisper – оптимізованій реалізації моделі Whisper від OpenAI [3]. Ця реалізація використовує квантизацію ваг до формату INT8 на основі фреймворку

CTranslate2, що дозволяє знизити обсяг оперативної пам’яті та пришвидшити виведення моделі приблизно у чотири рази порівняно з оригінальною реалізацією без відчутної втрати точності. У системі використовується модель розміру small, що містить 244 мільйони параметрів і при int8-квантизації займає близько 250 МБ оперативної пам’яті.

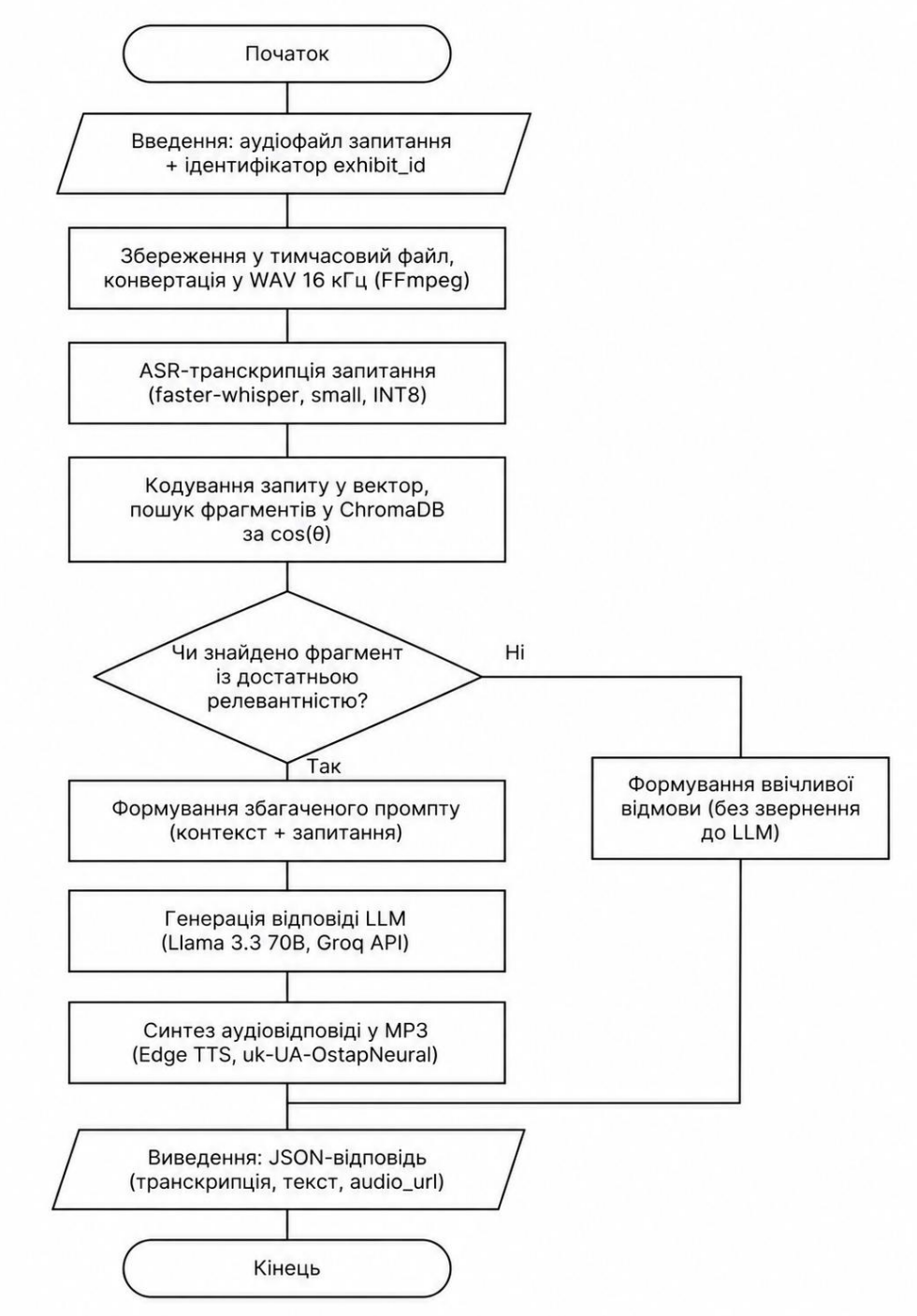

Рисунок 2.2 – Блок-схема алгоритму функціонування голосового RAG-конвеєра

Якість розпізнавання вимірюється метрикою Word Error Rate (WER), що визначається як

$$WER = (S + D + I) / N \quad (2.1)$$

де S – кількість замін (substitutions);

D – кількість видалень (deletions);

I – кількість вставок (insertions);

N – загальна кількість слів в еталонній транскрипції.

Підсистема синтезу мовлення реалізована за допомогою бібліотеки Edge TTS [7] – інструменту, що надає доступ до нейромережевих TTS-сервісів Microsoft Azure без необхідності платної підписки. Для озвучення відповідей персонажів застосовується голос uk-UA-OstapNeural, що забезпечує природне та виразне звучання українського мовлення. Після отримання текстової відповіді від мовної моделі підсистема асинхронно генерує mp3-аудіофайл та зберігає його у директорії статичних ресурсів сервера з унікальним іменем для уникнення конфліктів при паралельних запитах. Схему взаємодії підсистем розпізнавання та синтезу мовлення у рамках голосового конвеєра наведено на рисунку 2.3

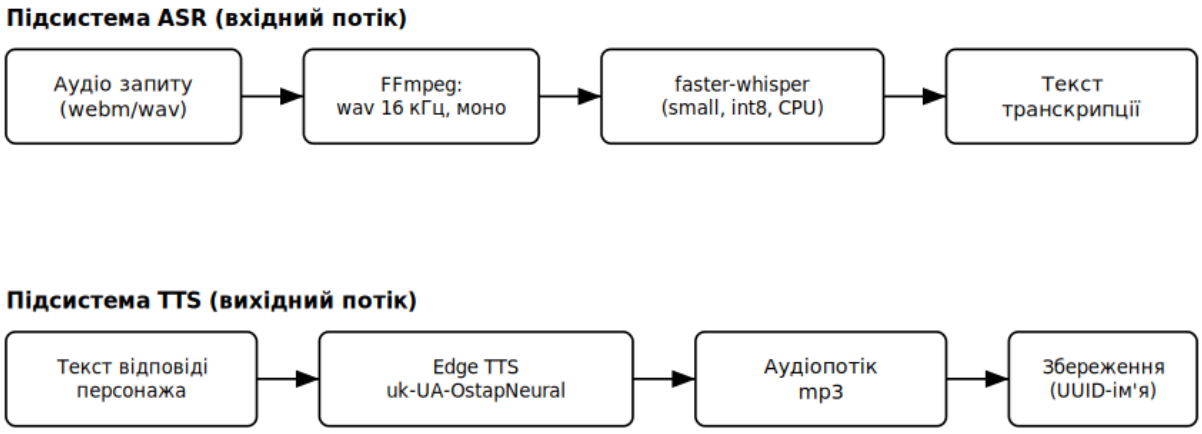

Рисунок 2.3 – Схема підсистеми розпізнавання та синтезу мовлення

Центральним алгоритмічним елементом системи є рівень підготовки контексту за технологією RAG. Текст транскрибованого запитання перетворюється

тією самою моделлю sentence-transformers, що й при індексуванні. Отриманий вектор запиту порівнюється з усіма збереженими векторами колекції персонажа методом косинусної подібності

$$cos(\theta) = (q \cdot d) / (\|q\| \cdot \|d\|) \quad (2.2)$$

де $q \cdot d$ – скалярний добуток векторів запиту і фрагмента;

$\|q\|$ – евклідова норма вектора запиту;

$\|d\|$ – евклідова норма вектора фрагмента;

θ – кут між векторами у просторі ембедингів.

Відбирається фіксована кількість фрагментів із найвищим значенням косинусної подібності, які надалі формують контекст для генерації. Схему підготовки бази знань та процедуру семантичного пошуку наведено на рисунку 2.4.

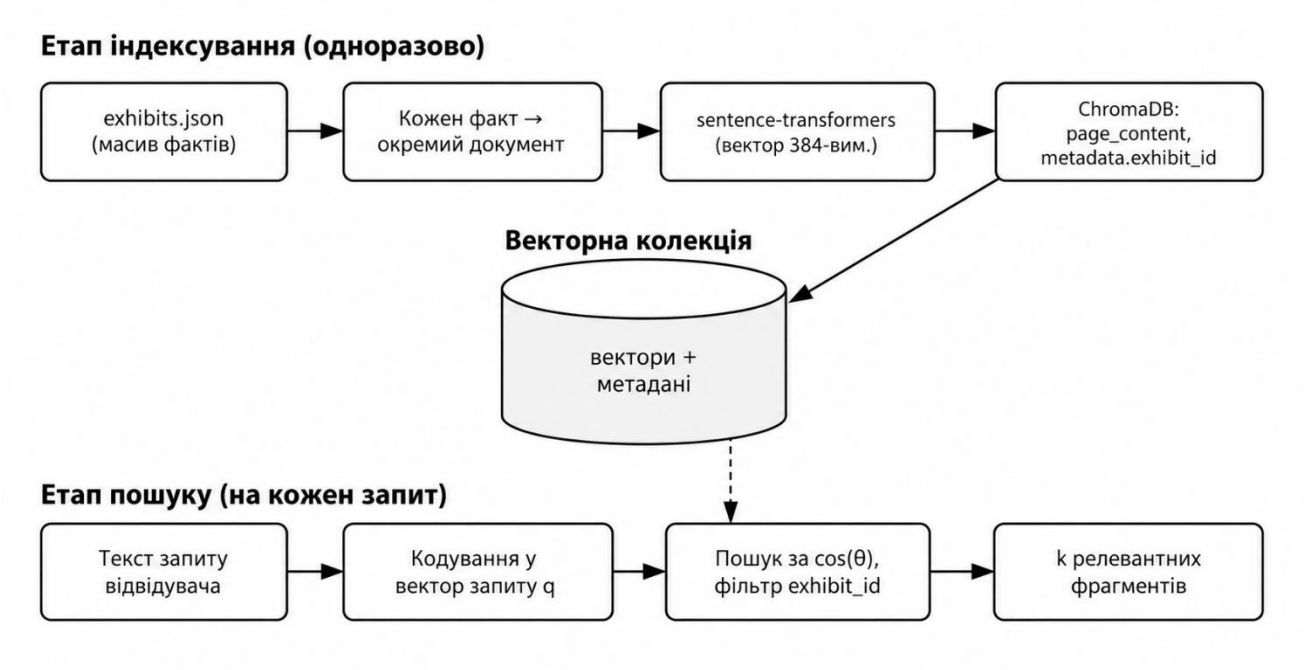

Рисунок 2.4 – Схема рівня RAG: підготовка бази знань та семантичний пошук

Управління персонажем реалізовано через механізм системних підказок. Системна підказка кожного персонажа включає три обов’язкові компоненти: визначення ролі (ким є персонаж, від імені якого артефакта надається відповідь);

інструкцію щодо стилю (відповідати виключно від першої особи, спираючись лише на наданий контекст); інструкцію щодо поведінки за відсутності релевантного контексту (повертати ввічливу відмову без звернення до загальних знань мовної моделі). Схему формування запиту до Groq API наведено на рисунку 2.5.

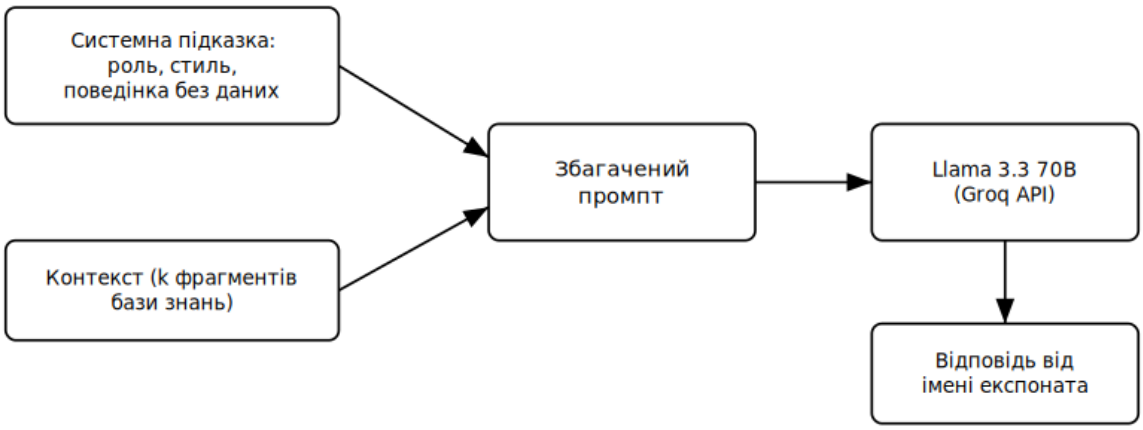

Рисунок 2.5 – Взаємодія підказки, контексту та великої мовної моделі

2.3 Інформаційне забезпечення

2.3.1 Опис вхідної та вихідної інформації

Інформаційне забезпечення системи визначається складом вхідних та вихідних даних, що циркулюють у голосовому конвеєрі. Вхідними даними системи є такі типи інформації.

1. Аудіофайл запитання – голосове звернення відвідувача у форматі webm або wav, обсягом до 10 МБ. Формат – бінарний потік, що передається методом multipart/form-data. Частота надходження визначається активністю відвідувача і відповідає одному файлу на кожне голосове запитання.

2. Ідентифікатор експоната (exhibit_id) – рядковий ідентифікатор обраного персонажа. Формат – текстовий рядок. Надходить разом із кожним запитом і однозначно визначає колекцію бази знань для семантичного пошуку.

3. Текстовий запит (опціонально) – альтернатива голосовому введенню; текстовий рядок із запитанням відвідувача, що надходить при використанні текстового способу взаємодії.

Вихідними даними системи, що формуються в результаті обробки, є такі типи інформації.

1. Текст транскрипції – результат розпізнавання голосового запитання, текстовий рядок українською мовою.

2. Текстова відповідь персонажа – згенерований мовною моделлю текст відповіді від першої особи; повертається у складі JSON-об’єкта.

3. Аудіофайл відповіді – синтезоване озвучення відповіді у форматі mp3; повертається у вигляді URL-адреси для автоматичного відтворення на клієнті.

Уся вхідна та вихідна інформація обробляється в межах одного HTTP-запиту, що спрощує управління станом та забезпечує синхронність формування результату для відвідувача.

2.3.2 Структура інформаційного забезпечення

Логічну модель даних системи утворюють дві взаємопов’язані структури: документ бази знань у форматі JSON, що є джерелом фактів, та колекція векторної бази ChromaDB, що зберігає ембедінги цих фактів для семантичного пошуку.

Текстові документи бази знань зберігаються у файлі exhibits.json, що містить масив об’єктів-експонатів. Кожен об’єкт описує окремий експонат і містить його ідентифікатор, назву, опис, посилання на зображення та масив верифікованих фактів. Структуру одного запису бази знань наведено у таблиці 2.2

Логічну модель колекції ChromaDB утворюють векторні документи, кожен з яких відповідає одному факту з бази знань і містить такі поля: page_content – оригінальний текст факту; metadata.exhibit_id – ідентифікатор персонажа, якому належить факт; embedding – числовий вектор розмірністю 384, отриманий моделлю sentence-transformers. Поле metadata.exhibit_id використовується як фільтр під час пошуку, що гарантує вибірку фрагментів виключно з бази знань обраного персонажа й унеможливлює змішування контексту між різними експонатами.

Таблиця 2.2 – Структура запису бази знань експоната (exhibits.json)

Поле	Призначення
id	Унікальний ідентифікатор експоната для метаданої фільтрації у ChromaDB
name	Назва експоната, що відображається у користувацькому інтерфейсі
description	Короткий опис для картки експоната на головній сторінці
image_url	Шлях до зображення експоната
facts	Масив верифікованих фактів; кожен факт індексується як окремий векторний фрагмент

Стратегія індексування полягає у тому, що кожен окремий факт перетворюється на самостійний векторний документ. Такий підхід, за якого один семантичний фрагмент відповідає одному факту, забезпечує точну вибірку релевантної інформації без зайвого «шуму» у контексті, оскільки модель пошуку оперує атомарними смисловими одиницями, а не цілими абзацами. Структуру даних колекції ChromaDB та зв’язок між документом JSON і векторними записами наведено на рисунку 2.6.

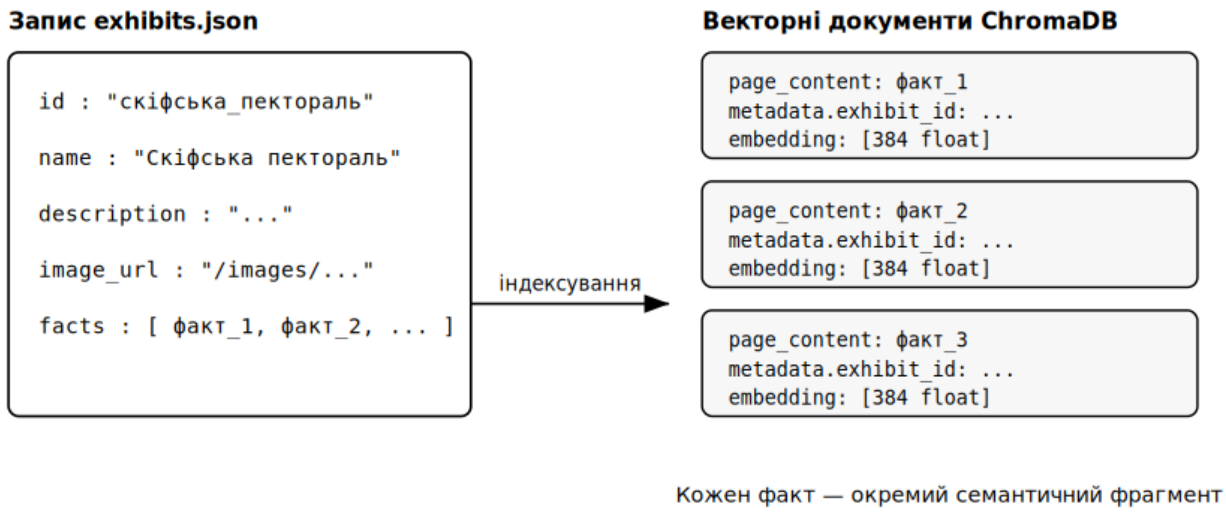

Рисунок 2.6 – Структура векторної бази ChromaDB

Таким чином, у другому розділі розроблено алгоритмічне та інформаційне забезпечення нейромережевої системи інтерактивного діалогу з музейними експонатами. Сформовано загальну клієнт-серверну архітектуру системи, у якій клієнтська частина забезпечує голосову взаємодію з користувачем, а серверна частина виконує розпізнавання мовлення, семантичний пошук, генерацію відповіді та синтез мовлення. Описано алгоритм роботи голосового RAG-конвеєра, механізм перевірки релевантності контексту та спосіб запобігання недостовірним відповідям. Визначено структуру інформаційного забезпечення, що включає базу знань експонатів у форматі JSON та векторне сховище ChromaDB для семантичного пошуку.

3 РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖЕВОЇ СИСТЕМИ ІНТЕРАКТИВНОГО ДІАЛОГУ З МУЗЕЙНИМИ ЕКСПОНАТАМИ

3.1 Реалізація програмного забезпечення

Серверну частину системи реалізовано мовою програмування Python версії 3.10 з використанням асинхронного веб-фреймворку FastAPI. Архітектуру серверного застосунку побудовано за принципом чіткого розподілу відповідальності між окремими модулями, кожен з яких відповідає за конкретний етап обробки голосового запиту.

Файлову структуру проєкту наведено у таблиці 3.1

Таблиця 3.1 – Файлова структура проєкту

Каталог / файл	Призначення
app/main.py	Точка входу, конфігурація застосунку та проміжного ПЗ CORS
app/routes/	Шар маршрутизації HTTP-запитів (кінцеві точки API)
app/speech/	Підсистема автоматичного розпізнавання мовлення (ASR)
app/rag/	Реалізація RAG-конвеєра та інтеграція з векторною базою
app/llm/	Інтеграція з великою мовною моделлю через Groq API
app/tts/	Підсистема синтезу мовлення на основі Edge TTS
app/prompts/	Шаблони системних підказок персонажів
app/database/	Скрипт одноразового індексування векторної бази
app/data/	База знань експонатів у форматі JSON
frontend/	Клієнтський SPA на React 18 + Vite + Tailwind CSS
docker-compose.yml	Опис двох сервісів (backend, frontend) та тому даних

Вибір конкретних інструментів реалізації обґрунтовано порівняльним аналізом доступних альтернатив за критеріями продуктивності, простоти розгортання та відповідності вимогам проєкту. Веб-фреймворк FastAPI обрано перед Django та Flask завдяки нативній підтримці асинхронного виконання

(async/await), що є критичним для конвеєра з кількома послідовними мережевими та обчислювальними операціями, а також завдяки автоматичній генерації інтерактивної документації Swagger UI, яка спрощує тестування програмного інтерфейсу. Векторну базу ChromaDB обрано перед хмарними рішеннями Pinecone та Weaviate з огляду на можливість повністю локального розгортання без хмарної залежності та нативну інтеграцію з LangChain, що відповідає вимозі автономності музейного стенда. Клієнтську бібліотеку React обрано перед Vue та Angular завдяки розвиненій екосистемі, широкій підтримці браузерного MediaRecorder API та зручності побудови адаптивних односторінкових застосунків.

Загальну структуру взаємодії програмних модулів серверної частини наведено на рисунку 3.1: модуль маршрутизації приймає запит і послідовно делегує його обробку підсистемам розпізнавання мовлення, семантичного пошуку, генерації та синтезу, а скрипт індексування одноразово наповнює векторну базу фактами з бази знань.

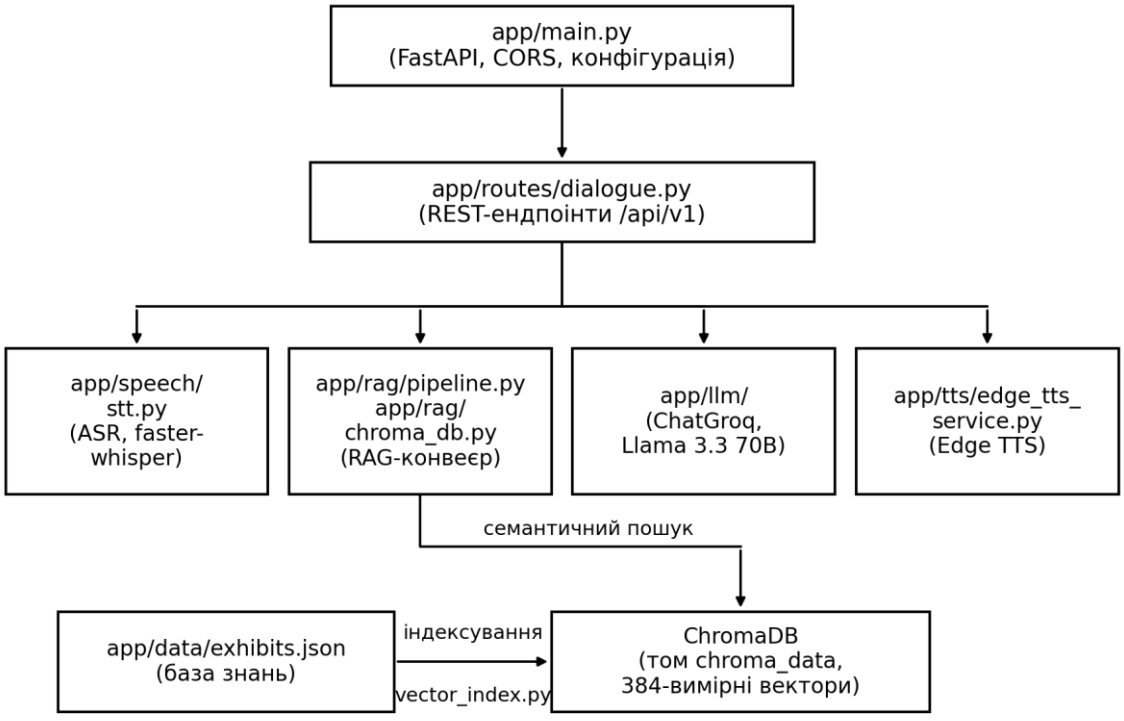

Рисунок 3.1 – Структура взаємодії програмних модулів серверної частини

Під час запуску застосунку FastAPI завантажуються змінні оточення з файлу

.env, конфігурується проміжне програмне забезпечення CORS для взаємодії з клієнтським застосунком та підключається маршрутизатор діалогу з префіксом /api/v1 (код наведено у додатку Б).

Центральним елементом серверної частини є модуль маршрутизації app/routes/dialogue.py, який реалізує чотири кінцеві точки програмного інтерфейсу: GET /exhibits – повертає перелік доступних експонатів; POST /chat/text – обробляє текстовий запит; POST /chat/audio – обробляє голосовий запит; GET /audio/{filename} – віддає клієнту згенерований аудіофайл. Основним для голосового діалогу є обробник /chat/audio, який послідовно виконує весь конвеєр обробки в межах одного HTTP-запиту: збереження завантаженого аудіо у тимчасовій директорії, розпізнавання мовлення, формування відповіді RAG-конвеєром, синтез mp3 та повернення JSON-об’єкта з результатом (код наведено у додатку Б).

Підсистему автоматичного розпізнавання мовлення реалізовано у модулі app/speech/stt.py на основі бібліотеки faster-whisper. Модель розміру small ініціалізується одноразово під час запуску застосунку з параметрами device="cpu" та compute_type="int8", що завдяки квантизації ваг знижує споживання оперативної пам’яті та пришвидшує виведення приблизно у чотири рази без відчутної втрати точності. Розпізнавання виконується з явним зазначенням мови uk та параметром beam_size=5. Оскільки розпізнавання є синхронною обчислювально-інтенсивною операцією, її винесено в окремий потік через run_in_executor, аби не блокувати асинхронний цикл подій FastAPI (код наведено у додатку Б).

Векторне сховище реалізовано у модулі app/rag/chroma_db.py на основі бази даних ChromaDB у режимі персистентного клієнта, що зберігає вектори та метадані локально у директорії файлової системи. Семантичне кодування тексту виконується моделлю sentence-transformers paraphrase-multilingual-MiniLM-L12-v2, яка підтримує українську мову та формує 384-вимірні вектори. Інтеграцію з ChromaDB забезпечує бібліотека LangChain через обгортку Chroma, що надає уніфікований інтерфейс retriever для семантичної вибірки (код наведено у додатку Б).

Логіку RAG-конвеєра реалізовано у модулі app/rag/pipeline.py з

використанням механізму LangChain Expression Language. Для заданого ідентифікатора експоната створюється retriever з метаданим фільтром за полем exhibit_id, що гарантує вибірку фрагментів виключно з бази знань обраного персонажа. Із колекції відбирається фіксована кількість найрелевантніших фрагментів за косинусною подібністю, які форматуються у єдиний блок контексту та підставляються у шаблон підказки разом із запитанням користувача, після чого ланцюжок послідовно передає результат до великої мовної моделі й парсера рядкового виводу (код наведено у додатку Б).

Як велику мовну модель обрано Llama 3.3 (70B), доступну через Groq API, що завдяки спеціалізованим LPU-чіпам забезпечує низьку затримку генерації. Інтеграцію реалізовано у модулі app/llm за допомогою обгортки ChatGroq бібліотеки LangChain. Параметр temperature=0.7 задає помірний рівень варіативності відповідей, а max_tokens=512 обмежує їхню довжину для збереження стислості та контролю часу генерації. Поведінку персонажа задано системною підказкою, що містить три ключові інструкції: визначення ролі (відповідати від першої особи від імені артефакта), обмеження джерела знань (спиратися виключно на наданий контекст) та правило поведінки за відсутності інформації (чесно повідомити, що відповідь невідома) (код наведено у додатку Б).

Синтез мовлення реалізовано у модулі app/tts/edge_tts_service.py за допомогою бібліотеки Edge TTS. Для озвучення відповідей застосовується українськомовний голос uk-UA-OstapNeural. Асинхронна функція отримує текст відповіді, генерує аудіопотік і зберігає його у файл формату mp3, який згодом віддається клієнту для автоматичного відтворення (код наведено у додатку Б).

База знань зберігається у файлі app/data/exhibits.json, структуру якого описано у підрозділі 2.3.2. Скрипт індексування app/database/vector_index.py виконується одноразово: він зчитує JSON, перетворює кожен окремий факт на самостійний документ LangChain із метаданим полем exhibit_id, після чого генерує ембединги та зберігає їх у колекцію ChromaDB (код наведено у додатку Б). На поточному етапі база знань містить три музейні експонати, що репрезентують різні історичні епохи України: золоту скіфську пектораль IV ст. до н.е., бойовий меч гетьмана Петра Конашевича-Сагайдачного XVII ст. та керамічний глечик

трипільської культури (5400–2700 рр. до н.е.).

Для забезпечення портативності та відтворюваності розгортання систему упаковано у Docker-контейнери за допомогою Docker Compose. Конфігурація визначає два сервіси: backend та frontend. Серверний образ будується на базі python:3.10-slim, встановлює системну залежність FFmpeg для конвертації аудіо, інсталює Python-залежності, а також на етапі збирання попередньо завантажує нейромережеві моделі та виконує одноразове індексування векторної бази, що гарантує швидкий старт контейнера. Клієнтський образ будується на базі node:18-alpine. Дані векторної бази зберігаються у Docker-томі chroma_data, що забезпечує їхнє збереження між перезапусками.

Програмні залежності системи наведено у таблиці 3.2

Таблиця 3.2 – Програмні залежності системи

Бібліотека / технологія	Призначення у проєкті
1	2
fastapi, uvicorn	Асинхронний REST API та ASGI-сервер
faster-whisper	Розпізнавання українського мовлення на CPU
langchain, langchain-chroma, langchain-groq	Оркестрація RAG-конвеєра та інтеграція компонентів
chromadb	Локальне векторне сховище ембедингів
sentence-transformers	Семантичне кодування тексту в ембединги
edge-tts	Синтез українського мовлення у формат mp3
python-multipart	Обробка multipart/form-data із аудіофайлом
react, vite, tailwindcss	Клієнтський SPA, збирання та стилізація

3.2 Інтерфейс користувача

Користувацький інтерфейс є ключовою ланкою взаємодії відвідувача із системою, тому до нього висуваються вимоги інтуїтивності, мінімалізму та адаптивності. Оскільки система призначена для використання на музейних сенсорних стендах та мобільних пристроях відвідувачів, інтерфейс має бути зрозумілим з першого погляду й не вимагати від користувача спеціальних навичок.

Клієнтську частину реалізовано як односторінковий застосунок (Single Page Application, SPA) на основі бібліотеки React 18 із системою збирання Vite та утилітарним CSS-фреймворком Tailwind CSS. Для анімації елементів інтерфейсу використано бібліотеку Framer Motion, для іконок – Lucide React, а для маршрутизації між сторінками – React Router.

Застосунок складається з двох основних сторінок. Головна сторінка (Home.jsx) відображає каталог доступних експонатів у вигляді адаптивної сітки карток. Під час завантаження сторінка асинхронно запитує перелік експонатів із серверного ендпоінту GET /api/v1/exhibits і відображає індикатор завантаження до отримання відповіді. Кожен експонат представлений компонентом ExhibitCard, що містить зображення, назву, короткий опис та посилання для переходу до діалогу.

Інтерфейс головної сторінки у стані вибору експоната наведено на рисунку 3.2.

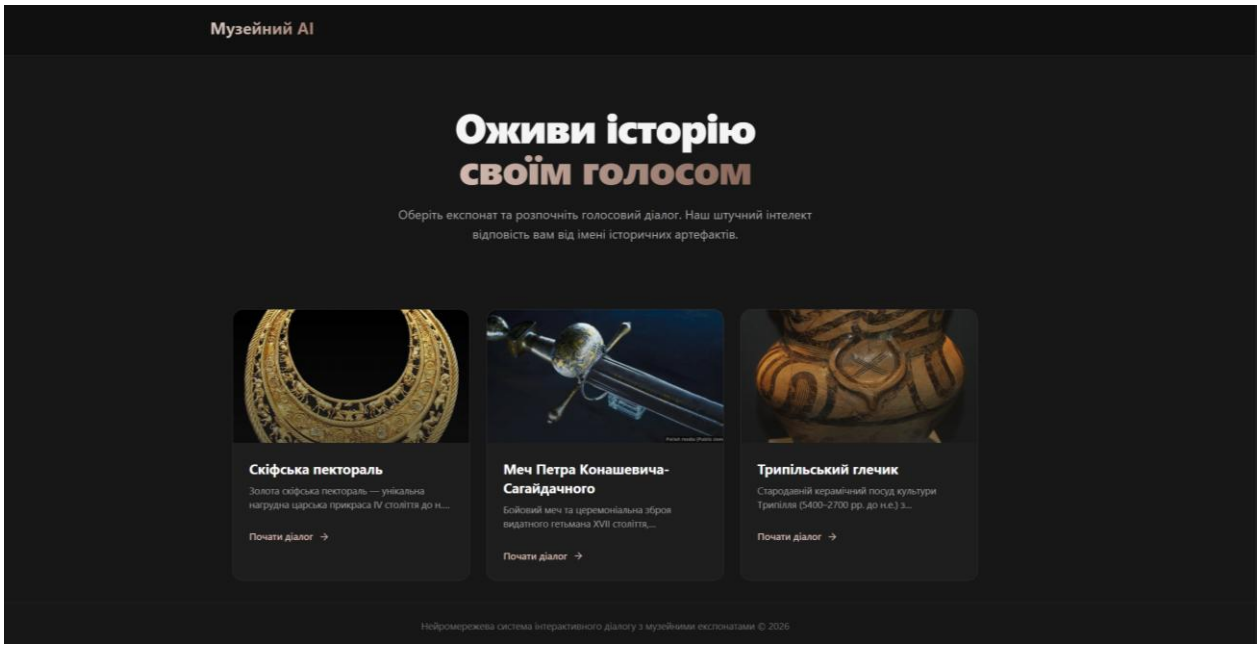

Рисунок 3.2 – Інтерфейс головної сторінки: вибір музейного експоната

Серед застосованих UX-рішень головної сторінки слід відзначити адаптивну сітку карток, що автоматично перебудовує кількість стовпців залежно від ширини екрана, забезпечуючи читабельність як на широкому стенді, так і на мобільному пристрої. Кожна картка має зображення-прев’ю та лаконічний опис, що дозволяє відвідувачу швидко зорієнтуватися у складі експозиції. До завершення

завантаження даних відображається анімований індикатор, що інформує користувача про процес отримання переліку експонатів і запобігає враженню «зависання» інтерфейсу.

Сторінка діалогу є центральним екраном взаємодії. Вона містить заголовок з мініатюрою та назвою обраного експоната, область історії повідомлень, текстове поле введення та кнопку голосового запису. Інтерфейс підтримує два способи введення: текстовий – через поле вводу, і голосовий – через кнопку мікрофона. Повідомлення користувача та відповіді експоната відображаються у вигляді діалогових «бульбашок» із візуальним розрізненням сторін розмови.

Інтерфейс сторінки діалогу наведено на рисунку 3.3.

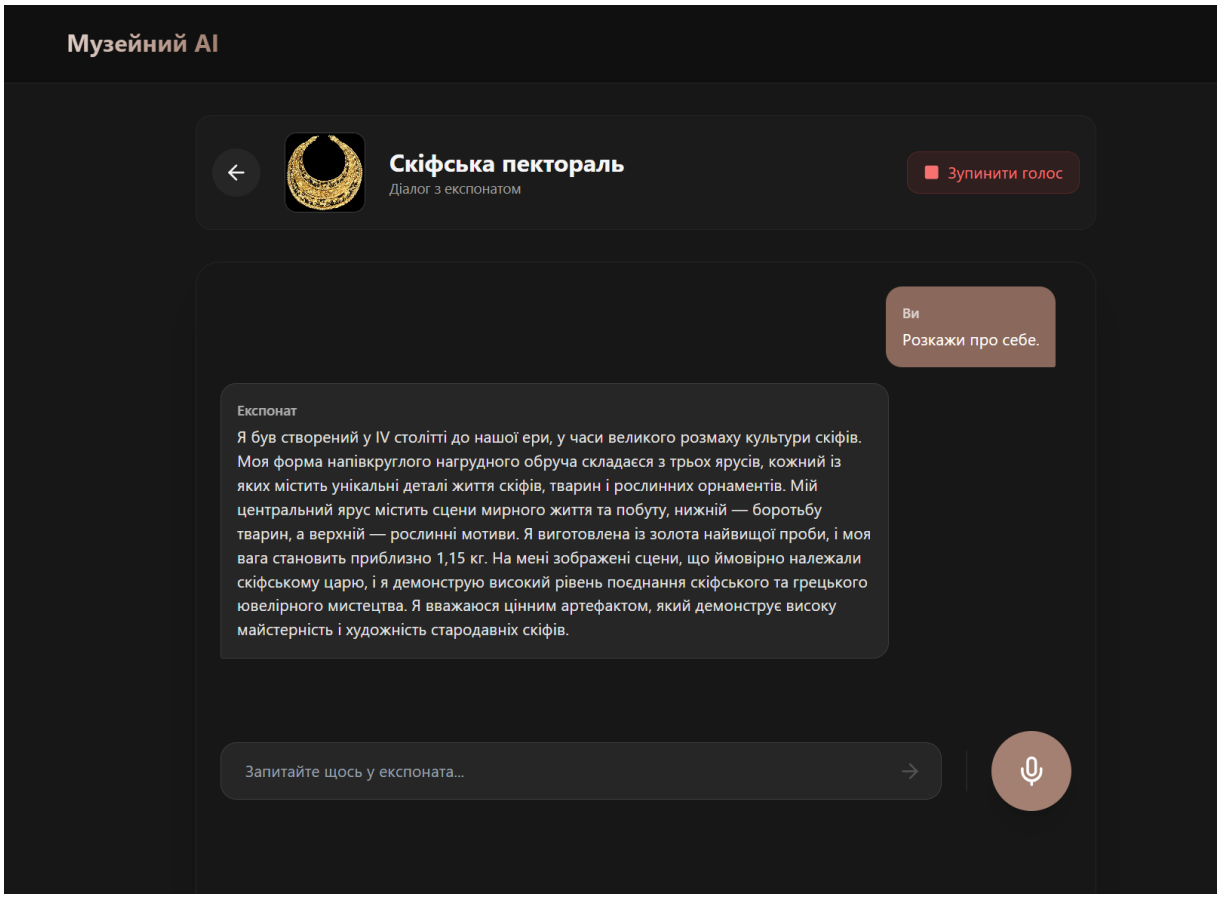

Рисунок 3.3 – Інтерфейс сторінки діалогу з експонатом

UX-рішення сторінки діалогу спрямовані на природність голосової взаємодії. Повідомлення оформлено у вигляді діалогових бульбашок із вирівнюванням за сторонами розмови, що інтуїтивно відображає чергування реплік відвідувача й персонажа. Кнопка мікрофона має два чітко розрізнювані візуальні стани –

очікування та активного запису – причому під час запису відображається анімований пульсуючий індикатор, що підтверджує захоплення звуку. У період обробки запиту демонструється анімований індикатор «Аналізую запит...», який повідомляє користувача про виконання конвеєра і знижує відчуття затримки.

Голосове введення реалізовано із застосуванням браузерного MediaRecorder API. При натисканні кнопки мікрофона запитується доступ до мікрофона, після чого розпочинається запис аудіопотоку; по завершенні зібрані фрагменти об’єднуються у Blob формату audio/webm і передаються на сервер (код наведено у додатку Б, лістинг Б.10). Взаємодію з серверним API інкапсульовано у модулі services/api.js на основі бібліотеки axios, що реалізує функції отримання переліку експонатів, надсилання текстового та голосового запитів. Отриманий від сервера аудіофайл автоматично відтворюється засобами HTML5 Audio API (код наведено у додатку Б, лістинг Б.11).

Таким чином, побудований інтерфейс реалізує повний цикл голосової взаємодії: вибір експоната, запис голосового запитання, відображення транскрипції та текстової відповіді, а також автоматичне відтворення синтезованого аудіо. Адаптивне компонування на основі Tailwind CSS забезпечує коректне відображення як на широких екранах сенсорних стендів, так і на мобільних пристроях відвідувачів.

3.3 Тестування програмного забезпечення

Метою тестування розробленого програмного забезпечення була перевірка коректності функціонування окремих підсистем, правильності їхньої взаємодії у складі голосового RAG-конвеєра, а також оцінювання якісних і часових характеристик системи. Тестування проводилося у три етапи: функціональне тестування окремих підсистем, інтеграційне тестування повного конвеєра та оцінювання нефункціональних характеристик (час відгуку, точність розпізнавання, релевантність відповідей).

На першому етапі перевірено працездатність кожної підсистеми окремо. Програмний інтерфейс зручно тестувати через автоматично згенеровану

документацію Swagger UI, доступну за адресою /docs. Для підсистеми розпізнавання мовлення перевірено коректність транскрипції українськомовних аудіозаписів різної тривалості та тематики. Для RAG-конвеєра перевірено, що метаданий фільтр за exhibit_id коректно обмежує вибірку фрагментами лише обраного експоната. Для підсистеми синтезу мовлення перевірено генерацію коректних mp3-файлів та їхнє відтворення у браузері.

Результати функціонального тестування підсистем наведено у таблиці 3.3

Таблиця 3.3 – Результати функціонального тестування підсистем

№	Сценарій тестування	Очікуваний результат	Статус
1	Отримання переліку експонатів (GET /exhibits)	JSON зі списком із трьох експонатів	Пройдено
2	Транскрипція голосового запиту українською	Коректний текст транскрипції	Пройдено
3	Вибірка контексту з фільтром за exhibit_id	Фрагменти лише обраного експоната	Пройдено
4	Генерація відповіді від першої особи	Відповідь у ролі персонажа за контекстом	Пройдено
5	Синтез mp3 та автовідтворення на клієнті	Аудіофайл генерується й відтворюється	Пройдено
6	Запит поза межами бази знань	Ввічлива відмова без вигаданих фактів	Пройдено

На другому етапі перевірено наскрізну роботу повного голосового конвеєра – від запису голосового запитання до автоматичного відтворення синтезованої відповіді. Сценарій тестування полягав у тому, що тестувальник обирав експонат, натискав кнопку мікрофона, ставив запитання природною мовою та очікував озвучену відповідь. Перевірялися такі аспекти: коректність послідовного проходження запиту через усі підсистеми, відповідність транскрипції сказаному, релевантність та фактологічна точність відповіді відносно бази знань, а також якість синтезованого мовлення.

Приклад успішного діалогу з експонатом «Скіфська пектораль» наведено у таблиці 3.4

На третьому етапі оцінено часові та якісні характеристики системи на

стандартному обладнанні без GPU. Виміряно середній час виконання окремих стадій конвеєра для типового голосового запиту. Найбільш ресурсомісткими стадіями є розпізнавання мовлення та генерація відповіді мовною моделлю. Завдяки квантизації моделі faster-whisper до INT8 та використанню Groq API із малим часом відгуку повного конвеєра залишається у межах кількох секунд, що є прийнятним для інтерактивного музейного застосування.

Таблиця 3.4 – Приклад наскрізного діалогу з експонатом

Етап діалогу	Зміст
Запитання відвідувача (голос)	«Коли і де тебе знайшли?»
Транскрипція (faster-whisper)	«Коли і де тебе знайшли?»
Вибраний контекст (ChromaDB)	Факт про знахідку 21 червня 1971 р. археологом Б. Мозолевським у кургані Товста Могила
Відповідь експоната (Llama 3.3)	Розгорнута відповідь від першої особи з опорою на знайдений факт, без вигаданих деталей
Озвучення (Edge TTS)	mp3-файл голосом uk-UA-OstapNeural, відтворено автоматично

Усереднені результати вимірювань наведено у таблиці 3.5

Таблиця 3.5 – Час виконання стадій голосового конвеєра

Стадія конвеєра	Технологія	Час, с
Розпізнавання мовлення (STT)	faster-whisper	≈ 1,4
Семантичний пошук	ChromaDB	≈ 0,4
Генерація відповіді (LLM)	Groq / Llama 3.3	≈ 1,1
Стадія конвеєра	Технологія	Час, с
Синтез мовлення (TTS)	Edge TTS	≈ 1,5
Загальний час відгуку	–	≈ 3–5

Для оцінювання точності розпізнавання мовлення використано метрику Word Error Rate (WER), визначену у формулі 2.1 На наборі тестових українськомовних запитів модель faster-whisper розміру small продемонструвала прийнятний рівень точності, достатній для коректної семантичної вибірки навіть за наявності окремих помилок транскрипції, оскільки семантичний пошук на основі ембедингів є стійким до незначних відмінностей у формулюванні. Якість відповідей оцінювалася експертно за критеріями релевантності, фактологічної точності та дотримання рольового стилю персонажа. У переважній більшості випадків система формувала змістовні відповіді, що спиралися виключно на верифіковані факти бази знань, а у разі відсутності релевантної інформації коректно повідомляла про це без генерації вигаданих фактів.

Проведене тестування підтвердило працездатність розробленого програмного забезпечення та коректність взаємодії всіх підсистем у складі єдиного голосового RAG-конвеєра. Система забезпечує природний україномовний голосовий діалог з музейними експонатами на стандартному CPU-обладнанні без потреби у дорогих GPU-прискорювачах, що підтверджує досяжність поставлених у роботі завдань.

Отже, у третьому розділі виконано практичну реалізацію нейромережевої системи інтерактивного діалогу з музейними експонатами. Реалізовано серверну частину на основі FastAPI, компоненти ASR, RAG, LLM і TTS, а також клієнтський веб-інтерфейс для вибору експоната, запису голосового запиту та відтворення відповіді. Проведено тестування основних функцій системи, зокрема розпізнавання мовлення, пошуку релевантного контексту, генерації текстової відповіді та синтезу аудіофайлу. Результати тестування підтвердили працездатність розробленого програмного забезпечення, коректність обробки запитів у межах бази знань та можливість використання системи як прототипу інтерактивного музейного AI-асистента.

ВИСНОВКИ

Метою кваліфікаційної роботи була розробка та програмна реалізація нейромережевої системи інтерактивного голосового діалогу з музейними експонатами на основі RAG-архітектури.

У процесі виконання роботи вирішено такі основні завдання:

1. Проведено порівняльний аналіз десяти реалізацій інтерактивних музейних AI-систем (Museum of Zoology Cambridge, Warsaw Artistic Chatbot, Centre Pompidou/Ask Mona та ін.) за критеріями концепції персонажного діалогу, технологічного стеку та відкритості архітектури. Встановлено, що найближчі аналоги реалізовані на закритих комерційних платформах без відтворюваної документації, що обґрунтовує доцільність розробки відкритого рішення на сучасному NLP-стеку.

2. Обґрунтовано вибір технологічного стеку (faster-whisper + LangChain + ChromaDB + Groq API + Edge TTS) шляхом аналізу технічних характеристик за критеріями якості, швидкодії та вимог до апаратного забезпечення. Це визначило конфігурацію системи, що не потребує GPU і придатна для розгортання на стандартному музейному обладнанні.

3. Розроблено дворівневу клієнт-серверну архітектуру та спроєктовано п'ять підсистем: ASR, RAG, генерація LLM, TTS і персистентність бази. Введено метрику WER (формула 2.1) для оцінки ASR та формулу косинусної подібності (формула 2.2) як математичну основу семантичного пошуку. Визначена exhibit_id-фільтрація унеможливлює змішування контексту між персонажами і є основою фактологічної точності відповідей.

4. Реалізовано повнофункціональне програмне забезпечення у вигляді серверного конвеєра (FastAPI, ASR, RAG із пороговим відсіканням, TTS) та адаптивного React-застосунку з голосовим введенням, упаковане у Docker-контейнери. Це забезпечило відтворюваність розгортання та можливість розширення системи новими персонажами без змін у програмному коді, а також підтвердило працездатність усіх підсистем у складі єдиного конвеєра під час функціонального та інтеграційного тестування.

5. Практична цінність роботи полягає у тому, що розроблена система демонструє досяжність якісного україномовного голосового діалогового інтерфейсу без залежності від дорогих комерційних хмарних рішень. Система готова до використання у музейних експозиціях у вигляді сенсорних стендів або вебзастосунків і може бути адаптована для інших галузей – освіти, архівної справи та туристичного сервісу. Подальші дослідження доцільно спрямувати на розширення бази знань, підтримку багатоходових діалогів із пам’яттю контексту та інтеграцію додаткових мов розпізнавання та синтезу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OpenAI. GPT-4o documentation and API integration methods. 2025. URL: https://platform.openai.com/docs (дата звернення: 12.05.2026)
2. Google DeepMind. Gemini 1.5 Pro technical documentation: architecture and capabilities. 2025. URL: https://ai.google.dev (дата звернення: 12.05.2026)
3. Radford A., Kim J. W., Xu T. et al. Robust Speech Recognition via Large-Scale Weak Supervision. arXiv:2212.04356. 2022. URL: https://arxiv.org/abs/2212.04356 (дата звернення: 12.05.2026)
4. Squeezeformer-CTC XS (uk-UA): модель розпізнавання українського мовлення. 2024. URL: https://huggingface.co/theodotus/stt_uk_squeezeformer_ctc_xs (дата звернення: 12.05.2026)
5. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401. 2020. URL: https://arxiv.org/abs/2005.11401 (дата звернення: 12.05.2026)
6. faster-whisper: швидка реалізація моделі Whisper з використанням CTranslate2. 2024. URL: https://github.com/SYSTRAN/faster-whisper (дата звернення: 11.05.2026)
7. edge-tts: бібліотека синтезу мовлення на основі Microsoft Edge TTS. 2024. URL: https://github.com/rany2/edge-tts (дата звернення: 11.05.2026)
8. LangChain. Офіційна документація фреймворку. 2024. URL: https://docs.langchain.com (дата звернення: 11.05.2026)
9. Groq. Офіційна документація API. 2024. URL: https://console.groq.com/docs (дата звернення: 12.05.2026)
10. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084. 2019. URL: https://arxiv.org/abs/1908.10084 (дата звернення: 12.05.2026)
11. ChromaDB. Офіційна документація векторної бази даних. 2024. URL: https://docs.trychroma.com (дата звернення: 12.05.2026)
12. FastAPI. Офіційна документація вебфреймворку. 2024. URL: https://fastapi.tiangolo.com (дата звернення: 12.05.2026)

13. React. Офіційна документація бібліотеки. 2024. URL: https://react.dev (дата звернення: 12.05.2026)

14. Vite. Офіційна документація системи збирання. 2024. URL: https://vitejs.dev (дата звернення: 12.05.2026)

15. Tailwind CSS. Офіційна документація CSS-фреймворку. 2024. URL: https://tailwindcss.com/docs (дата звернення: 16.05.2026)

16. Docker. Офіційна документація платформи контейнеризації. 2024. URL: https://docs.docker.com

17. MDN Web Docs. MediaRecorder API. 2024. URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder (дата звернення: 16.05.2026)

18. Meta AI. Llama 3.3 model card and documentation. 2024. URL: https://www.llama.com (дата звернення: 16.05.2026)

19. Schaffer S., Reitberger W. Conversational Agents in Museums: A Systematic Review of Design and Evaluation. International Journal of Human-Computer Studies. 2023. Vol. 175. P. 103–121.

20. Gao Y., Xiong Y., Gao X. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997. 2024. URL: https://arxiv.org/abs/2312.10997 (дата звернення: 16.05.2026)

21. Vassos S., Malliaraki E., dal Falco F. et al. Art-Bots: Toward Chat-Based Conversational Experiences in Museums. Interactive Storytelling: 9th International Conference. Cham: Springer, 2016. P. 433–437.

22. Hoy M. B. Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. Medical Reference Services Quarterly. 2018. Vol. 37(1). P. 81–88.

23. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008.

24. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008. 506 p.

25. Ji Z., Lee N., Frieske R. et al. Survey of Hallucination in Natural Language Generation. ACM Computing Surveys. 2023. Vol. 55(12). P. 1–38.

26. Navarrete T. Digital Heritage Tourism: Innovations in Museums. World Leisure Journal. 2019. Vol. 61(3). P. 200–214.

27. Museum of Zoology, University of Cambridge. Talking museum specimens powered by AI. 2024. URL: https://www.museum.zoo.cam.ac.uk (дата звернення: 15.05.2026)

28. Ask Mona. AI assistant for cultural institutions. 2024. URL: https://askmona.fr (дата звернення: 16.05.2026)

29. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781. 2013. URL: https://arxiv.org/abs/1301.3781 (дата звернення: 16.05.2026)

30. Wang W., Wei F., Dong L. et al. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957. 2020. URL: https://arxiv.org/abs/2002.10957 (дата звернення: 16.05.2026)

31. Shanahan M., McDonell K., Reynolds L. Role-Play with Large Language Models. Nature. 2023. Vol. 623. P. 493–498.

32. Костюкевич Н.Ю. Нейромережева система інтерактивного голосового діалогу з музейними експонатами на основі архітектури RAG. Інтелектуальні комп'ютерні системи та мережі» (ІКСМ) : матеріали IV Всеукраїнської науково-практичної конференції молодих вчених і студентів (м. Тернопіль, ЗУНУ, 20 травня 2026 р.). Тернопіль, 2026. С. 179–181. URL: https://ki.wunu.edu.ua/conference/archive/2026_1.pdf. (дата звернення: 20.05.2026)

33. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

34. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

35. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd ed. draft. Stanford University, 2025. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 16.05.2026)

36. Kucia F. J., Grabek B., Trochimiak S. D., Wróblewska A. How to Make Museums More Interactive? Case Study of Artistic Chatbot. arXiv:2509.00572. 2025. URL: https://arxiv.org/abs/2509.00572 (дата звернення: 16.05.2026)

37. Trichopoulos G., Konstantakis M., Alexandridis G., Caridakis G. Crafting a Museum Guide Using ChatGPT4. Big Data and Cognitive Computing. 2023. Vol. 7(3). Article 148. DOI: 10.3390/bdcc7030148

38. Karpukhin V., Oguz B., Min S. et al. Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020. P. 6769–6781.

39. Touvron H., Lavril T., Izacard G. et al. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971. 2023. URL: https://arxiv.org/abs/2302.13971 (дата звернення: 18.05.2026)

40. Brown T., Mann B., Ryder N. et al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877–1901.

41. University of Cambridge. Public invited to chat to museum animals in novel AI experiment. 2024. URL: https://www.cam.ac.uk/research/news/public-invited-to-chat-to-museum-animals-in-novel-ai-experiment (дата звернення: 17.05.2026)

42. Thiel S. Talk to me! Chatbots in Museums. ZKM | Center for Art and Media Karlsruhe. 2020. URL: https://zkm.de/en/talk-to-me-chatbots-in-museums (дата звернення: 17.05.2026)

Додаток А

Апробація отриманих результатів

Костюкевич Н.Ю.
студент 4 курсу ФКІТ ЗУНУ
Науковий керівник д.т.н., професор Комар М.П., кафедра ІОСУ ЗУНУ

НЕЙРОМЕРЕЖЕВА СИСТЕМА ІНТЕРАКТИВНОГО ГОЛОСОВОГО ДІАЛОГУ З МУЗЕЙНИМИ ЕКСПОНАТАМИ НА ОСНОВІ АРХІТЕКТУРИ RAG

Вступ. Сучасний розвиток інформаційних технологій і систем штучного інтелекту суттєво змінює підходи до організації музейного простору. Водночас традиційні формати взаємодії з відвідувачами дедалі частіше демонструють свої обмеження. Класичні аудіогіди, сенсорні панелі та інформаційні таблички працюють у режимі односторонньої передачі інформації, де користувач фактично залишається пасивним споживачем заздалегідь підготовленого матеріалу. За такого підходу відсутня можливість поставити уточнювальне запитання й отримати відповідь у режимі реального часу [1, 2].

Поява великих мовних моделей (Large Language Models, LLM) та сучасних методів обробки природної мови (NLP) створила передумови для реалізації повноцінного природномовного діалогу безпосередньо в музейному середовищі. Однак пряме застосування LLM без опори на перевірені джерела інформації призводить до виникнення так званих «галюцинацій» — генераування переконливих, але фактично помилкових відповідей. Для культурно-освітньої сфери, у якій система сприймається як офіційне джерело даних, така поведінка є критично небажаною. Архітектура Retrieval-Augmented Generation (RAG) дає змогу усунути цю проблему завдяки жорсткій прив'язці генерації відповіді до фрагментів верифікованої бази знань [5].

Постановка задачі. Аналіз наявних рішень у сфері інтерактивних музейних AI-систем показав, що більшість із них реалізовано на закритих або комерційних платформах без відкритої технічної документації. Проєкти Museum of Zoology (Cambridge) та Artistic Chatbot (Warsaw Academy of Fine Arts) підтверджують актуальність голосового персонажного діалогу, проте не надають відтворюваного рішення, побудованого на сучасному NLP-стеку. Крім того, системи, орієнтовані виключно на текстову взаємодію або централізовані хмарні обчислення, не відповідають умовам роботи музейних сенсорних стендів, де GPU-обладнання зазвичай недоступне. Об'єктом дослідження є процес автоматизованої обробки природномовних голосових запитів у межах інтерактивних музейних систем. Предмет дослідження охоплює методи, моделі та програмні засоби побудови голосового RAG-конвеєра на базі відкритих NLP-технологій. Метою роботи є проєктування та реалізація нейромережевої системи інтерактивного голосового діалогу з музейними експонатами, здатної забезпечувати розпізнавання й синтез українського мовлення на CPU-обладнанні, семантичний пошук у верифікованій базі знань і генерацію відповіді від імені обраного персонажа.

Основний матеріал. Узагальнену архітектуру програмної системи спроєктовано як сукупність кількох взаємопов'язаних модулів.

Першим функціональним компонентом виступає модуль клієнтського інтерфейсу. Його основне призначення полягає у захопленні голосового введення відвідувача через браузерний MediaRecorder API та передачі бінарного аудіопотоку у форматі WebM на серверну частину за допомогою асинхронного HTTP-запиту. Клієнтський застосунок реалізовано як SPA на базі React 18 із використанням Vite для збирання та Tailwind CSS для стилізації інтерфейсу. AI-обробка на стороні клієнта не виконується. Фактично клієнтська частина виконує роль тонкого інтерфейсного шару, який також забезпечує відтворення MP3-файлу відповіді через стандартний браузерний Audio API.

Другим компонентом системи є модуль приймання та оркестрації запитів. Його реалізовано на основі асинхронного вебфреймворку FastAPI, що працює на ASGI-сервері Uvicorn і забезпечує неблокувальне виконання всіх I/O-операцій pipeline. Модуль приймає multipart-запит із аудіофайлом та ідентифікатором персонажа, маршрутизує його до відповідних сервісів і координує послідовність подальших етапів обробки. Центральним

менеджером конвеєра виступає фреймворк LangChain, який забезпечує уніфіковані механізми обміну даними між AI-компонентами системи.

Третім і ключовим компонентом є модуль розпізнавання мовлення (ASR). Його побудовано на бібліотеці faster-whisper - оптимізованій реалізації моделі Whisper від OpenAI на рушії CTranslate2. Модуль виконує транскрипцію аудіо виключно на CPU та не потребує GPU-прискорення. Використання int8-квантизації ваг моделі small (244 млн параметрів) дало змогу зменшити обсяг оперативної пам'яті та пришвидшити інференс без статистично значущого зниження точності розпізнавання українського мовлення. Перед етапом транскрипції WebM-аудіо обов'язково конвертується у формат WAV із частотою дискретизації 16 кГц засобами FFmpeg.

Четвертим компонентом виступає модуль семантичного пошуку (RAG). Розпізнаний текст запиту перетворюється на числовий вектор за допомогою embedding-моделі Sentence Transformers. Далі векторна база ChromaDB здійснює пошук найближчих фрагментів бази знань за метрикою косинусної подібності в іменованій колекції обраного персонажа. Ключовим елементом цього модуля є механізм перевірки релевантності: отриманий коефіцієнт подібності порівнюється з попередньо встановленим пороговим значенням. Якщо жоден із фрагментів не перевищує визначений поріг, система формує ввічливу відмову без звернення до LLM, що дає змогу превентивно блокувати появу галюцинацій. Саме цей підхід принципово відрізняє систему від прямого LLM-промптингу.

Алгоритм функціонування голосового RAG-конвеєру наведено на рисунку 1.

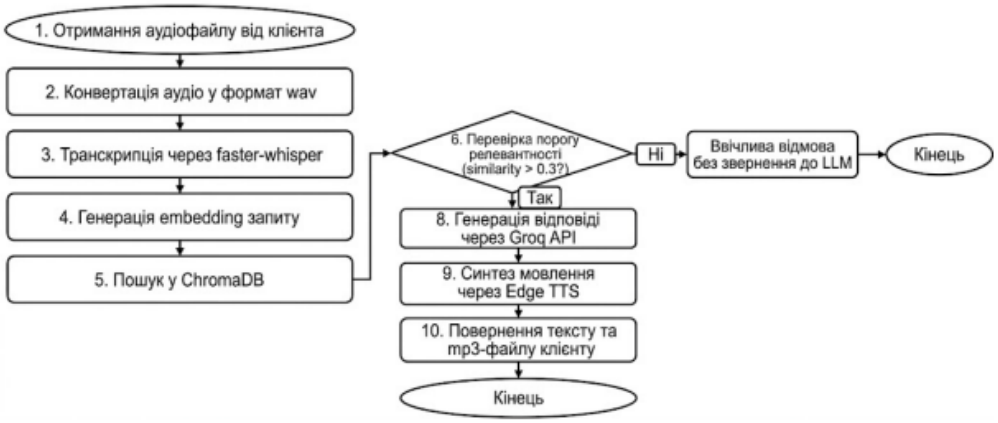

Рисунок 1 - Алгоритм функціонування голосового RAG-конвеєру

П'ятим компонентом є модуль генерації відповіді. Якщо семантичний пошук завершується успішно, знайдені фрагменти включаються до розширеного промпту разом із системним описом персонажа та запитанням користувача. Після цього промпт передається до генеративної моделі Llama 3.3 (70B) через хмарний inference API сервісу Groq, апаратна архітектура LPU якого забезпечує затримку генерації в межах 0.3–0.8 секунди. Використання параметрів temperature 0.7 та обмеження у 512 токенів дозволяє зберігати природність і варіативність відповіді без втрати стилістики персонажа [9].

Шостим компонентом є модуль синтезу мовлення (TTS). Згенерована текстова відповідь передається до бібліотеки Edge TTS, яка надає доступ до нейромережевих голосових рушіїв Microsoft Azure без необхідності оформлення платної підписки. Модуль асинхронно створює MP3-файл із природним українським мовленням голосом uk-UA-PolinaNeural та зберігає його у директорії статичних ресурсів із унікальним UUID-ідентифікатором. У відповідь клієнт отримує JSON-структуру з текстом відповіді та URL-адресою аудіофайлу для автоматичного відтворення. Повний час виконання pipeline на CPU-обладнанні становить у середньому 3–5 секунд.

Увесь програмний комплекс контейнеризовано за допомогою Docker та Docker Compose із виокремленням двох сервісів – фронтенду та бекенду. Векторна база ChromaDB зберігається між перезапусками через іменований том. Додавання нового персонажа не потребує внесення змін до програмного коду. Достатньо підготувати текстові документи та одноразово виконати скрипт індексування.

Висновки. У межах роботи спроєктовано модульну архітектуру системи, яка охоплює клієнтський інтерфейс захоплення голосу, серверну оркестрацію, модулі розпізнавання мовлення, семантичного пошуку у базі знань, генерації відповіді LLM та синтезу мовлення. Запропоновано реалізацію ASR-модуля на базі faster-whisper із використанням int8-квантизації, що забезпечує ефективну транскрипцію українського мовлення виключно на CPU-обладнанні.

Встановлено, що механізм порогового відсікання у модулі RAG є ключовим засобом запобігання галюцинаціям LLM. Під час тестування на запитаннях, відсутніх у базі знань, система жодного разу не сформувала сфабрикованої інформації. Загальний час відгуку в межах 3–5 с відповідає вимогам інтерактивних музейних стендів.

Список літератури

1. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W., Rocktäschel T., Riedel S., Kiela D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2020. URL: https://arxiv.org/abs/2005.11401
2. Gao Y., Xiong Y., Gao X. та ін. Retrieval-Augmented Generation for Large Language Models: A Survey. 2024. URL: https://arxiv.org/abs/2312.10997
3. Radford A., Kim J. W., Xu T., Brockman G., McLeavey C., Sutskever I. Robust Speech Recognition via Large-Scale Weak Supervision. 2022. URL: https://arxiv.org/abs/2212.04356
4. Guillem Pages. faster-whisper: Fast Automatic Speech Recognition via CTranslate2. 2025. URL: https://github.com/SYSTRAN/faster-whisper
5. Rany2. edge-tts: Use Microsoft Edge's online text-to-speech service from Python without an official API. 2025. URL: https://github.com/rany2/edge-tts
6. Chase H. LangChain: фреймворк для розробки застосунків на основі LLM та технології Retrieval-Augmented Generation. 2025. URL: https://python.langchain.com
7. Groq Inc. Groq API Documentation: Llama 3.3 70B Versatile – LPU Inference Engine. 2025. URL: https://console.groq.com/docs/models
8. Chroma. ChromaDB: відкрита AI-нативна векторна база даних. 2025. URL: https://www.trychroma.com
9. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. URL: https://arxiv.org/abs/1706.03762

Додаток Б

Код програми

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from app.routes import dialogue
from dotenv import load_dotenv

load_dotenv()

app = FastAPI(
    title="Museum AI API",
    description="API for interactive dialogue with museum exhibits",
    version="1.0.0")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"])

app.include_router(dialogue.router, prefix="/api/v1")
```

Ініціалізація застосунку FastAPI (app/main.py)

```
@router.post("/chat/audio")
async def chat_audio(exhibit_id: str = Form(...),
                     audio: UploadFile = File(...)):
    file_ext = audio.filename.split('.')[-1]
    input_filename = f"{uuid.uuid4()}.{file_ext}"
    input_filepath = os.path.join(UPLOAD_DIR, input_filename)
    with open(input_filepath, "wb") as buffer:
        shutil.copyfileobj(audio.file, buffer)
    # 1. Speech-to-Text
    user_text = await transcribe_audio(input_filepath)
```

```
    # 2. LLM + RAG
    response_text = await generate_response(exhibit_id, user_text)
    # 3. Text-to-Speech
    output_filename = f"{uuid.uuid4()}.mp3"
    output_filepath = os.path.join(OUTPUT_DIR, output_filename)
    await text_to_speech(response_text, output_filepath)
    return {
        "user_text": user_text,
        "response_text": response_text,
        "audio_url": f"/api/v1/audio/{output_filename}"}
```

Обробник ендпоінту /chat/audio (app/routes/dialogue.py)

```
from faster_whisper import WhisperModel
import asyncio

MODEL_SIZE = "small"
model = WhisperModel(MODEL_SIZE, device="cpu",
                     compute_type="int8")

async def transcribe_audio(file_path: str) -> str:
    def _transcribe():
        segments, info = model.transcribe(
            file_path, beam_size=5, language="uk")
        text = " ".join([s.text for s in segments])
        return text.strip()
    loop = asyncio.get_running_loop()
    return await loop.run_in_executor(None, _transcribe)
```

ASR-сервіс (app/speech/stt.py)

```
import chromadb
from langchain_chroma import Chroma
from langchain_community.embeddings import HuggingFaceEmbeddings

embeddings = HuggingFaceEmbeddings(
    model_name="paraphrase-multilingual-MiniLM-L12-v2",
    model_kwargs={'device': 'cpu'},
```

```
        encode_kwargs={'normalize_embeddings': False})

    def get_vector_store():
        client = chromadb.PersistentClient(path="app/chroma_db")
        return Chroma(client=client,
                      collection_name="museum_exhibits",
                      embedding_function=embeddings)
```

Ініціалізація векторного сховища ChromaDB (app/rag/chroma_db.py)

```
    async def generate_response(exhibit_id, user_question):
        vector_store = get_vector_store()
        llm = get_llm()
        retriever = vector_store.as_retriever(
            search_kwargs={"k": 10,
                           "filter": {"exhibit_id": exhibit_id}})
        prompt = PromptTemplate.from_template(MUSEUM_EXHIBIT_PROMPT)
        rag_chain = (
            {"context": retriever | format_docs,
             "question": RunnablePassthrough()}
            | prompt | llm | StrOutputParser())
        loop = asyncio.get_running_loop()
        return await loop.run_in_executor(
            None, lambda: rag_chain.invoke(user_question))
```

RAG-конвеєр (app/rag/pipeline.py)

```
    import os
    from langchain_groq import ChatGroq

    def get_llm():
        api_key = os.getenv("GROQ_API_KEY")
        if not api_key:
            raise ValueError("GROQ_API_KEY is not set")
        return ChatGroq(
            model="llama-3.3-70b-versatile",
            temperature=0.7,
            max_tokens=512,
```

```
            groq_api_key=api_key)
```

Ініціалізація мовної моделі (app/llm)

```
MUSEUM_EXHIBIT_PROMPT = """Ти – інтерактивний музейний експонат.
Користувач підійшов до тебе в музеї та ставить запитання.
Відповідай ВІД ІМЕНІ експоната українською мовою.

ВАЖЛИВІ ПРАВИЛА:
1. Відповідай ТІЛЬКИ на основі наданого контексту.
   Не вигадуй історичних фактів, дат чи імен.
2. Якщо в контексті немає інформації – чесно скажи,
   що ця інформація втрачена впродовж століть.
3. Відповіді розгорнуті, 4-6 речень.
4. Говори виключно українською мовою.

Контекст про тебе: {context}
Запитання відвідувача: {question}
Твоя відповідь від імені експоната:"""
```

Системна підказка персонажа (app/prompts/system_prompt.py)

```
import edge_tts

VOICE = "uk-UA-OstapNeural"

async def text_to_speech(text: str, output_path: str):
    communicate = edge_tts.Communicate(text, VOICE)
    await communicate.save(output_path)
```

TTS-сервіс (app/tts/edge_tts_service.py)

```
def build_index():
    with open("app/data/exhibits.json", encoding="utf-8") as f:
        exhibits = json.load(f)
    documents = []
    for exhibit in exhibits:
        for fact in exhibit["facts"]:
            documents.append(Document(
                page_content=fact,
```

```
                    metadata={"exhibit_id": exhibit["id"]}))
        vector_store = get_vector_store()
        vector_store.add_documents(documents)
```

Скрипт індексування бази знань (app/database/vector_index.py)

```
  const startRecording = async () => {
    const stream = await navigator.mediaDevices
                          .getUserMedia({ audio: true });
    mediaRecorderRef.current = new MediaRecorder(stream);
    mediaRecorderRef.current.ondataavailable = (e) => {
      if (e.data.size > 0) chunksRef.current.push(e.data);
    };
    mediaRecorderRef.current.onstop = () => {
      const audioBlob = new Blob(chunksRef.current,
                                 { type: 'audio/webm' });
      onRecordingComplete(audioBlob);
    };
    mediaRecorderRef.current.start();
  };
```

Компонент голосового запису (VoiceButton.jsx)

```
  export const chatWithAudio = async (exhibitId, audioBlob) => {
    const formData = new FormData();
    formData.append('exhibit_id', exhibitId);
    formData.append('audio', audioBlob, 'recording.webm');
    const response = await api.post('/chat/audio', formData, {
      headers: { 'Content-Type': 'multipart/form-data' },
    });
    return response.data;
  };
```