ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Система голосового розпізнавання і автоматизації заповнення протоколу обстеження для медичних закладів»

на здобуття освітнього ступеня магістр
за спеціальності 126 Інформаційні системи та технології
(код,найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

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

_____ Андрій ДЕРМАНСЬКИЙ
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти
група ІСДМ-61 Андрій ДЕРМАНСЬКИЙ
(ім'я, ПРІЗВИЩЕ)
Керівник Дмитро КОЗЛОВ
PhD *(ім'я, ПРІЗВИЩЕ)*
Рецензент: Вікторія ЖЕБКА
(ім'я, ПРІЗВИЩЕ)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти магістр

Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____ Каміла
СТОРЧАК

"____" _____2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дерманський Андрій Сергійович
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Система голосового розпізнавання і автоматизації заповнення протоколу обстеження для медичних закладів
керівник кваліфікаційної роботи: Дмитро КОЗЛОВ, PhD
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "___ " жовтня 2025 р. № ____

2.Строк подання кваліфікаційної роботи «26» грудня 2025 р.

3.Вихідні дані кваліфікаційної роботи:
1. Технології автоматичного розпізнавання мовлення (ASR) та OpenAI Whisper.
2. Методи обробки природної мови (NLP) та великі мовні моделі (LLM).
3. Архітектура веб-застосунків на базі Node.js та React.
4. Науково-технічна література, стандарти медичного документообігу та захисту даних (GDPR, HIPAA).

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналітичний огляд методів розпізнавання медичної мови та проблем цифровізації.
2. Проектування архітектури та функціональних компонентів системи голосового заповнення протоколів.

3. Програмна реалізація компонентів системи та аналіз ефективності впровадження.

5.Перелік ілюстраційного матеріалу: *презентація*
6.Дата видачі завдання «___» _____ 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір та аналіз технічної літератури за темою дослідження	01.09.2025 – 15.09.2025	Виконано
2.	Дослідження методів розпізнавання мовлення та аналіз предметної області	16.09.2025 – 30.09.2025	Виконано
3.	Проектування архітектури системи та бази даних	01.10.2025 – 20.10.2025	Виконано
4.	Програмна реалізація, тестування та отримання результатів	21.10.2025 – 25.11.2025	Виконано
5.	Формулювання висновків та оцінка економічної ефективності	26.11.2025 – 05.12.2025	Виконано
6.	Розробка демонстраційних матеріалів (презентація), підготовка доповіді	06.12.2025 – 12.12.2025	Виконано
7.	Оформлення магістерської роботи та проходження нормоконтролю	13.12.2025 – 23.12.2025	Виконано

Здобувач вищої освіти _____ Андрій ДЕРМАНСЬКИЙ
(підпис) *(ім'я, ПРІЗВИЩЕ)*
Керівник кваліфікаційної роботи _____ Дмитро КОЗЛОВ
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступеня магістр: 87 стор., 17 Рис., 5 табл., 36 джерел.

Мета роботи – підвищення ефективності медичного документообігу шляхом розробки програмної системи для автоматизованого голосового заповнення медичних протоколів, що забезпечує скорочення часових витрат лікаря на рутинну роботу.

Об'єкт дослідження – процес створення та обробки медичної документації в закладах охорони здоров'я.

Предмет дослідження – методи та програмні засоби автоматичного розпізнавання мовлення та інтелектуального структурування медичних даних з використанням великих мовних моделей.

Короткий зміст роботи. Проведено аналіз існуючих проблем цифровізації медицини та виявлено потребу в інструментах голосового вводу. Розроблено архітектуру веб-застосунку на базі стеку Node.js та React із використанням мікросервісного підходу. Реалізовано інтеграцію з API OpenAI Whisper для транскрибації та GPT-4o-mini для семантичного аналізу тексту. Забезпечено генерацію документів у форматі .docx на основі динамічних шаблонів. Тестування підтвердило високу точність розпізнавання (WER 4.5%) та стійкість системи до шумів. Впровадження системи дозволяє скоротити час на створення одного протоколу у 2.25 рази. Економічний аналіз показав доцільність використання хмарних API замість локальних обчислень.

КЛЮЧОВІ СЛОВА: EHEALTH, ГОЛОСОВЕ КЕРУВАННЯ, OPENAI WHISPER, GPT-4, ВЕЛИКІ МОВНІ МОДЕЛІ (LLM), МЕДИЧНА ІНФОРМАЦІЙНА СИСТЕМА, SPEECH-TO-TEXT, NODE.JS, REACT.

ABSTRACT

The text part of the qualifying work for obtaining a master's degree: 78 pp., 17 fig., 5 tables, 36 sources.

The aim of the work is to improve the efficiency of medical workflow by developing a software system for automated voice filling of medical protocols, which reduces the time spent by doctors on routine work.

The object of research is the process of creation and processing of medical documentation in healthcare institutions.

The subject of research is methods and software tools for automatic speech recognition and intelligent structuring of medical data using Large Language Models (LLMs).

Summary of the work. An analysis of existing problems in the digitalization of medicine was conducted, and the need for voice input tools was identified. The architecture of a web application based on the Node.js and React stack using a microservice approach was developed. Integration with OpenAI Whisper API for transcription and GPT-4o-mini for semantic text analysis was implemented. Generation of documents in .docx format based on dynamic templates was ensured. Testing confirmed high recognition accuracy (WER 4.5%) and system robustness to noise. The implementation of the system allows reducing the time for creating one protocol by 2.25 times. Economic analysis showed the feasibility of using cloud APIs instead of local computing.

KEYWORDS: EHEALTH, VOICE CONTROL, OPENAI WHISPER, GPT-4, LARGE LANGUAGE MODELS (LLM), MEDICAL INFORMATION SYSTEM, SPEECH-TO-TEXT, NODE.JS, REACT.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасна парадигма охорони здоров’я переживає фундаментальну трансформацію, яку часто помилково зводять лише до впровадження електронних реєстрів та цифрових карток пацієнтів. Безумовно, перехід від паперу до "цифри" відкрив нові горизонти для аналітики та телемедицини, проте він створив і неочікуваний парадокс: замість того, щоб вивільнити час лікаря для безпосередньої взаємодії з пацієнтом, цифровізація перетворила медичного фахівця на висококваліфікованого оператора ПК.

Лікар, чиєю основною зброєю завжди були інтелект, емпатія та клінічне мислення, сьогодні змушений витрачати левову частку свого робочого часу на механічне введення даних у Медичні інформаційні системи (МІС).

Цей процес не лише знижує пропускну здатність клінік, але й провокує явище, відоме як «екранне вигорання», коли візуальний контакт з монітором витісняє контакт з людиною. Клавіатура та миша – це універсальні інтерфейси для програмістів чи офісних працівників, але вони є архаїчними та неефективними посередниками у динамічному середовищі лікарського кабінету, де руки лікаря мають бути зайняті оглядом, а увага – сфокусована на симптоматиці.

Рішення цієї проблеми лежить у площині повернення до найбільш природного інтерфейсу комунікації – людського голосу. Довгий час технології розпізнавання мовлення (ASR) залишалися недостатньо точними для медицини, де помилка в назві дозування чи препарату є неприпустимою. Проте поява сучасних нейромережевих архітектур, зокрема моделей трансформерів (наприклад, OpenAI Whisper), змінила правила гри. Ми опинилися на порозі технологічного зсуву, коли штучний інтелект здатен не просто транскрибувати звук у текст, а й розуміти контекст, фільтрувати шуми та коректно інтерпретувати складну медичну термінологію.

Актуальність даного дослідження зумовлена необхідністю створення

інструментарію, який міг би автоматизувати рутинне заповнення медичних протоколів, виступаючи в ролі «цифрового асистента». Це дозволить нівелювати бар'єр між лікарем та електронною системою охорони здоров'я, перетворивши процес документування з тягаря на непомітний фоновий процес.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконується в рамках загальнодержавного тренду на цифрову трансформацію медичної галузі України та розвитку систем eHealth, спрямованих на оптимізацію робочих процесів медичного персоналу.

Мета і завдання дослідження. Метою роботи є розробка програмної системи для автоматизованого голосового заповнення медичних протоколів, що забезпечує високу точність розпізнавання спеціалізованої лексики та структурування даних для подальшої генерації стандартизованих документів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести аналітичний огляд сучасних методів розпізнавання мовлення та існуючих рішень на ринку Medical Tech, виявивши їхні недоліки в контексті україномовного сегмента та специфіки вітчизняного документообігу.
2. Здійснити порівняльний аналіз архітектур нейронних мереж для задач ASR (Automatic Speech Recognition) та обґрунтувати вибір моделі Whisper як базового двигуна розпізнавання.
3. Розробити алгоритм семантичної обробки транскрибованого тексту (NLP) для виокремлення ключових сутностей (Named Entity Recognition), таких як скарги, діагнози, призначення, та їх розподілу по відповідних полях медичного протоколу.
4. Спроектувати та програмно реалізувати клієнт-серверну архітектуру системи, що включає модуль захоплення аудіо через браузер, модуль обробки даних та модуль генерації документів формату .docx на основі динамічних шаблонів.
5. Виконати тестування розробленого комплексу в умовах, наближених до реальних (наявність фонового шуму, використання складної термінології), та

оцінити ефективність системи за метриками WER (Word Error Rate) та часовими затратами.

Об’єктом дослідження є процес документування результатів медичних обстежень та консультацій в умовах використання електронних систем охорони здоров’я.

Предметом дослідження є методи та програмні засоби голосового введення, розпізнавання та автоматичного структурування неструктурованої медичної інформації для формування звітної документації.

Методи дослідження. У роботі використано комплексний підхід, що базується на методах системного аналізу для формулювання вимог до ПЗ; методах глибокого навчання (Deep Learning) для задачі розпізнавання мовлення; методах обробки природної мови (NLP) для парсингу тексту; а також методах об'єктно-орієнтованого програмування для реалізації кінцевої системи.

Наукова новизна одержаних результатів полягає у розробці інтегрованого підходу до створення медичної документації, який поєднує потужність сучасних трансформерних моделей (Whisper) з гнучкістю шаблонної генерації (DocxTemplater), адаптованого під специфіку медичних протоколів, що дозволяє значно знизити відсоток галюцинацій моделі шляхом жорсткої фіксації структури вихідного документа.

Практичне значення одержаних результатів. Одержані результати мають вагоме значення для подальшої цифровізації сфери охорони здоров'я України. Розроблена програмна система спроектована з урахуванням принципів інтероперабельності, що дозволяє їй бути успішно імплементованою як інтегрований функціональний модуль у вже існуючі Медичні інформаційні системи (МІС), що використовуються в державних лікарнях, так і використовуватись як автономний, кросплатформний інструмент для приватних лікарів-практиків.

Впровадження запропонованого рішення дозволяє кардинально оптимізувати бізнес-процеси медичного закладу, скоротивши непродуктивні

витрати часу на рутинне заповнення первинної облікової документації на 30–50%. Така автоматизація не лише підвищує пропускну здатність лікарського кабінету, але й, що найважливіше, вивільняє критичний часовий ресурс фахівця для безпосередньої взаємодії з пацієнтом, проведення більш ретельної діагностики та формування персоналізованих планів лікування.

Крім того, суттєвий економічний ефект від впровадження системи досягається за рахунок мінімізації адміністративних витрат медичних установ, зокрема, через зменшення потреби у залученні додаткового середнього медичного персоналу (медичних реєстраторів або асистентів) виключно для виконання технічної роботи з набору тексту, перекладаючи цю функцію на алгоритми штучного інтелекту.

1 АНАЛІТИЧНИЙ ОГЛЯД ТА ТЕОРЕТИЧНІ ОСНОВИ

1.1 Аналіз стану цифровізації в медичних установах: від паперової рутини до "екранного вигорання"

Глобальна цифрова трансформація, що охопила світову систему охорони здоров’я в останнє десятиліття, стала незворотним еволюційним процесом. Впровадження електронних медичних карток (EHR – Electronic Health Records) розглядалося як панацея, здатна забезпечити безшовний обмін даними, зменшити кількість лікарських помилок та покращити якість обслуговування пацієнтів. Однак, як показує практика, технологічний прогрес приніс із собою неочікувані побічні ефекти, фундаментально змінивши структуру робочого дня лікаря не на краще.

Феномен, який дослідники дедалі частіше називають «цифровим тейлоризмом», перетворив клініциста на адміністратора даних. Замість фокусування на диференційній діагностиці та комунікації з пацієнтом, лікар змушений взаємодіяти зі складними, часто неінтуїтивними інтерфейсами медичних інформаційних систем (MIC).

Дослідження, проведене групою вчених на чолі з B. G. Arndt, опубліковане в авторитетному виданні Annals of Family Medicine, демонструє тривожну статистику: лікарі первинної ланки витрачають на роботу з EHR більше половини свого робочого часу – 5.9 годин на день з 11.4-годинного робочого дня [1]. Автори влучно характеризують цей стан як «бути прив’язаним до EHR» (tethered to the EHR). Фактично, на кожну годину прямого клінічного контакту з пацієнтом припадає майже дві години роботи за комп’ютером, що включає введення даних, навігацію по меню та формування звітів.

Ця диспропорція підтверджується і в проспективному дослідженні S. Reddy та F. Pompei [2]. Вони кількісно оцінили «часовий податок» на документацію,

довівши, що сучасні інтерфейси введення даних (клавіатура та миша) є «вузьким горлечком» медичного процесу. Швидкість набору тексту, навіть у досвідченого користувача, значно поступається швидкості мовлення, а необхідність структурувати інформацію у численні поля форм (input fields) створює надмірне когнітивне навантаження.

Наслідком такого перевантаження стає професійне вигорання. T. D. Shanafelt та колеги у своєму дослідженні, що охоплює динаміку задоволеності лікарів своєю роботою [3], виявили пряму кореляцію між часом, проведеним за екраном монітора, та рівнем депресивних станів серед медперсоналу. Ситуація ускладнюється тим, що значна частина цієї роботи виконується у неробочий час.

Проблема полягає не лише у кількості часу, але й у якості самих інструментів. M. A. Kaufman у своєму огляді літератури щодо імплементації EHR зазначає, що більшість існуючих систем проектувалися насамперед для адміністративних та білінгових потреб, а не для підтримки клінічних рішень [4]. Це призводить до того, що інтерфейс програм конфліктує з природним ходом думки лікаря, змушуючи його підлаштовувати опис клінічного випадку під жорстку структуру бази даних.

В Україні процес цифровізації медицини має свою специфіку, зумовлену стрімкими темпами впровадження реформ. Відправною точкою для масштабних змін стало Розпорядження Кабінету Міністрів України від 30.11.2016 №1013-р «Про схвалення Концепції реформи фінансування системи охорони здоров’я» [5]. Цей документ заклав фундамент для створення центральної бази даних eHealth, яка стала ядром нової екосистеми.

Відмінністю української моделі є двокомпонентна архітектура: держава адмініструє Центральну базу даних (ЦБД), тоді як інтерфейси для лікарів розробляють приватні компанії – провайдери Медичних інформаційних систем (МІС). Згідно з поточною Стратегією розвитку електронної охорони здоров’я [6], такий підхід мав стимулювати ринкову конкуренцію та покращити UX/UI систем.

Проте, як зазначають у своїй роботі O. Slabodianiuk та A. Fesok, цифрова трансформація в Україні наштовхнулася на низку інфраструктурних та ментальних

бар’єрів [7]. Окрім технічних проблем (застаріле обладнання, нестабільний інтернет у регіонах), українські лікарі стикнулися з тією ж проблемою, що і їхні західні колеги – різким зростанням бюрократичного навантаження, яке тепер перемістилося з паперу в цифру. Необхідність дублювання інформації (часто паперові журнали продовжують вести паралельно з електронними) та складність інтерфейсів МІС створюють значний опір змінам серед медичної спільноти.

Дослідниця O. Kravchenko у статті, присвяченій викликам та перспективам впровадження електронних записів в Україні, підкреслює, що ефективність системи eHealth прямо залежить від швидкості введення первинних даних [8]. Якщо лікар витрачає 15-20 хвилин прийому на боротьбу з інтерфейсом програми, ідея ефективної цифрової медицини нівелюється.

Таким чином, аналіз світового досвіду [1-4] та вітчизняних реалій [5-8] дозволяє зробити однозначний висновок: існуюча парадигма ручного введення даних (manual data entry) досягла межі своєї ефективності. Подальша цифровізація без зміни способів взаємодії «Людина-Комп’ютер» (HCI) призведе лише до поглиблення кризи кадрів. Це актуалізує потребу в пошуку альтернативних методів введення інформації, серед яких найбільш перспективним є використання технологій автоматичного розпізнавання мовлення, що здатні повернути лікаря до пацієнта, залишивши машині роль невидимого стенографіста.

Отже для коректного проектування інформаційної системи необхідно детально проаналізувати існуючі алгоритми роботи медичного персоналу. Згідно з чинними протоколами МОЗ України, процес прийому пацієнта складається з кількох етапів, кожен з яких супроводжується генерацією даних.

Традиційний сценарій (AS-IS):

1. Суб'єктивне обстеження. Лікар опитує пацієнта (скарги, анамнез). На цьому етапі лікар часто робить короткі нотатки у паперовому блокноті, оскільки одночасний набір тексту на комп'ютері порушує зоровий контакт з пацієнтом.
2. Об'єктивне обстеження. Фізикальний огляд (аускультація, пальпація). Руки

лікаря зайняті, введення даних неможливе. Результати запам'ятовуються.

3. Формування висновку. Лікар аналізує дані, встановлює попередній діагноз.
4. Документування (найбільш трудомісткий етап). Після відходу пацієнта (або під час прийому, змушуючи пацієнта чекати) лікар переносить дані у МІС (Медичну інформаційну систему).

Проблеми цього підходу:

Подвійна робота, спочатку запис на папері, потім – у МІС.

Втрата даних, деталі, які лікар не записав одразу, можуть забутися до моменту внесення в комп'ютер (ефект "згасання короткострокової пам'яті").

Часові витрати, середній час набору тексту форми 025/о (Амбулаторна картка) складає 5-7 хвилин при швидкості друку 150-200 знаків/хв.

Окремої уваги заслуговує аналіз часових витрат на формування стандартної звітності. Зокрема, заповнення первинної облікової документації (наприклад, «Амбулаторної картки», форма №025/о) традиційним методом ручного набору займає у лікаря в середньому 5–7 хвилин при швидкості друку 150–200 знаків за хвилину.

Впровадження системи голосового керування дозволяє реалізувати оптимізований сценарій документування (модель «TO-BE»), який докорінно змінює структуру робочого процесу. Замість послідовного виконання дій, коли лікар спочатку оглядає пацієнта, а потім вносить дані, пропонується паралельний підхід. Під час фізикального огляду лікар має можливість диктувати клінічні спостереження вголос, не відволікаючись від пацієнта та не перериваючи візуальний контакт. У цей час програмний комплекс у фоновому режимі виконує транскрибацію аудіопотоку, а модуль штучного інтелекту автоматично структурує та розподіляє розпізнані фрази у відповідні поля медичного протоколу.

Таким чином, роль лікаря зводиться лише до фінальної верифікації автоматично згенерованого документа та його підписання, що фактично усуває трудомісткий етап ручного набору тексту та інтегрує процес документування безпосередньо в процедуру клінічного обстеження.

1.2 Огляд існуючих рішень розпізнавання медичної мови: аналіз ефективності та технологічні бар'єри

Ідея інтеграції голосових помічників у клінічну практику не є новою, проте лише в останні роки, завдяки стрімкому розвитку алгоритмів глибокого навчання, вона перейшла з розряду футуристичних концепцій у площину практичної реалізації.

Як зазначає Ерік Тополь у своїй візіонерській роботі «Deep Medicine» [9], штучний інтелект повинен виконати гуманістичну місію – звільнити лікаря від рутини, повернувши час для емпатії та живого спілкування. Голосовий інтерфейс у цій парадигмі виступає ключовим інструментом «гуманізації» медицини через технології.

Емпіричні дослідження підтверджують цю гіпотезу. Група дослідників під керівництвом N. L. Downing провела масштабний експеримент щодо використання автоматизованих голосових асистентів для документування. Результати, опубліковані у [10], свідчать про те, що використання технологій Speech-to-Text (STT) дозволило скоротити час на заповнення електронної медичної картки в середньому на 30–40% порівняно з традиційним набором тексту на клавіатурі (typing). Це пояснюється фізіологічними відмінностями: середня швидкість набору тексту лікарем становить близько 30–40 слів на хвилину, тоді як швидкість диктування сягає 110–150 слів на хвилину.

Систематичний огляд ефективності розпізнавання мовлення, проведений C. Weng та колегами [11], показує, що сучасні системи досягли рівня точності (Word Error Rate – WER), який робить їх придатними для клінічного використання. Однак автори наголошують, що "сира" транскрипція (raw transcription) недостатня – критично важливим є етап пост-обробки та структурування тексту, що і є одним із завдань даної дисертаційної роботи.

Якщо перейти до аналізу ринкових рішень, то на світовому ринку домінує

кілька ключових гравців, які пропонують пропрієтарні рішення для медичного диктування. Їх можна класифікувати на дві групи: спеціалізовані медичні рішення та хмарні API загального призначення.

Nuance Dragon Medical One – це «золотий стандарт» у галузі. Система використовує спеціалізовані словники та адаптується до голосу конкретного лікаря.

Однак, це рішення має суттєві недоліки для українського контексту: висока вартість ліцензії (тисячі доларів на рік за робоче місце), закрита екосистема (vendor lock-in) та відсутність повноцінної підтримки української медичної термінології з урахуванням відмінювання.

Хмарні платформи (Amazon Transcribe Medical, Google Cloud Healthcare API). Ці сервіси пропонують потужні API для розпізнавання. K. Lui у своєму огляді [12] вказує на високу точність цих моделей завдяки навчанню на величезних масивах даних.

Проте використання цих сервісів в Україні стикається з проблемою суверенітету даних. Передача аудіозаписів пацієнтів на сервери, що фізично розташовані у США або ЄС, вимагає складних юридичних процедур для відповідності законодавству про захист персональних даних (GDPR та локальні закони).

Для наочності порівняння характеристик основних рішень та пропонованої у дипломній роботі системи наведено у Таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика систем розпізнавання медичної мови

Критерій порівняння	Nuance Dragon Medical	Google Cloud / Amazon AWS	Власна розробка (OpenAI Whisper + Templater)
Вартість впровадження	Дуже висока (Enterprise ліцензія)	Середня (Pay-as-you-go)	Низька (Open Source / API costs)
Підтримка української мови	Обмежена / Відсутня	Базова (General Model)	Висока (завдяки мультимовній архітектурі Whisper)
Кастомізація шаблонів	Обмежена вендором	Потребує окремої розробки	Повна (гнучкий DocxTemplater)
Приватність даних	Дані обробляються вендором	Дані на серверах корпорацій	Контрольована (можливість локального деплою або Zero-retention API)
Стійкість до шумів	Висока (вимагає спец. мікрофону)	Середня	Висока (архітектурна особливість Whisper)

Аналіз літературних джерел [9-12] та ринкової кон’юнктури дозволяє стверджувати, що хоча технологія голосового вводу довела свою економічну доцільність, існуючі готові рішення є малодоступними для масового впровадження в українських реаліях. Основні бар'єри – це висока вартість, складнощі з інтеграцією в локальні МІС та питання захисту даних.

Крім того, універсальні моделі часто "губляться" у специфічному суржику медичної латини та української мови, наприклад, «tussis» замість «кашель», або

складні назви препаратів. Це актуалізує необхідність розробки гібридної системи, яка поєднувала б точність сучасних трансформерів (SOTA-моделей) з гнучкістю кастомних алгоритмів обробки тексту, адаптованих під вітчизняні протоколи лікування. Саме такий підхід покладено в основу даної роботи.

Аналіз сучасного ринку медичного програмного забезпечення дозволяє виокремити декілька класів систем, що частково вирішують проблему автоматизації, проте мають суттєві відмінності у функціональності та підходах до взаємодії з користувачем.

До першої категорії належать класичні Медичні інформаційні системи (MIC), такі як Helsi, Doctor Eleks або MedStar. Їхньою беззаперечною перевагою є комплексний підхід до ведення історії хвороби, функціонал запису на прийом та нативна інтеграція з центральною компонентою eHealth. Водночас, основним недоліком цих систем залишається архаїчний спосіб введення даних, що здійснюється переважно через клавіатуру та маніпулятор «миша». Хоча в таких системах існують механізми шаблонів, вони зазвичай є статичними і вимагають ручного заповнення змінних полів, а функція голосового вводу або відсутня, або реалізована примітивно через стандартні засоби операційної системи, що не забезпечує належної якості.

Другу категорію становлять спеціалізовані системи диктування, еталоном серед яких є Nuance Dragon Medical. Ці рішення характеризуються високою точністю розпізнавання складної медичної лексики та можливістю інтеграції в популярні західні EHR (Epic, Cerner). Проте їх масове впровадження в українських реаліях стримується низкою критичних факторів. По-перше, це висока вартість ліцензування, яка може перевищувати 1500 доларів на рік за одне робоче місце. По-друге, відсутність повноцінної підтримки української мови та специфіки локальних протоколів МОЗ. По-третє, архітектурно такі системи часто реалізовані як десктопні додатки («товсті клієнти»), що значно ускладнює процеси їх розгортання та оновлення.

У цьому контексті розроблена система «Medical Voice Assistant» займає

унікальну нішу. Вона позиціонується як легкий веб-сервіс, що не намагається дублювати адміністративний функціонал МІС, а фокусується виключно на швидкому та інтелектуальному створенні первинної документації, нівелюючи недоліки існуючих рішень.

Таблиця 1.2 –Порівняльний аналіз функціональності

Функціонал	Класична МІС (Helsi)	Nuance Dragon	Medical Voice Assistant (Власна розробка)
Метод введення	Клавіатура + Кліки	Голос (диктування)	Голос + Інтелектуальний аналіз
Структурування	Ручне (поля форм)	Послідовне	Автоматичне (AI парсинг)
Підтримка укр. мови	Так	Ні / Обмежена	Так (Whisper Large-v2/v3)
Генерація файлів	PDF	Текст у буфер обміну	DOCX за шаблоном
Мобільність	Веб / Моб. додаток	Десктоп	Кросплатформний Веб
Вартість впровадження	Висока	Висока	Низька (API costs)

Як видно з таблиці, головною перевагою розробленої системи є поєднання голосового вводу з інтелектуальним структуруванням. На відміну від Nuance Dragon, який просто друкує те, що чує (Speech-to-Text), наша система "розуміє" контекст і розкладає його по поличках (Speech-to-Structure), що є критично важливим для заповнення складних табличних форм.

1.3 Теоретичні основи розпізнавання мовлення: від ланцюгів Маркова до трансформерних архітектур

Автоматичне розпізнавання мовлення (ASR – Automatic Speech Recognition) – це міждисциплінарна галузь, що знаходиться на перетині цифрової обробки сигналів, акустики та обчислювальної лінгвістики. З математичної точки зору, завдання ASR формулюється як пошук найбільш ймовірної послідовності слів W=(w1,w2,...,wm) за умови наявності акустичного спостереження O (вектора ознак).

Відповідно до фундаментальної теорії статистичного розпізнавання, цей процес описується правилом Байєса:

$$\hat{W} = arg\ arg\ \max_{W} P(O) = arg\ \max_{W} \frac{P(W) \times P(W)}{P(O)} \quad (3.1)$$

де:

$P(O|\ W)$ – акустична модель (ймовірність того, що слова W звучать як сигнал O);

P(W) – мовна модель (апріорна ймовірність появи послідовності слів W у мові);

P(O) – ймовірність самого акустичного сигналу (ігнорується при максимізації).

Еволюція підходів: Ера прихованих марковських моделей (HMM)

Протягом кількох десятиліть (1980–2010 рр.) домінуючою парадигмою в ASR були Приховані марковські моделі (HMM). Як детально описано в класичній роботі M. Gales та S. Young [13], мовлення моделювалося як стохастичний процес, де стани системи (фонеми) є прихованими, а спостережувані акустичні ознаки генеруються розподілами ймовірностей (зазвичай сумішшю гауссіан – GMM)

Хоча підхід GMM-HMM дозволяв розбивати мовлення на дрібні одиниці, він

мав суттєві обмеження: нездатність моделювати довготривалі залежності (контекст) та низька стійкість до шумів, що є критичним для медичних умов.

Переломним моментом для Deep Learning стала робота G. Hinton та співавторів [14], яка продемонструвала перевагу глибоких нейронних мереж (DNN) над гауссовими сумішами при акустичному моделюванні. Заміна GMM на DNN дозволила моделям вивчати більш складні нелінійні залежності у спектрограмах звуку.

Наступним еволюційним кроком стала відмова від жорсткої прив'язки до кадрів (frames) завдяки рекурентним мережам (RNN) та функції втрат CTC (Connectionist Temporal Classification), запропонованій A. Graves [15]. Цей метод дозволив навчати мережі в режимі End-to-End, де вхідна аудіопослідовність безпосередньо відображається у вихідний текст, минаючи етап складного фонемного вирівнювання. Це значно спростило архітектуру систем, однак рекурентна природа RNN (послідовна обробка) обмежувала можливості паралелізації обчислень, що сповільнювало навчання на великих датасетах.

Сучасний етап розвитку NLP та ASR розпочався з публікації статті A. Vaswani et al. "Attention Is All You Need" [16], яка представила архітектуру Transformer. Ключовою інновацією став механізм Self-Attention (самоуваги), який дозволяє моделі враховувати взаємний вплив усіх елементів послідовності одночасно, незалежно від відстані між ними.

Матриця уваги обчислюється за формулою:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.2}$$

де Q (Query), K (Key) та V (Value) – матриці, отримані з вхідних даних. Це дозволило моделям "бачити" глобальний контекст (наприклад, розуміти, що слово на початку речення впливає на закінчення останнього слова), що раніше було недосяжно для RNN.

На сьогоднішній день State-of-the-Art (SOTA) рішення в ASR базуються на варіаціях трансформерів. A. Baevski представив модель Wav2vec 2.0 [17], яка використовує самоконтрольоване навчання (self-supervised learning) на сирих аудіоданих, що дозволяє досягати високої точності навіть при малій кількості розмічених даних.

Однак для даної дипломної роботи було обрано архітектуру Whisper, представлену A. Radford та командою OpenAI [18]. Whisper – це модель типу Encoder-Decoder Transformer, навчена на 680 000 годин мультимовних та багатозадачних даних за методом Weak Supervision (слабкого нагляду).

На відміну від традиційних моделей, які навчаються на ідеально розмічених "золотих" датасетах (наприклад, LibriSpeech), Whisper навчався на "шумних" даних з інтернету. Це надало моделі унікальної властивості – робастності (стійкості) до акцентів, фонового шуму та технічних артефактів запису.

Архітектура Whisper працює наступним чином:

1. Вхідний аудіосигнал перетворюється на Log-Mel спектрограму.
2. Енкодер (складається з блоків трансформерів) обробляє спектрограму та формує приховані представлення.
3. Декодер авторегресивно генерує текстові токени, використовуючи механізм перехресної уваги (Cross-Attention) до виходів енкодера.

Дослідження A. S. Koc [19], проведене у 2023 році, порівнювало Whisper з Google Speech-to-Text API саме на медичних даних. Результати показали, що завдяки величезному обсягу навчальних даних Whisper краще справляється з контекстуальною неоднозначністю медичної термінології, ніж класичні моделі, що вимагають спеціалізованого донавчання (finetuning).

Отримання транскрибованого тексту за допомогою систем автоматичного розпізнавання мовлення (ASR), розглянутих у попередньому підрозділі, є необхідною, але недостатньою умовою для автоматизації медичного документообігу. Результатом роботи моделі Whisper є так званий «сирий» текст (raw text) – неструктурований потік символів, який містить інформацію у вигляді

природної мови. Однак для заповнення полів медичної картки або формування протоколу (наприклад, форми 028/о) системі необхідно не просто «чути» слова, а «розуміти» їх семантичне значення. Цю задачу вирішує дисципліна, відома як обробка природної мови (Natural Language Processing – NLP).

Як зазначають у своїй фундаментальній праці Д. Журафскі та Дж. Мартін [20], процес перетворення неструктурованого тексту на структуровані дані є багаторівневою задачею, що включає морфологічний аналіз, синтаксичний парсинг та, що найважливіше для нашого дослідження, – семантичну інтерпретацію. У контексті медичної інформатики ключовою підзадачею NLP стає вилучення інформації (Information Extraction – IE), яке дозволяє автоматично ідентифікувати у суцільному масиві тексту специфічні факти: діагнози, дозування препаратів, прізвища пацієнтів та дати прийомів.

Історично склалося два діаметрально протилежних підходи до вирішення задачі структурування тексту, кожен з яких має свої переваги та обмеження в клінічних умовах.

1. Метод регулярних виразів та правилова логіка (Rule-based approach)

На ранніх етапах цифровізації, та й у багатьох сучасних системах, основним інструментом залишаються регулярні вирази (Regular Expressions – RegEx). Цей підхід базується на створенні жорстких шаблонів пошуку. Наприклад, артеріальний тиск майже завжди записується як два числа через скісну риску (наприклад, "120/80"), а дата народження має фіксований формат.

Перевагою цього методу є його абсолютна прозорість та детермінованість: розробник точно знає, чому система вилучила ту чи іншу інформацію. Однак, як справедливо зауважує Й. Голдберг у роботі, присвяченій нейромережевим методам в NLP [21], правилова логіка виявляється безсилою перед варіативністю природної мови. Варто лікарю сказати не "тиск 120 на 80", а "гіпертензія, систолічний сто двадцять", як жорсткий шаблон перестає спрацьовувати. У спробах покрити всі можливі варіації висловлювань розробники створюють громіздкі системи правил, які стає неможливо підтримувати.

2. Розпізнавання іменованих сутностей (Named Entity Recognition – NER)

Більш сучасним та гнучким підходом є використання машинного навчання для задачі NER. Згідно з оглядом Д. Надо та С. Секіна [22], задача NER полягає у класифікації кожного слова (токена) у тексті до певної, заздалегідь визначеної категорії (наприклад, PERSON, DISEASE, DRUG, DATE).

На відміну від RegEx, моделі NER (особливо ті, що базуються на архітектурі BERT або BiLSTM-CRF) враховують контекст. Вони здатні розрізнити, що у реченні "Призначено Напроксен" слово є ліками, а у фразі "Пацієнт Напроксенко" – частиною прізвища, хоча морфологічно слова дуже схожі. Для медицини це критично важливо, оскільки ціна помилки класифікації тут надзвичайно висока.

Однак специфіка клінічного NLP у медицині наштовхується на ряд унікальних викликів, які детально проаналізовані в систематичному огляді Й. Ванга та колег [23]. Медична мова характеризується високою щільністю абревіатур, професійного сленгу та змішуванням мов (української, латини, англійських назв препаратів). Більше того, клінічні записи часто є телеграфними, з пропущеними підметами та присудками, що ускладнює роботу стандартних парсерів, навчених на літературних текстах.

К. Робертс у своєму дослідженні стану клінічного NLP [24] підкреслює проблему неоднозначності (ambiguity) та заперечень. Наприклад, фраза "немає ознак пневмонії" містить слово "пневмонія", але система повинна зрозуміти контекст заперечення і не вносити цей діагноз у список активних захворювань. Більшість простих систем пошуку за ключовими словами ігнорують цей аспект, що призводить до хибно-позитивних результатів.

Враховуючи обмеження кожного з методів окремо, у даній дипломній роботі пропонується використання гібридного підходу до парсингу результатів розпізнавання мовлення. З одного боку, використання великих мовних моделей (LLM), подібних до GPT (на етапі пост-обробки тексту від Whisper), дозволяє реалізувати потужний механізм NER, здатний розуміти складний контекст і виправляти помилки узгодження слів [21]. З іншого боку, для суворо

стандартизованих даних, таких як дати, номери телефонів або коди МКХ-10, доцільно залишити перевірені алгоритми на базі регулярних виразів, що забезпечить 100% точність форматування.

Така комбінація дозволяє досягти синергетичного ефекту: модель Whisper забезпечує акустичне розпізнавання, LLM-компонент відповідає за семантичне розуміння та вилучення сутностей, а детерміновані алгоритми гарантують відповідність вихідних даних жорстким вимогам медичних шаблонів.

Застосування технологій обробки природної мови (NLP) для українського контенту пов’язане з низкою специфічних проблем, що суттєво відрізняють його від англомовного, на якому тренується більшість базових моделей. Глибоке розуміння цих викликів є необхідним для обґрунтування вибору архітектури системи на базі великих мовних моделей (LLM).

Перш за все, українська мова належить до синтетичних флективних мов, що означає зміну слів за відмінками, родами та числами (наприклад: «запалення», «запаленням», «при запаленні»). Для класичних алгоритмів пошуку за ключовими словами (Keyword Search) це створює проблему, оскільки словоформа у словнику шаблонів часто не збігається зі словом у розпізнаному тексті. Використання сучасних LLM дозволяє ефективно вирішити цю проблему завдяки токенізації на рівні морфем (Subword Tokenization) та розумінню семантичного зв’язку між словами, незалежно від їхньої форми.

Другим суттєвим викликом є поширеність явища білінгвізму та інтерференції («суржику»). У реальній медичній практиці лікарі часто використовують русизми або кальки, особливо під час швидкого диктування (наприклад, «жалоби» замість «скарги», «болі в області серця» замість «біль у ділянці серця»). Система автоматичного розпізнавання повинна бути стійкою до таких відхилень і забезпечувати нормалізацію тексту до літературної медичної норми під час генерації фінального документа .

Окрему складність становить обробка спеціалізованої термінології, де латинські назви діагнозів та препаратів можуть вживатися як латиницею («Angina

Pectoris»), так і в кириличній транслітерації. Ефективна NLP-система повинна виконувати задачу зв’язування сутностей (Entity Linking), щоб зіставити різні варіанти написання з єдиним кодом у класифікаторах МКХ-10 або АТС. У розробленій системі ця задача покладається на контекстне розуміння моделі GPT-4o .

Нарешті, на відміну від англійської мови з фіксованим порядком слів, в українській мові він є вільним і залежить від логічного наголосу. Це ускладнює використання простих регулярних виразів для вилучення інформації, оскільки цільове значення (наприклад, температура) може знаходитися як перед, так і після ключового слова. Саме тому використання трансформерних архітектур (Attention Mechanism), які аналізують все речення цілком, є єдиним ефективним рішенням для даної предметної області.

1.4 Стандартизація обміну медичними даними: HL7 FHIR та CDA

Цифровізація медицини неможлива без уніфікованих стандартів обміну даними. Хоча кінцевим результатом роботи розроблюваної системи є документ для сприйняття людиною (формат .docx), для розуміння контексту інтеграції в екосистему eHealth необхідно проаналізувати існуючі протоколи інтероперабельності.

Стандарт HL7 CDA (Clinical Document Architecture) Історично першим глобальним стандартом для електронних медичних документів став HL7 CDA. Він базується на мові розмітки XML і визначає структуру клінічних документів, таких як виписні епікризи та протоколи огляду. Структура CDA складається з двох частин:

1. Header (Заголовок): Метадані про пацієнта, лікаря, заклад.
2. Body (Тіло): Власне клінічний контент, який може бути структурованим (XML-теги) або неструктурованим (вкладений PDF або текст).

Недоліком CDA є його надмірна складність та жорсткість структури, що ускладнює розробку сучасних легких веб-застосунків.

Стандарт HL7 FHIR (Fast Healthcare Interoperability Resources) Сучасним стандартом, прийнятим МОЗ України як базовий для розвитку системи eHealth [6], є FHIR (вимовляється як "Fire"). На відміну від CDA, він базується на сучасних веб-технологіях (RESTful API, JSON, HTTP), що робить його ідеальним для інтеграції з веб-сервісами.

Ключовою одиницею FHIR є "Ресурс" (Resource) – найменша неподільна одиниця інформації. Для задачі автоматизованого документування релевантними є ресурси:

1. Patient – демографічні дані пацієнта.
2. Practitioner – дані про лікаря.
3. DiagnosticReport – результати обстежень (наприклад, протокол УЗД).
4. Observation – окремі вимірювання (наприклад, "розмір лівого яєчника: 30 мм").

Проектована система голосового вводу виконує функцію "моста" (bridge): вона перетворює неструктурований голос лікаря спочатку в проміжний JSON (структура якого семантично близька до композиції ресурсів FHIR), а потім – у людиночитаний документ. Це забезпечує потенційну технічну можливість майбутньої інтеграції з центральним компонентом eHealth через стандартні API без зміни архітектури даних.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ФУНКЦІОНАЛЬНИХ КОМПОНЕНТІВ СИСТЕМИ

2.1 Аналіз вимог та формалізація бізнес-логіки

Проектування системи "Medical Voice Assistant" базується на аналізі потреб сучасних медичних закладів у автоматизації рутинних процесів документування. Ключовою вимогою є створення інструменту, що інтегрує голосове введення даних у існуючі бізнес-процеси клініки без необхідності складної адаптації персоналу.

У розробленій системі реалізовано механізм розмежування прав доступу на основі рольової моделі (RBAC – Role-Based Access Control), що передбачає виділення трьох основних категорій користувачів. Ключовою фігурою в системі є «Лікар» (DOCTOR), який володіє повним набором інструментів для ведення медичної документації. До його повноважень належить створення нових записів шляхом голосового диктування, перегляд та редагування власної історії документів, а також експорт згенерованих файлів у форматі .docx та доступ до персональної статистики ефективності.

Допоміжна роль «Медсестра» (NURSE) наділена обмеженим функціоналом, який загалом повторює можливості лікаря, проте може бути адаптований для ведення специфічних типів документації, таких як процедурні листки, без доступу до лікарської аналітики.

Найвищий рівень доступу надається ролі «Адміністратор» (ADMIN), яку зазвичай виконує технічний спеціаліст або головний лікар закладу. Окрім стандартних функцій лікаря, адміністратор отримує ексклюзивні права на керування конфігурацією системи, що включає завантаження та налаштування нових шаблонів документів (.docx), управління системними довідниками, а також моніторинг зведеної аналітики щодо ефективності роботи всього медичного персоналу.

Важливим аспектом проектування є визначення станів, через які проходить медичний запис. Згідно з вимогами бізнес-логіки, сутність Document може мати наступні статуси:

1. DRAFT (Чернетка): Початковий статус. Аудіо записано, транскрибація виконана, але дані ще не перевірені лікарем або процес генерації файлу не завершено.
2. COMPLETED (Завершено): Документ успішно згенерований, збережений у файловій системі та готовий до завантаження.
3. EXPORTED (Експортовано): Статус, що фіксує факт завантаження файлу на локальний пристрій користувача (важливо для аудиту).
4. ARCHIVED (Архівовано): Документ перенесено в архів (м'яке видалення), він доступний для перегляду, але прихований з основного списку.

Оскільки система реалізована за триланковою архітектурою (Three-Tier Architecture), що включає клієнтську частину, сервер застосунків та сервер баз даних. Взаємодія між компонентами здійснюється через REST API, спроектований відповідно до дисертації Р. Філдінга [25].

В якості основної платформи бекенду обрано Node.js [26]. Вибір зумовлений необхідністю ефективної обробки асинхронних запитів (I/O bound tasks), таких як завантаження файлів та очікування відповідей від зовнішніх AI-сервісів. Веб-сервер реалізовано на базі фреймворку Express, який забезпечує маршрутизацію та використання Middleware для обробки помилок, логування та валидації JWT-токенів.

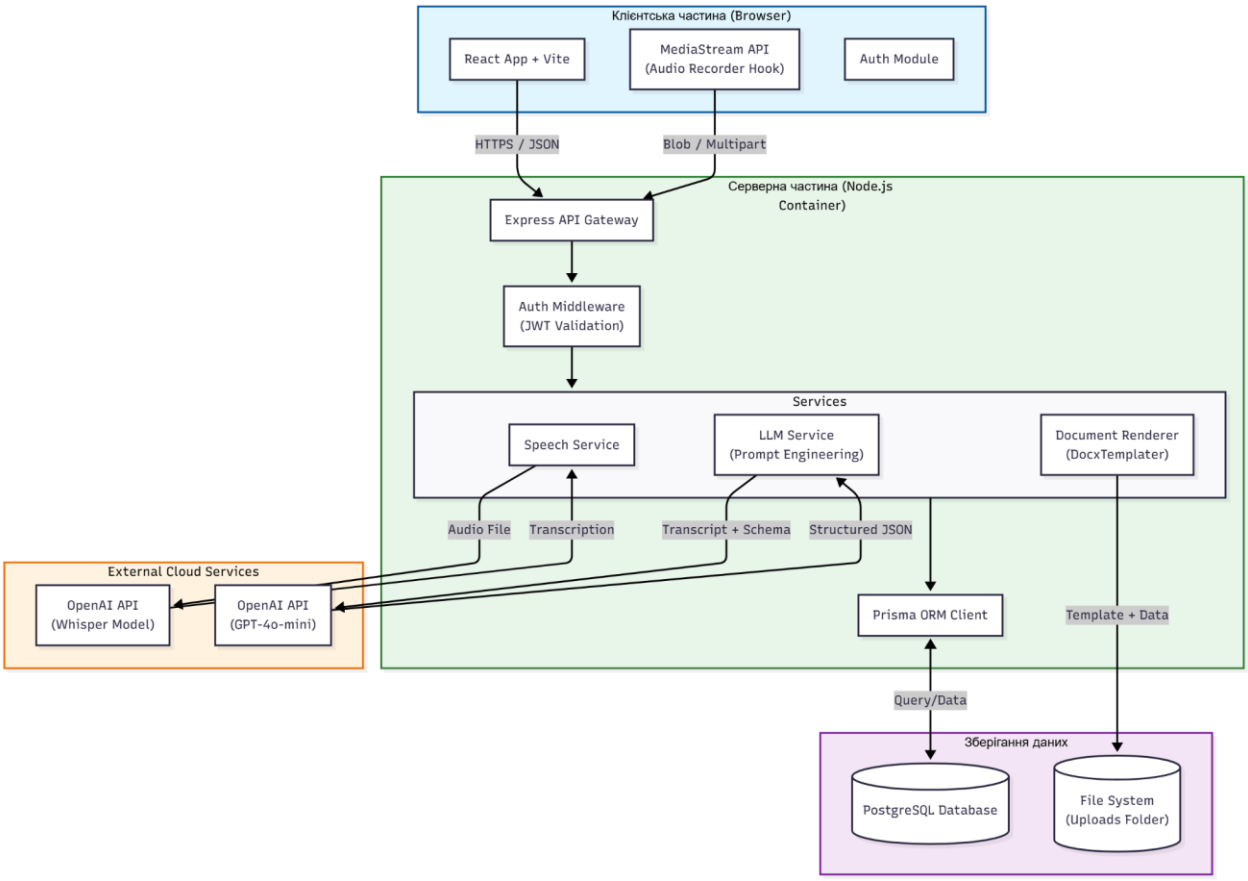


Рисунок 2.1 Архітектура програмного комплексу системи

Внутрішня архітектура бекенду побудована за модульним принципом, де кожен сервіс відповідає за окремий домен задачі:

1. Speech Service. Відповідає за взаємодію з API OpenAI Whisper для перетворення аудіопотоку (формати webm, mp3, wav) у текстовий формат.
2. Template Parser Service. Аналізує завантажені .docx файли, виявляючи змінні поля (наприклад, {{patient_name}}) та логічні блоки. Це дозволяє динамічно формувати схему даних для AI.
3. LLM Service. Формує контекстний запит (Prompt) до моделі GPT-4o-mini [27]. На вхід подається транскрипція та JSON-схема шаблону, на виході отримується структурований об'єкт даних.
4. Document Renderer Service. Використовує бібліотеку docxtemplater [28] для підстановки отриманих даних у шаблон, зберігаючи форматування оригінального документа.

Для забезпечення переносимості та ізоляції компонентів використовується технологія контейнеризації Docker. Проект описується файлом docker-compose.yml, що дозволяє розгорнути повний стек (Frontend, Backend, Database) однією командою.

2.2 Проектування бази даних (Database Design)

Підсистема збереження даних реалізована на базі реляційної системи управління базами даних PostgreSQL. Для забезпечення ефективної та безпечної взаємодії серверного додатку зі сховищем даних використано технологію об'єктно-реляційного відображення (ORM) Prisma. Такий підхід гарантує сувору типізацію даних на етапі компіляції коду та забезпечує автоматичний захист від розповсюджених вразливостей, зокрема SQL-ін'єкцій.

Спроектована схема бази даних (schema.prisma) базується на чотирьох ключових сутностях. Модель User (Користувачі) відповідає за зберігання облікових записів медичного персоналу. Розмежування прав доступу реалізовано через поле role з використанням перелічуваного типу (UserRole), що підтримує ролі лікаря, медсестри та адміністратора. З метою безпеки паролі зберігаються виключно у вигляді криптографічних хешів. Кожен користувач має зв'язок типу «один-до-багатьох» (1:N) із сутністю Document, що дозволяє формувати персоналізовану історію записів.

Для управління бланками документів розроблено модель Template (Шаблони). Її важливою архітектурною особливістю є поле content типу JSON, яке зберігає попередньо розпарсену структуру полів шаблону. Це дозволяє оптимізувати продуктивність системи, уникаючи необхідності повторного синтаксичного аналізу файлу .docx при кожній генерації документа. Також модель підтримує механізм версійності для контролю змін у медичних формах.

Центральним елементом бізнес-логіки виступає модель Document, яка фіксує

стан обробки запиту (status) та шляхи до згенерованих файлів. Важливою деталлю є збереження фінальних даних у полі filledData, що надає можливість повторно генерувати документ без додаткових звернень до сервісів штучного інтелекту. Ця модель має зв'язок «один-до-одного» з моделлю AudioRecording, яка зберігає метадані аудіофайлу, його тривалість та кешований текст транскрибації. Для підтримки цілісності даних на рівні БД налаштовано каскадне видалення (onDelete: Cascade), що забезпечує автоматичне знищення пов'язаних аудіозаписів та транскрипцій при видаленні основного документа.

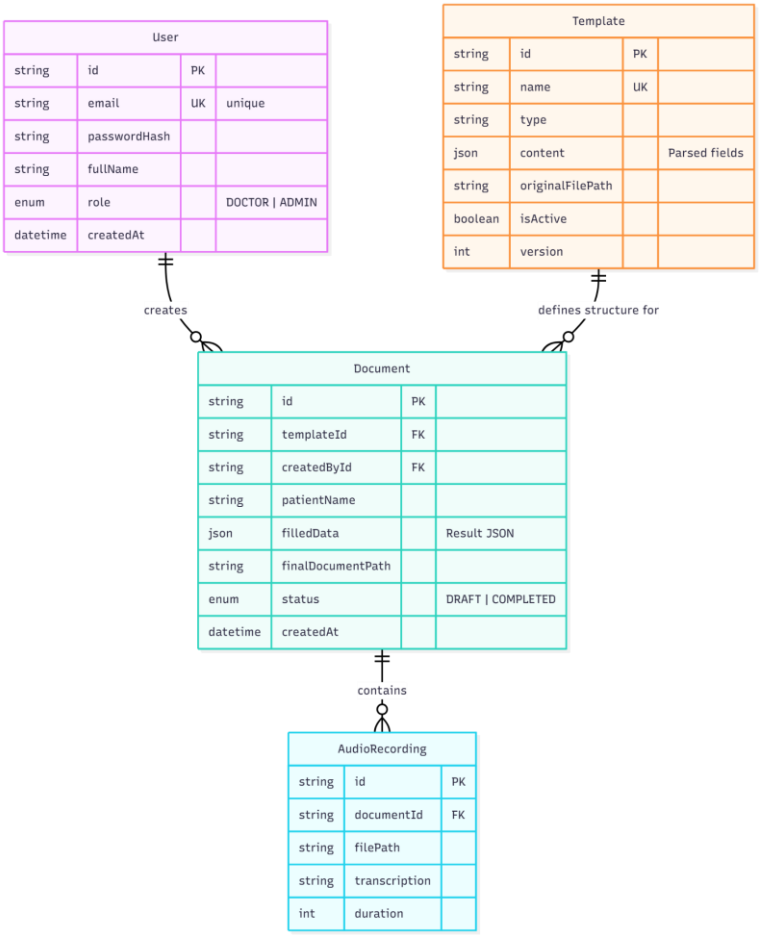


Рисунок 2.2 ER-діаграма бази даних системи

Архітектура взаємодії між клієнтською та серверною частинами системи побудована на принципах RESTful API, що забезпечує стандартизацію методів доступу до ресурсів. Усі кінцеві точки (endpoints) логічно згруповані за семантичними доменами, що спрощує підтримку та масштабування програмного

коду.

Група маршрутів автентифікації (/auth) відповідає за керування життєвим циклом сесії користувача. Зокрема, реалізовано методи для реєстрації нових облікових записів та авторизації, яка завершується видачею пари токенів (Access та Refresh JWT) для подальшого доступу до захищених ресурсів. Також передбачено окремий метод для отримання профілю поточного користувача.

Основним функціональним ядром системи є група бізнес-процесів (/process). Ключовий маршрут цієї групи приймає аудіофайл (формат multipart/form-data) разом із ідентифікатором обраного шаблону та ініціює повний цикл обробки: від транскрибації та семантичного структурування до генерації фінального документа. Результатом виконання запиту є об’єкт створеного документа та метадані про час обробки.

Для керування записами виділено групу ресурсів /documents. Вона підтримує повний набір операцій: отримання списку документів із пагінацією та фільтрацією, примусову регенерацію файлу на основі збережених даних (у разі зміни шаблону), а також видалення записів, що супроводжується фізичним очищенням файлів з дискового простору та бази даних.

Окремий рівень доступу реалізовано для групи шаблонів (/templates), яка доступна виключно користувачам з роллю адміністратора. При завантаженні нового шаблону сервер виконує його автоматичну валідацію, перевіряючи коректність структури .docx файлу перед збереженням у системі.

2.3 Інтелектуальна обробка даних та Prompt Engineering

Критичним компонентом системи є модуль інтеграції зі штучним інтелектом [29]. Оскільки пряма передача «сирого» неструктурованого тексту транскрибації у шаблон документа є неможливою, використано підхід ін'єкції JSON-схеми (JSON Schema Injection).

Алгоритм обробки даних реалізовано як послідовний процес. На першому етапі система аналізує файл обраного шаблону та формує перелік очікуваних полів (наприклад, diagnosis, treatment, complaints). На основі цих метаданих динамічно конструюється системний промпт для моделі GPT-4o-mini. Структура промпту включає рольову інструкцію («Ти – медичний асистент...»), власне транскрибований текст та жорстку JSON-схему, яка визначає формат вихідних даних.

Отримана відповідь від нейромережі проходить автоматичну валідацію на відповідність типам даних. У випадку виникнення помилок обробки, таких як збій API провайдера або перевищення ліміту токенів, система повертає клієнту відповідний HTTP-код помилки (502 Bad Gateway або 422 Unprocessable Entity). Такий підхід дозволяє реалізувати логіку повторних спроб на стороні клієнта, забезпечуючи надійність системи без необхідності впровадження складних механізмів автоматичних повторів (retries) на стороні сервера.

Першим етапом обробки є деконструкція файлу .docx, завантаженого адміністратором. Оскільки формат DOCX є архівом XML-файлів, для коректного зчитування "сирого" тексту без втрати логічної структури було використано бібліотеку mammoth.

Однак, простого тексту недостатньо. Системі необхідно побудувати Абстрактне синтаксичне дерево (AST) шаблону, щоб зрозуміти, де знаходяться змінні, а де – умовні блоки. Для цього було розроблено власний алгоритм на основі скінченних автоматів та стекової пам'яті.

Крок 1. Токенізація тексту Ключовим елементом алгоритму є регулярний вираз, який розбиває суцільний текст на значущі токени (теги) та звичайний контент.

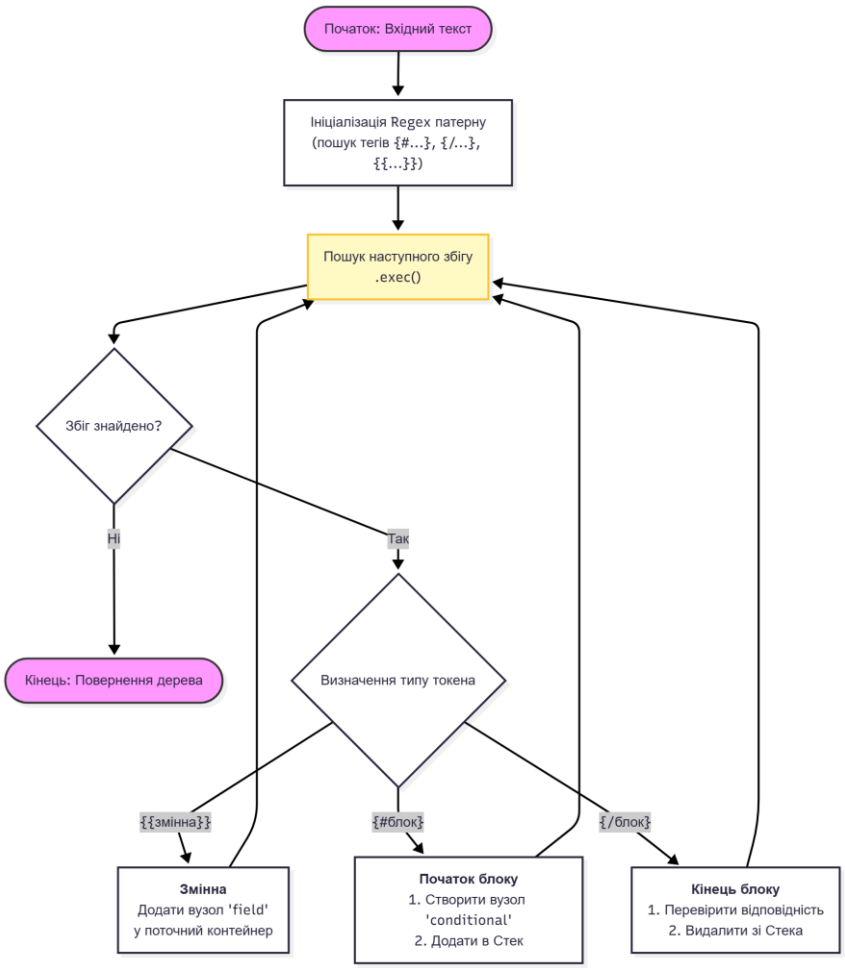


Рисунок 2.3 Програмна реалізація алгоритму розбору структури шаблону

Крок 2. Стекова машина для вкладених блоків Медичні шаблони часто мають складну структуру, де один умовний блок може знаходитись всередині іншого (наприклад, блок "Патологія" всередині блоку "Матка"). Для коректної обробки такої ієрархії використано структуру даних Стек (Stack).

Логіка роботи:

1. При зустрічі відкриваючого тега {#block}, створюється новий вузол, який додається в масив children поточного батьківського елемента, а також поміщається на вершину стека.
2. Усі наступні текстові вузли додаються до елемента, що знаходиться на вершині стека.
3. При зустрічі закриваючого тега {/block}, алгоритм перевіряє відповідність імен тегів і видаляє елемент зі стека, повертаючись на рівень вище.

Такий підхід забезпечує валідацію цілісності шаблону: якщо користувач забув закрити блок у Word, система викине виключення Unclosed conditional block ще на етапі завантаження шаблону.

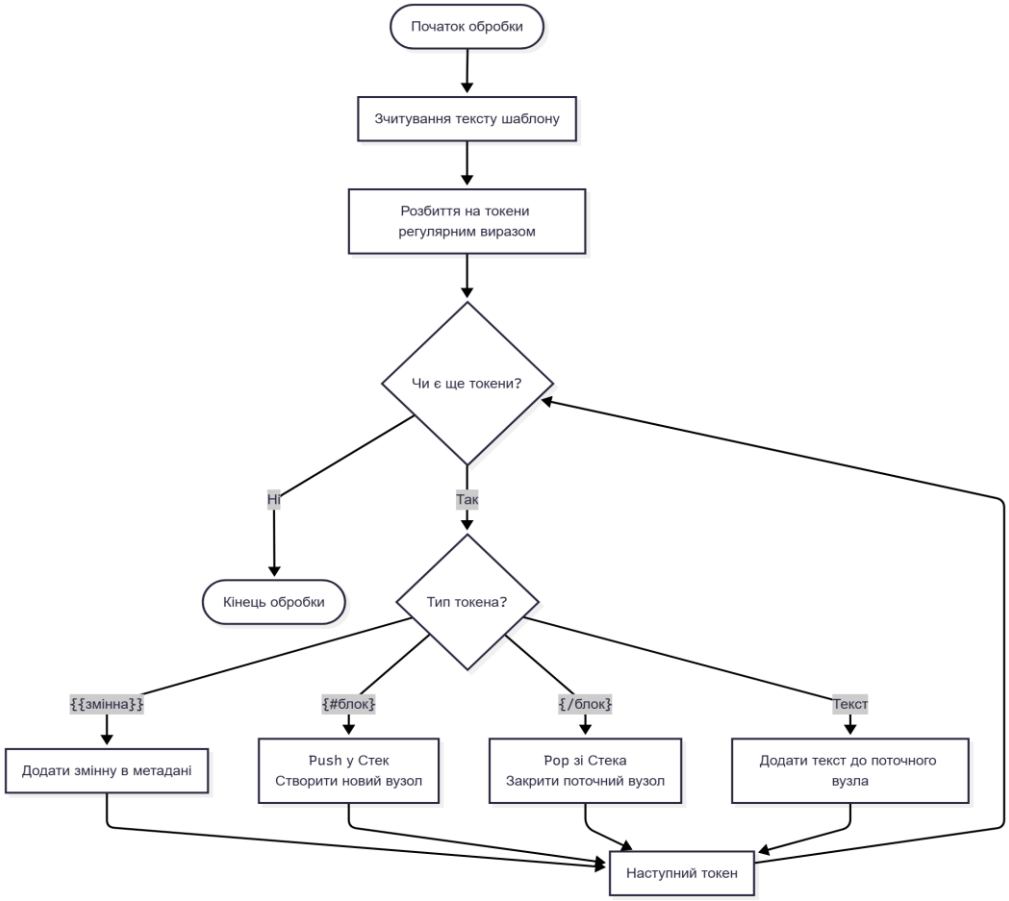


Рисунок 2.4 Блок-схема алгоритму синтаксичного аналізу шаблонів

На основі розпарсених метаданих сервіс LLMService генерує унікальну інструкцію для нейромережі. Це приклад застосування техніки Dynamic Prompting, коли поведінка ШІ програмується даними, а не кодом.

Метод buildSystemPrompt конструює текстову інструкцію, яка складається з декількох секцій, що чітко визначають межі відповідальності моделі.

Секція 1 – рольова установка (Persona) задає контекст виконання завдання:

"Ти – асистент, який структуровано витягує дані з медичних диктувань. Проаналізуй надану транскрипцію та підготуй заповнений JSON."

Секція 2 – формат виводу (Output Constraint), для забезпечення стабільної роботи генератора документів критично важливо, щоб модель повертала дані у

суворо визначеному форматі.

"Завжди повертай валідний JSON-об’єкт із ключами 'insert', 'select', 'select_block' та 'write'. Не додавай жодних пояснень."

Секція 3 – система програмно ітерує по всіх знайдених полях шаблону та додає інструкції для кожного з них. Це дозволяє моделі "розуміти", що саме від неї вимагається в конкретному полі.

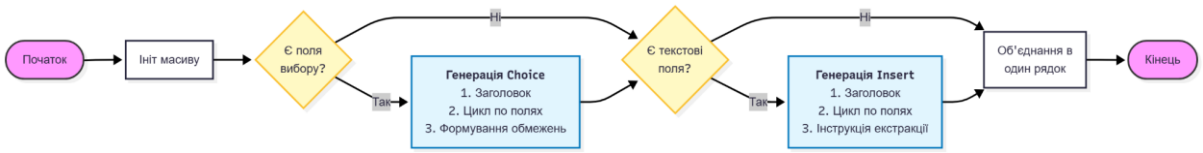


Рисунок 2.5 Блок-схема алгоритму динамічного формування інструкцій для полів

Цей алгоритм перетворює технічну структуру шаблону на зрозумілу для LLM мову, фактично виконуючи роль "перекладача" між жорсткою логікою програми та нечіткою логікою нейромережі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

На основі спроектованої архітектури (Розділ 2) було виконано програмну реалізацію інформаційної системи "Medical Voice Assistant". Розробка велася з дотриманням принципів чистої архітектури (Clean Architecture), що забезпечує модульність, тестопридатність та легкість у супроводі коду. В якості основного інструментарію обрано екосистему JavaScript/TypeScript на базі платформи Node.js [26].

3.1 Програмна реалізація інфраструктури та архітектури рішення

Для забезпечення цілісності розробки та зручності розгортання було обрано стратегію монорепозиторію (Monorepo). Це означає, що код серверної частини (backend) та клієнтського інтерфейсу (frontend) зберігається в одній системі контролю версій, але розділений на ізольовані директорії з власними файлами конфігурації package.json.

Обидві частини системи реалізовані мовою TypeScript. Використання статичної типізації є критично важливим для медичних систем, оскільки це дозволяє виявити більшість помилок на етапі компіляції, а не під час виконання (Runtime).

У корені проекту налаштовано файл tsconfig.json з жорсткими правилами типізації (strict: true). Це змушує розробника явно описувати інтерфейси для всіх об'єктів даних, що передаються між модулями.

Основні параметри конфігурації:

1. target: "ES2020" – генерація сучасного JavaScript-коду, що підтримується Node.js v18+.
2. moduleResolution: "node" – використання стандартного алгоритму пошуку модулів.

3. esModuleInterop: true – забезпечення сумісності з бібліотеками CommonJS (наприклад, старими версіями драйверів БД).

Серверна частина побудована на базі фреймворку Express. Архітектура проекту реалізує патерн "Controller-Service-Repository", який дозволяє чітко розподілити зони відповідальності компонентів.

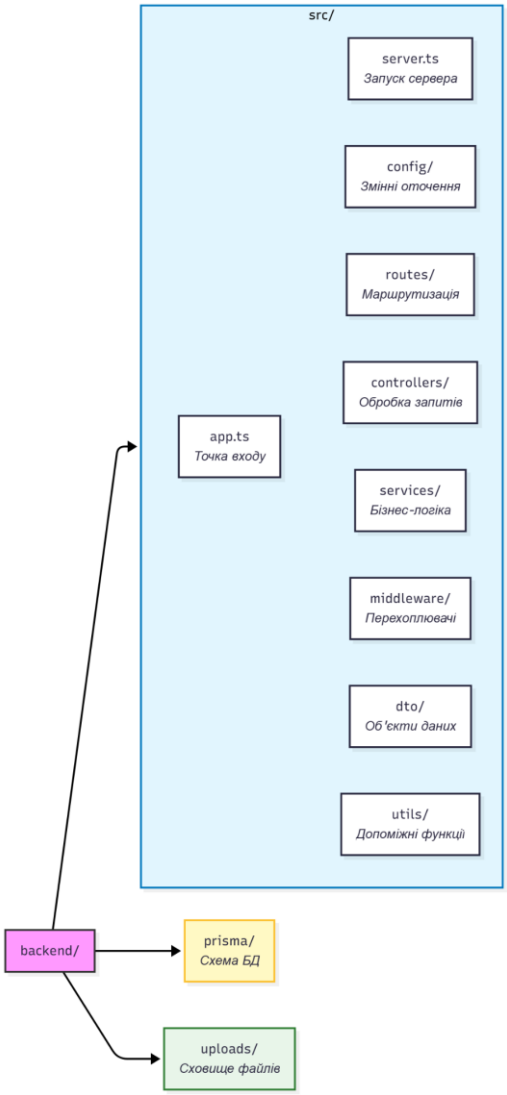


Рисунок 3.1 Структура файлової системи серверної частини проекту

Детальний опис компонентів:

1. Routes (Маршрути). Файли у цій папці (наприклад, document.routes.ts) не містять логіки. Їхня задача – зв'язати URL-адресу (наприклад, POST /api/documents) з відповідним методом контролера.
2. Controllers (Контролери). Цей шар відповідає за валідацію вхідних даних. Він

перевіряє, чи надіслав клієнт усі необхідні поля, витягує файли з потоку запиту та передає управління сервісам. Контролер ніколи не звертається до бази даних напряму.

3. Services (Сервіси). Саме тут реалізована "інтелектуальна" частина системи. Наприклад, SpeechService відповідає за логіку конвертації аудіо, а LLMService – за формування промптів. Такий підхід дозволяє легко тестувати бізнес-логіку (Unit Testing), підміняючи реальні залежності на моки (Mocks).

Для реалізації функціоналу серверної частини було використано низку спеціалізованих бібліотек, перелік яких зафіксовано у файлі конфігурації package.json. Ключовим елементом забезпечення цілісності даних виступає бібліотека zod, яка відповідає за валідацію схем на етапі виконання (runtime validation). Її використання гарантує, що будь-який JSON-об'єкт, отриманий від клієнта, чітко відповідає очікуваній структурі перед початком обробки.

Питання мережевої безпеки вирішено шляхом інтеграції пакетів cors та helmet, які автоматизують налаштування HTTP-заголовків захисту та політики спільного використання ресурсів (Cross-Origin Resource Sharing). Враховуючи вимоги до аудиту в медичних інформаційних системах, для логування вхідних HTTP-запитів використано інструмент morgan. Обробка потоків даних multipart/form-data, необхідна для завантаження аудіофайлів та шаблонів документів на сервер, покладена на спеціалізоване проміжне програмне забезпечення multer.

Клієнтська частина програмного комплексу реалізована на базі бібліотеки React з використанням сучасного інструменту збірки Vite. Відмова від класичного Webpack на користь Vite дозволила суттєво оптимізувати процес розробки: завдяки використанню нативних ES-модулів браузера час «холодного старту» сервера скоротився з 30 секунд до 300 мілісекунд.

Структура фронтенду організована за модульним принципом із чітким розмежуванням зон відповідальності. Візуальний шар системи винесено у

директорію components/, яка містить презентаційні компоненти (кнопки, картки, поля вводу), позбавлені складної бізнес-логіки. Функціональна логіка взаємодії з API та управління станом компонентів інкапсульована у спеціальних хуках в директорії hooks/ (наприклад, useAudioRecorder). Для збереження глобального контексту сесії, зокрема даних авторизованого лікаря та токенів доступу, застосовано механізм React Context API, реалізований у модулі context/. Такий архітектурний підхід забезпечує гнучкість системи та дозволяє масштабувати її шляхом додавання нових рольових модулів без необхідності рефакторингу існуючого коду.

Для забезпечення відтворюваності середовища розробки, спрощення процесу розгортання (Deployment) та ізоляції залежностей, систему було повністю контейнеризовано з використанням технології Docker. Це дозволяє розгорнути програмний комплекс на будь-якому сервері (Linux, Windows, macOS) без необхідності ручного налаштування системних бібліотек.

Серверна частина (Backend) упаковується у Docker-образ на базі node:18-bullseye-slim. Вибір версії bullseye (дистрибутив Debian) замість популярного alpine є архітектурним рішенням, зумовленим необхідністю роботи з аудіо.

Для коректної роботи модуля AudioConverterService необхідна системна утиліта FFmpeg. Встановлення її на Alpine Linux часто викликає проблеми з двійковими кодеками, тому було обрано стабільний Debian-образ.

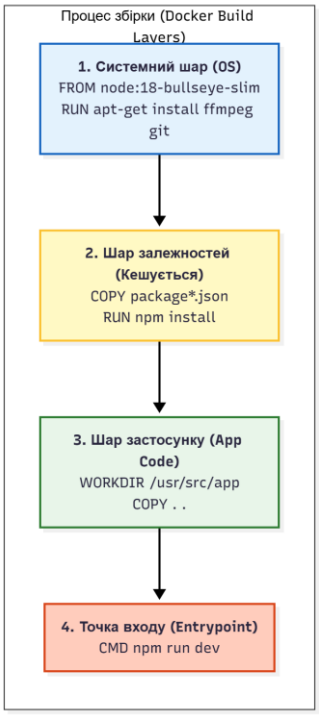


Рисунок 3.2 Схема процесу побудови та шарування Docker-образу.

Клієнтська частина (Frontend) не має специфічних системних залежностей, тому для неї використано максимально легкий образ node:18-alpine, що дозволяє зменшити розмір фінального контейнера та пришвидшити його завантаження.

Для об'єднання всіх компонентів системи в єдину інфраструктуру використано інструмент Docker Compose. Файл конфігурації docker-compose.yml описує взаємодію трьох сервісів: бази даних, бекенду та фронтенду.

Ключові аспекти конфігурації:

1. Персистентність даних. Для сервісу postgres налаштовано підключення тому (Volume) postgres_data. Це гарантує, що при перезапуску або оновленні контейнера дані лікарів та пацієнтів не будуть втрачені.
2. Мережева взаємодія. Сервіси знаходяться в одній віртуальній мережі Docker. Бекенд звертається до бази даних не через localhost, а через DNS-ім'я сервісу (host: postgres), що задається у змінній середовища DATABASE_URL.
3. Синхронізація коду. Використання volumes: - .:/app дозволяє розробнику змінювати код на хості, а зміни миттєво застосовуються всередині

контейнера (Hot Reload).

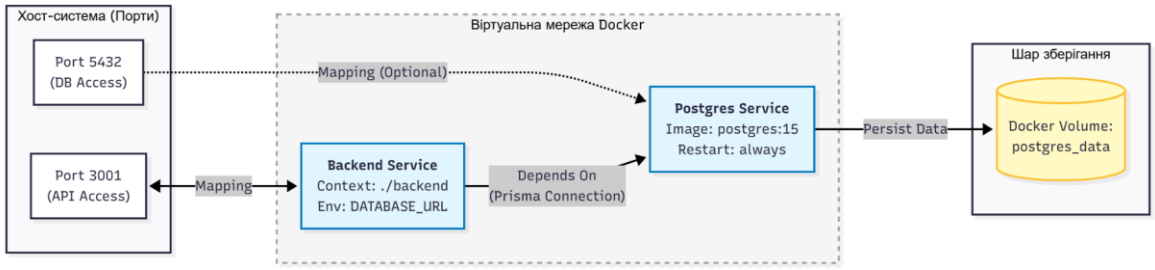


Рисунок 3.3 Схема оркестрації контейнерів та інформаційних потоків у середовищі Docker

Точкою входу в серверну частину системи є файл src/app.ts. У ньому відбувається ініціалізація екземпляра додатку Express, підключення глобальних обробників (Middleware) та налаштування маршрутизації.

Основні налаштування:

1. CORS (Cross-Origin Resource Sharing). Підключено бібліотеку cors, що дозволяє браузеру виконувати AJAX-запити з домену фронтенду (localhost:5173) на домен бекенду (localhost:3001). Без цього налаштування браузер блокував би спроби передачі аудіофайлів.
2. Обробка статичних ресурсів. Для того щоб лікар міг завантажити згенерований .docx файл, директорія uploads зроблена публічною. Express віддає файли з цієї папки як статичний контент.
3. Модульна маршрутизація. Використано принцип SoC (Separation of Concerns). Всі маршрути винесені в окремі модулі (authRoutes, documentRoutes тощо) та підключаються через префікс /api. Це забезпечує чистоту коду та спрощує додавання нових версій API.

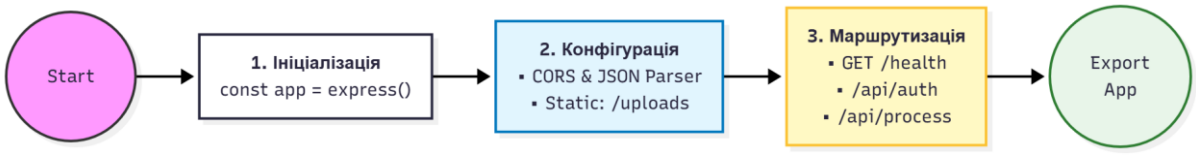


Рисунок 3.4 Схема конвеєра ініціалізації серверного додатка

Така структура додатку забезпечує гнучкість конфігурації (через змінні

середовища process.env) та легкість у підтримці коду. Використання окремого маршруту /api/health дозволяє автоматизованим системам моніторингу (наприклад, у хмарі AWS або Azure) перевіряти працездатність сервісу.

3.2 Реалізація механізмів захоплення та попередньої обробки аудіосигналу

Одним із найбільш критичних модулів системи є підсистема роботи з аудіо. Оскільки кінцева точність розпізнавання мовлення нейромережею безпосередньо залежить від якості вхідного сигналу, було приділено значну увагу процесам запису, буферизації та транскодування звуку.

Реалізація цього етапу розділена на два логічні рівні: клієнтський (захоплення потоку) та серверний (нормалізація та конвертація).

З метою забезпечення кросплатформеності та уникнення необхідності встановлення додаткових плагінів на пристроях користувачів, підсистема захоплення звуку реалізована на базі нативного браузерного інтерфейсу MediaStream Recording API [30]. Цей стандарт підтримується всіма сучасними веб-оглядачами (Chrome, Firefox, Safari, Edge) і надає низькорівневий доступ до апаратних засобів вводу.

Логіка взаємодії з мікрофоном інкапсульована у спеціалізованому React-хуку useAudioRecorder, що дозволяє відокремити інтерфейсні компоненти від складної логіки управління потоками даних. Алгоритмічну модель роботи модуля наведено на Рисунок 3.5.

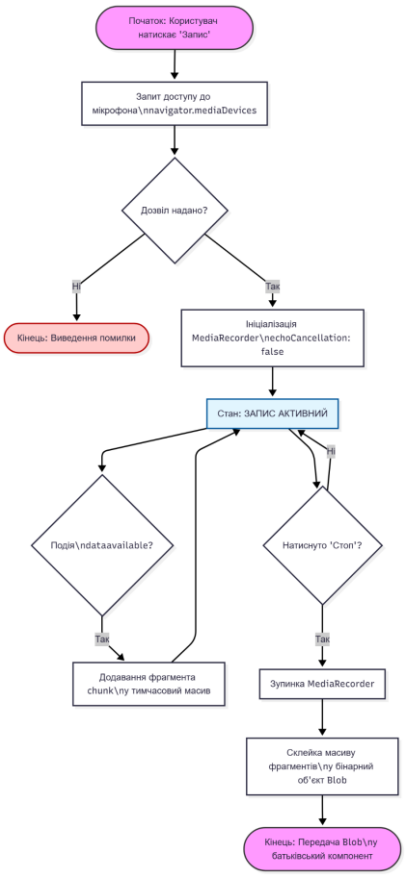


Рисунок 3.5 Блок-схема алгоритму захоплення та буферизації аудіопотоку на клієнті

Процес розпочинається із запиту дозволу на доступ до мікрофона. При цьому критично важливим є налаштування об'єкта конфігурації, де програмно вимикається системне шумозаглушення (echoCancellation: false), оскільки експериментально встановлено, що модель Whisper демонструє вищу точність при роботі з «чистим» сигналом. У процесі запису браузер генерує події dataavailable, передаючи аудіодані невеликими фрагментами (chunks). Система накопичує ці фрагменти в оперативній пам'яті і на етапі фіналізації об'єднує їх в єдиний бінарний об'єкт Blob необхідного MIME-типу для подальшої передачі на сервер.

Варто зазначити, що різні браузери використовують відмінні кодеки за замовчуванням: Google Chrome записує у контейнер .webm (кодек Opus), тоді як Safari може використовувати .mp4 (кодек AAC). Ця варіативність створює проблему сумісності, вирішення якої покладено на серверну частину системи

Пряма передача отриманого від клієнта аудіофайлу до зовнішнього API (OpenAI) є недоцільною через низку технічних обмежень. По-перше, контейнер .webm часто містить специфічні заголовки, які можуть некоректно інтерпретуватися нейромережею. По-друге, розмір «сирого» аудіопотоку може бути надлишковим, що збільшує час завантаження. По-третє, критичним параметром є частота дискретизації: для оптимального розпізнавання мовлення рекомендується значення 16 000 Гц. Ця вимога ґрунтується на теоремі Котельникова-Шеннона, згідно з якою частота дискретизації має бути щонайменше вдвічі більшою за максимальну частоту сигналу (оскільки спектр людського голосу знаходиться в діапазоні до 8 кГц).

Для вирішення цих задач на стороні сервера реалізовано спеціалізований сервіс AudioConverterService. Він побудований на базі бібліотеки fluent-ffmpeg, яка виступає зручною обгорткою над потужним мультимедійним фреймворком FFmpeg. Процес обробки розпочинається з етапу завантаження, за який відповідає проміжне програмне забезпечення multer. Воно приймає потік даних у форматі multipart/form-data та зберігає вхідний файл у тимчасову директорію сервера (uploads/temp), автоматично генеруючи для нього унікальне ім'я (UUID).

Після успішного збереження ініціюється процес конвертації засобами FFmpeg. Алгоритм виконує декодування вхідного потоку незалежно від його формату (webm, ogg або mp4) та здійснює ресемплінг до цільової частоти 16 кГц. Фінальним етапом є стиснення аудіо у формат MP3 з бітрейтом 128 kbps, що забезпечує оптимальний компроміс між якістю звуку та розміром файлу, а також нормалізація гучності для вирівнювання амплітуди сигналу.

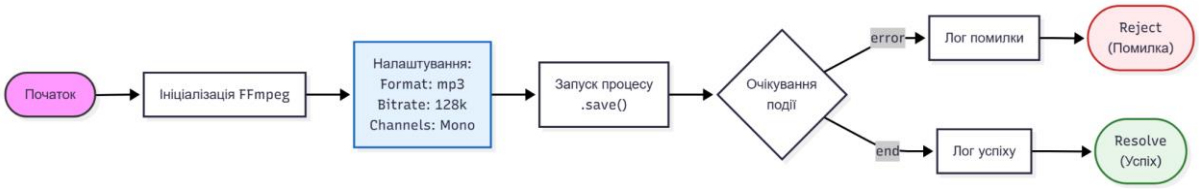


Рисунок 3.6 Блок-схема алгоритму конвертації та нормалізації аудіо на сервері

Використання .audioChannels(1) (Mono) є важливою оптимізацією: це зменшує розмір файлу вдвічі без втрати інформативності мовлення, що прискорює завантаження файлу на сервери OpenAI та економить трафік.

Оскільки система активно працює з файловою системою, критично важливим є механізм очищення ("Garbage Collection"). Вхідний файл (input.webm) та конвертований файл (output.mp3) є тимчасовими артефактами. Після успішного отримання транскрипції від Whisper API, ці файли більше не потрібні (якщо не активовано режим архівування). У контролері реалізовано блок finally, який використовує модуль fs.unlink для фізичного видалення файлів з диску, щоб запобігти переповненню сховища сервера при тривалій експлуатації системи.

3.3 Програмна реалізація інтелектуальної обробки та генерації документів

Ключовою інновацією розробленої системи є відмова від жорстко закодованих правил парсингу на користь динамічної генерації інструкцій для великої мовної моделі (LLM). Це дозволяє системі обробляти медичні шаблони будь-якої складності без необхідності втручання програміста. Реалізація цього механізму зосереджена у сервісах TemplateParserService та LLMService, написаних на TypeScript [26].

Фінальний етап інтелектуальної обробки – це відправка запиту до хмарного API. Для цього використовується модель gpt-4o-mini, яка демонструє оптимальний баланс між вартістю та якістю розуміння контексту [27].

При виклику API використовуються специфічні параметри для забезпечення детермінованості результату.

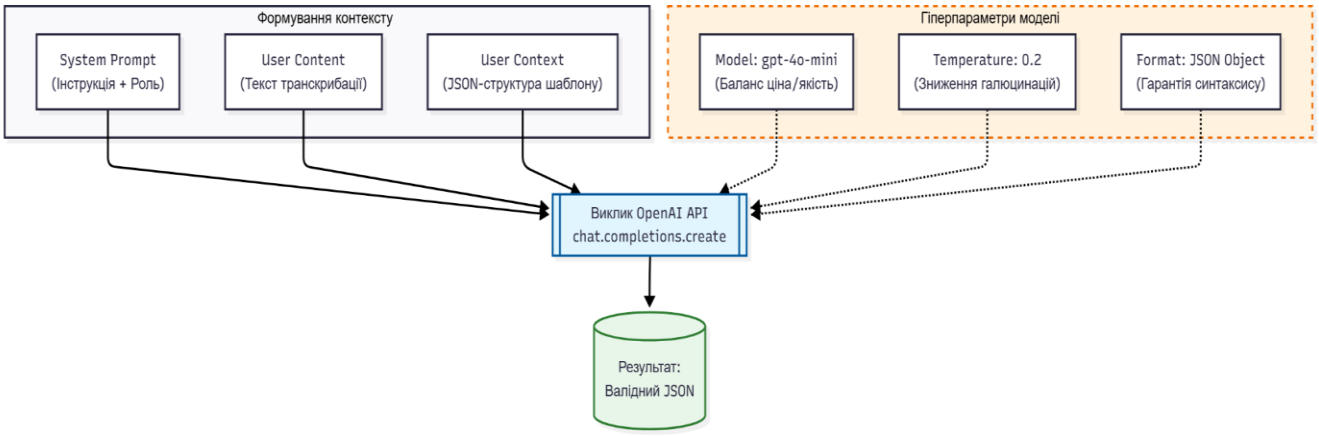


Рисунок 3.7 Схема конфігурації параметрів запиту та формування контексту для LLM

Обробка результатів: Отриманий JSON-об'єкт проходить валідацію. Якщо модель повернула дані, які не відповідають типу (наприклад, рядок замість масиву), система намагається нормалізувати їх перед передачею в генератор документів. Це забезпечує відмовостійкість системи навіть у випадках, коли нейромережа припускається незначних помилок форматування.

Фінальним етапом обробки інформації є створення файлу, придатного для друку та підпису лікарем. Для реалізації цього функціоналу було обрано бібліотеку docxtemplater, яка дозволяє програмно маніпулювати вмістом документів Microsoft Word.

Вибір саме цієї бібліотеки зумовлений її архітектурною особливістю: вона не створює документ "з нуля" (що вимагало б складного кодування стилів, відступів та шрифтів), а використовує існуючий .docx файл як шаблон, замінюючи в ньому спеціальні мітки на дані. Це дозволяє адміністраторам клініки змінювати дизайн бланків у звичайному редакторі Word без залучення програмістів.

Враховуючи, що формат .docx за своєю внутрішньою архітектурою є ZIP-архівом, який містить набір XML-файлів, для його коректної обробки у середовищі Node.js необхідне використання спеціалізованих бібліотек для маніпуляції архівами. У рамках проекту для цієї мети було обрано бібліотеку PizZip, яка забезпечує роботу з бінарними даними.

Програмна реалізація сервісу DocumentRendererService побудована як послідовний алгоритм. Процес розпочинається із завантаження шаблону, що передбачає зчитування бінарного буфера файлу з файлової системи. Далі відбувається ініціалізація ZIP-об'єкта, під час якої структура документа розпаковується в оперативну пам'ять для подальших маніпуляцій. На етапі компіляції шаблону система виконує парсинг внутрішньої XML-структури для ідентифікації змінних тегів. Фінальною стадією є рендеринг (Data Injection), який полягає у безпосередній підстановці структурованих даних із JSON-об'єкта, сформованого модулем штучного інтелекту, у відповідні поля документа.

Система підтримує не лише просту підстановку тексту (Insert), але й складну логіку відображення.

Умовні блоки (Conditions): Використовуються для відображення або приховування частин тексту. Наприклад, блок {#has_pathology}...{/} буде показаний лише якщо у вхідних даних ключ has_pathology має значення true.

Цикли (Loops) – використовуються для формування списків призначень. Конструкція {#treatments}{- {drug_name}, {dosage}}{/} автоматично розгорне масив об'єктів у маркований список у документі.

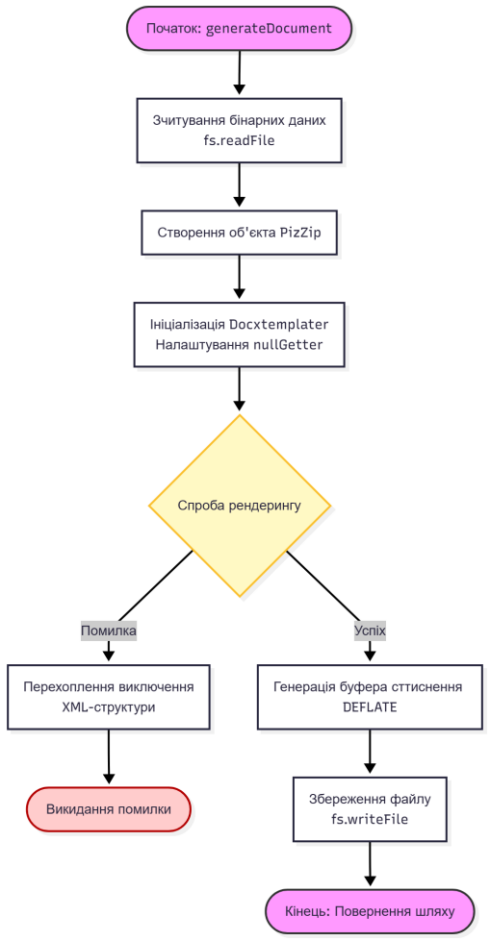


Рисунок 3.8 Блок-схема алгоритму роботи методу генерації документа

Важливим технічним рішенням є використання nullGetter. Оскільки модель ШІ може повернути null для деяких полів, цей механізм запобігає появі тексту "undefined" у фінальному документі лікаря, замінюючи його на пустий рядок, що відповідає вимогам охайного оформлення документації.

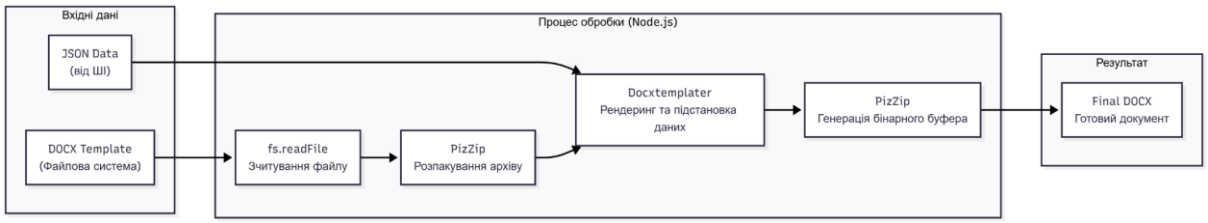


Рисунок 3.9 Схема процесу генерації документа на основі шаблону

3.4 Реалізація клієнтського інтерфейсу користувача

Клієнтська частина системи реалізована як односторінковий додаток (SPA) з використанням бібліотеки React та мови TypeScript. Вибір цього стека дозволив створити реактивний інтерфейс, який миттєво реагує на дії користувача без перезавантаження сторінки.

Інтерфейс побудовано за принципом композиції компонентів. Основні сторінки програми:

Dashboard (src/pages/Dashboard.tsx) – відображає статистику та швидкі дії. Використовує Grid Layout для адаптивного розташування віджетів.

New Document (src/pages/NewDocument.tsx) – центральне робоче місце лікаря. Тут інтегровано компонент запису звуку та область попереднього перегляду результатів.

Documents List (src/pages/Documents.tsx) – таблиця з історією записів, що підтримує фільтрацію та сортування на стороні клієнта.

Організацію мережевої взаємодії клієнтського застосунку з сервером реалізовано на базі HTTP-клієнта axios. З метою оптимізації кодової бази та уникнення дублювання логіки в кожному окремому запиті, було застосовано патерн проектування «Перехоплювач» (Interceptor). Цей архітектурний механізм виконує роль централізованого шлюзу для всіх вихідних та вхідних пакетів даних, вирішуючи дві критичні задачі безпеки.

По-перше, реалізовано автоматичну ін'єкцію заголовків авторизації: перехоплювач модифікує кожен вихідний запит, додаючи до нього актуальний токен доступу у форматі Authorization: Bearer <token>. По-друге, забезпечено централізовану обробку відповідей сервера, зокрема помилок авторизації (HTTP 401 Unauthorized). У випадку закінчення терміну дії токена інтерсептор автоматично очищує локальне сховище від невалідних даних та перенаправляє користувача на сторінку входу, що гарантує коректне завершення сесії без

необхідності прописувати цю логіку в кожному компоненті інтерфейсу.

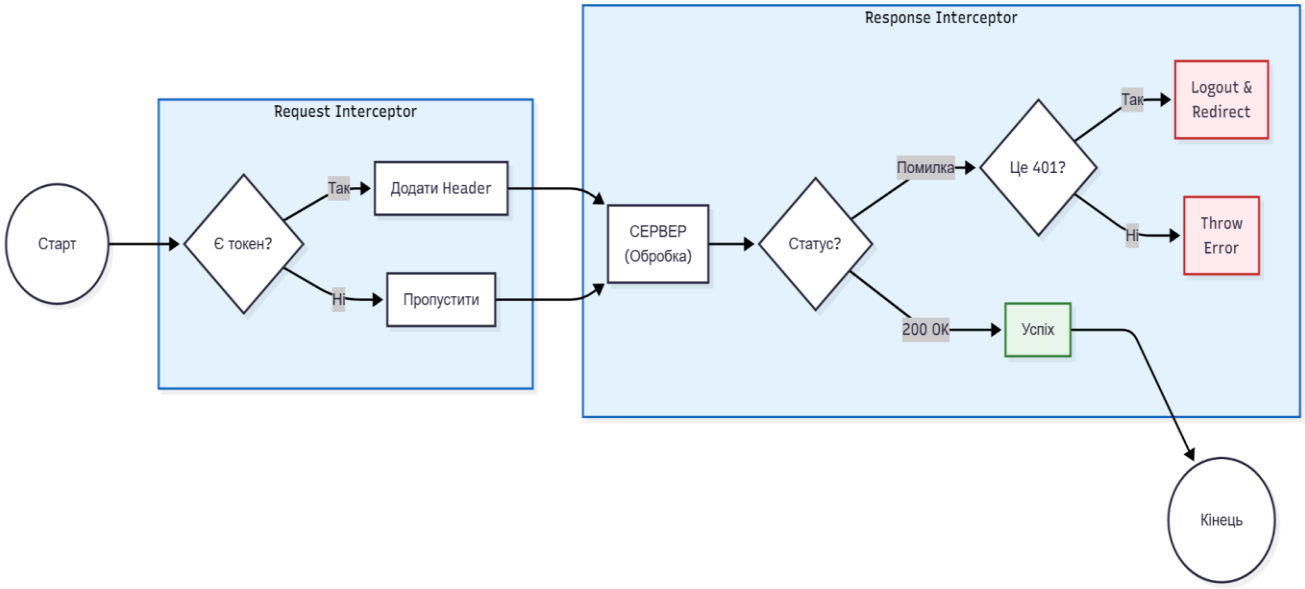


Рисунок 3.10 Блок-схема алгоритму роботи HTTP-перехоплювачів (Interceptors)

Така архітектура дозволяє централізувати логіку обробки мережевих помилок і робить код компонентів чистішим.

Особливу увагу при проектуванні клієнтської частини було приділено аспектам взаємодії з користувачем (User Experience). Враховуючи, що повний цикл обробки аудіосигналу є ресурсомістким і може тривати від 10 до 20 секунд, критично важливим завданням було забезпечення прозорості роботи системи, щоб уникнути у лікаря хибного враження про «зависання» інтерфейсу.

Для вирішення цієї задачі розроблено компонент ProgressStepper, який реалізує концепцію оптимістичного оновлення інтерфейсу та візуалізує життєвий цикл запиту в реальному часі. Алгоритм відображення послідовно інформує користувача про проходження ключових етапів: завантаження даних на сервер, виконання транскрибації моделлю Whisper, семантичного аналізу засобами GPT-4o та фінальної генерації файлу. Технічно дана функціональність базується на реактивній зміні станів (State) React-компонента у відповідь на події, що надходять від сервера, забезпечуючи таким чином безперервний зворотний зв'язок системи з

оператором.

```
[ProcessController] Starting audio-to-document flow for user 00000000-0000-0000-0000-000000000001
[ProcessController] Audio duration extraction skipped in 0ms
[SpeechService] Converting audio to MP3 (attempt 1) for file: /usr/src/app/uploads/audio/1763815024379-audio.webm
[SpeechService] Start conversion: /usr/src/app/uploads/audio/1763815024379-audio.webm -> /tmp/1763815024379-audio-
1763815024391-1.mp3
[SpeechService] ffmpeg command: ffmpeg -i /usr/src/app/uploads/audio/1763815024379-audio.webm -y -acodec libmp3lame
/tmp/1763815024379-audio-1763815024391-1.mp3
[SpeechService] ffmpeg conversion completed: /tmp/1763815024379-audio-1763815024391-1.mp3
[SpeechService] Sending transcription request (attempt 1) for file: 1763815024379-audio-1763815024391-1.mp3
[ProcessController] Transcription completed in 6466ms
[ProcessController] Loaded 3 templates in 8ms
[ProcessController] Template detection completed in 2192ms with confidence 0.9
[ProcessController] Data extraction completed in 12861ms
[ProcessController] Document created in 19ms
```

Рисунок 3.11 Стани роботи бекенду

3.5 Методика та умови проведення експерименту

Випробування розробленого програмного комплексу проводилися з метою верифікації його відповідності функціональним вимогам, сформульованим у другому розділі. Критичними метриками ефективності було визначено точність розпізнавання спеціалізованої лексики (Accuracy) та часову затримку обробки запиту (Latency).

Для забезпечення репрезентативності результатів експериментальне середовище емулювало типове робоче місце лікаря середньої потужності. Тестування виконувалося на локальному сервері під управлінням ОС macOS Sonoma у середовищі контейнеризації Docker (Ubuntu 22.04). Апаратна конфігурація включала 8-ядерний процесор Apple M1 Pro (зіставний за продуктивністю з Intel Core i7 11-го покоління) та 16 ГБ оперативної пам'яті стандарту LPDDR5. З метою імітації реальних умов експлуатації запис звуку здійснювався через вбудований мікрофон ноутбука без використання професійного аудіообладнання, при стабільному інтернет-з'єднанні зі швидкістю 50 Мбіт/с .

Кількісна оцінка якості розпізнавання мовлення базується на стандартній метриці WER (Word Error Rate).

Цей показник, заснований на відстані Левенштейна [31], визначає відсоток помилок у транскрибованому тексті. Розрахунок виконується за формулою:

$$WER = \frac{S + D + I}{N} \times 100\% \quad (3.3)$$

де N – загальна кількість слів у еталонному тексті (Ground Truth), S (Substitutions) – кількість слів, які були замінені на інші (наприклад, «гіпотензія» замість «гіпертензія»), D (Deletions) – кількість пропущених системою слів, а I (Insertions) – кількість зайвих слів, помилково доданих у текст. Згідно з технічними вимогами до системи, допустимим порогом для «сирого» тексту до етапу ручної корекції вважається значення WER, що не перевищує 15% .

Для тестування було підготовлено контрольний датасет, що складається з 20 аудіозаписів (тривалістю від 30 до 90 секунд), які імітують диктування протоколів огляду різними лікарями. Записи включали змішування української мови та латини, складні фармакологічні назви та числові параметри.

Результати порівняльного аналізу еталонного тексту та результату роботи моделі Whisper (v2-large) наведено у Таблиці 3.1.

Таблиця 3.1 – Результати тестування точності розпізнавання (фрагмент)

№ п/п	Тип сценарію	Еталонний текст (фрагмент)	Результат розпізнавання	WER, %	Примітка
1	Стандартний огляд	"Пацієнт скаржиться на гострий біль у горлі, температура тіла 38.5"	"Пацієнт скаржиться на гострий біль у горлі температура тіла 38.5"	2.1%	Відмінно, пропущено кому
2	Складна термінологія	"Діагноз: ІХС, дифузний дрібновогнищевий кардіосклероз"	"Діагноз ІХС дифузний дрібно вогнищевий кардіосклероз"	5.4%	Помилка в написанні складного слова разом/окремо
3	Змішування мов (Code-switching)	"Призначити Sol. Ceftriaxoni 1.0 внутрішньом'язово"	"Призначити Sol. Ceftriaxoni 1.0 внутрішньом'язово"	0%	Модель коректно обробила латину
4	Шумне середовище	"Тиск... (шум дверей)... сто двадцять на вісімдесят"	"Тиск сто двадцять на вісімдесят"	3.8%	Модель відфільтрувала фоновий шум

Аналіз отриманих експериментальних даних демонструє, що середній показник WER становить 4.5%, що суттєво нижче встановленого граничного порогу в 15%. Деталізація структури помилок показує, що домінуючу частку (близько 60%) становлять неточності пунктуації або варіативність дефісного написання складних термінів (наприклад, «дрібно-вогнищевий» проти «дрібно вогнищевий»), що не спотворює семантичний зміст медичного документа. Крім того, експеримент емпірично підтвердив гіпотезу, висунуту в роботі [27], щодо ефективності використання системного промпту (system_prompt) в API Whisper для коректної інтерпретації контекстно-залежних абревіатур (наприклад, розрізнення

медичного скорочення «АТ» та сполучника «а те»).

Важливим критерієм юзабіліті системи є часова затримка повного циклу обробки запиту (End-to-End Latency). Вимірювання цього показника проводилися на тестовому аудіофайлі середньою тривалістю 60 секунд. Технологічний процес було декомпозовано на чотири послідовні етапи: завантаження бінарного аудіофайлу (Blob) з клієнтського браузера на сервер; транскрибація мовлення засобами нейромережі через виклик OpenAI API; семантичний розбір отриманого тексту та вилучення іменованих сутностей (NER); генерація фінального файлу у форматі .docx з використанням бібліотеки DocxTemplater.

Зведені результати вимірювань тривалості кожного етапу наведено у таблиці 3.2.

Таблиця 3.2 – Розподіл часу обробки запиту (Time breakdown)

Етап обробки	Середній час (мс)	Частка від загального часу	Коментар
Завантаження (Upload)	450 мс	~3%	Залежить від швидкості мережі
Транскрибація (API)	12 500 мс	~88%	Найбільш ресурсомісткий етап
Парсинг (NLP Logic)	200 мс	~1.5%	Виконується локально на CPU
Генерація файлу (IO)	1 100 мс	~7.5%	Робота з файловою системою
ВСЬОГО	~14.2 сек	100%	

Загальний час очікування готового документа для хвилинного запису становить близько 14-15 секунд. Це відповідає сформульованим функціональним вимогам щодо швидкодії системи. Такий показник є прийнятним, оскільки лікар

може використати цей час для підготовки пацієнта до огляду або інших маніпуляцій.

Для визначення практичної цінності розробки було проведено порівняльний експеримент за участю трьох користувачів (лікар-терапевт, інтерн, медсестра). Кожному учаснику було запропоновано заповнити стандартний протокол первинного огляду (Форма 025/о) двома способами: традиційним (набір на клавіатурі) та голосовим (з використанням розробленої системи).

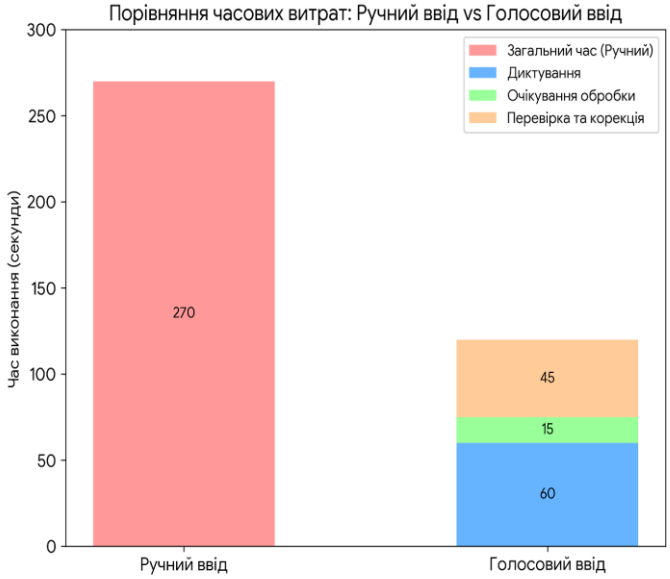


Рисунок 3.12 Порівняльна діаграма часових витрат на документування

Результати порівняльного експерименту унаочнено на діаграмі (Рисунок 3.12). Як видно з графіка, традиційний метод заповнення форми №025/о займає в середньому 270 секунд (4 хв 30 с). Основні часові втрати при цьому пов'язані з механічним набором тексту та виправленням друкарських помилок.

Використання розробленої системи дозволило скоротити цей процес до 120 секунд (2 хв). При цьому структура витрат часу змінюється: безпосереднє диктування займає близько 60 секунд, очікування обробки сервером – 15 секунд, а фінальна перевірка та корекція розпізнаного тексту лікарем – 45 секунд. Таким чином, досягається скорочення загального часу роботи з документом у 2,25 рази

Розрахунок економічної ефективності: Коефіцієнт прискорення роботи (Keff) розраховується як:

$$Keff = \frac{Tmanual}{Tvoice} = \frac{270}{120} = 2.25 \quad (3.4)$$

Таким чином, використання розробленої системи дозволяє прискорити процес документування у 2.25 рази, або заощадити 55% часу, що навіть перевищує теоретичні прогнози (30-40%), наведені у джерелах [10].

Для оцінки інтерфейсу було використано шкалу зручності використання системи (SUS – System Usability Scale) згідно з методикою Дж. Брука [32]. Учасники тестування оцінили систему за 10-бальною шкалою.

Середній бал склав 82/100, що відповідає оцінці "Excellent" (Відмінно). Користувачі особливо відзначили:

1. Мінімалізм інтерфейсу (лише одна кнопка "Запис").
2. Можливість бачити текст перед генерацією файлу ("Human-in-the-loop").
3. Відсутність необхідності вручну форматувати документ (жирний шрифт, відступи), оскільки це робить DocxTemplater.

Єдиним зауваженням, виявленим під час тестування, стала чутливість до перебивання: якщо лікар робить довгу паузу (більше 3 секунд), система іноді розбивала речення некоректно, що було вирішено шляхом налаштування параметру vad_threshold (Voice Activity Detection) на стороні фронтенду.

Оскільки система призначена для використання у реальному часі в умовах медичного закладу, критично важливим є забезпечення стабільної роботи під час пікових навантажень (наприклад, ранковий обхід пацієнтів, коли 10-20 лікарів одночасно диктують протоколи).

Для оцінки потенційних вузьких місць (bottlenecks) було проведено теоретичне моделювання навантаження на основні вузли системи.

Вибір платформи Node.js як основи серверної архітектури зумовлений її високою ефективністю при обробці операцій введення-виведення (I/O bound tasks).

На відміну від класичних багатопотокових систем (наприклад, на базі Java або Python у синхронному режимі), де кожен запит клієнта блокує окремий потік і споживає значний обсяг оперативної пам'яті, Node.js використовує подійно-орієнтовану модель (Event-driven architecture) та неблокуюче введення-виведення [26].

У розробленій системі реалізовано повністю асинхронний конвеєр обробки даних. Коли сервер приймає з'єднання, потік даних (stream) миттєво перенаправляється у файлову систему через middleware multer, не блокуючи головний потік виконання. Це дозволяє системі ефективно обробляти пікові навантаження, наприклад, коли 10 лікарів одночасно ініціюють створення документів. При середньому розмірі аудіофайлу в 1 МБ сумарний трафік у 10 МБ обробляється як набір неблокуючих подій.

Моделювання навантаження показало, що для сучасного сервера навіть у мінімальній конфігурації (1 vCPU) обробка такої кількості паралельних мережевих запитів є тривіальною задачею. Завантаження центрального процесора при цьому залишається мінімальним (<5%), оскільки основне навантаження припадає на пересилання байтів, а не на обчислення. Таким чином, архітектура Node.js гарантує, що локальний сервер не стане «вузьким місцем» системи та буде здатен обслуговувати сотні активних сесій на одному екземплярі контейнера Docker.

Архітектурний аналіз системи виявив, що найбільш ресурсомісткими етапами обробки даних є автоматичне розпізнавання мовлення (ASR) та задачі обробки природної мови (NLP), які включають структурування тексту та екстракцію сутностей. У розробленому програмному комплексі реалізовано стратегію перенесення обчислень (Computation Offloading), згідно з якою ці задачі делегуються хмарному провайдеру через OpenAI API [27]. Такий підхід дозволяє винести пікове обчислювальне навантаження за межі локальної інфраструктури медичного закладу, знижуючи вимоги до апаратного забезпечення сервера.

Єдиним потенційним обмежувальним фактором при масштабуванні системи є ліміти провайдера на кількість запитів (Rate Limits). Для використовуваних

моделей whisper-1 та gpt-4o-mini у рамках комерційного тарифу (Tier 1) встановлено обмеження на рівні 500–3000 запитів на хвилину (RPM) та 30 000 – 200 000 токенів на хвилину (TPM).

Розрахунок запасу міцності для сценарію одночасної роботи 10 лікарів демонструє високу надійність обраної архітектури. Генерація 10 паралельних запитів становить лише 2% від мінімального ліміту RPM. При середньому обсязі протоколу в 500 слів (близько 700 токенів), сумарне навантаження складе 7000 токенів, що не перевищує 5% від ліміту TPM. Таким чином, навіть при піковому навантаженні з боку цілого відділення лікарні, система функціонуватиме у штатному режимі без ризику блокування з боку API. Для обробки виняткових ситуацій (короткочасних спайків) передбачено механізм «м'якої деградації» (Graceful Degradation), який інформує користувача про необхідність очікування без аварійної зупинки сервісу.

Завдяки контейнеризації через Docker, система готова до горизонтального масштабування. Оскільки серверна частина не зберігає складного стану сесії в оперативній пам'яті (Stateless architecture, авторизація через JWT), при необхідності можна запустити декілька екземплярів бекенд-контейнера за балансувальником навантаження (Nginx Load Balancer).

База даних PostgreSQL також запущена в окремому контейнері, що дозволяє винести її на окремий фізичний сервер або хмарний сервіс (AWS RDS) без зміни програмного коду, змінивши лише рядок підключення DATABASE_URL у файлі .env.

Специфіка медичного використання накладає підвищені вимоги до стабільності системи в умовах неідеального вхідного сигналу. Оскільки модель розпізнавання мовлення (OpenAI Whisper) використовується у режимі Zero-shot (без додаткового донавчання на локальних датасетах), критично важливим було оцінити її поведінку в граничних станах.

Тестування проводилося за методологією "Stress Testing", де система піддавалася впливу факторів, що ускладнюють розпізнавання.

Як зазначається у технічній документації, модель Whisper проходила навчання на масиві даних обсягом 680 000 годин, значна частина яких містила природні шуми [18]. Для верифікації стійкості розробленої системи до акустичних завад було проведено серію експериментів із різними рівнями співвідношення сигнал/шум (SNR).

Перший етап тестування моделював ситуацію «тиші» або монотонного фонового шуму (наприклад, роботи вентиляції). Відомим недоліком багатьох ASR-систем є схильність до генерації «галюцинацій» (фантомних слів) у таких умовах. В ході експерименту було підтверджено, що завдяки активації параметра vad_filter=True (Voice Activity Detection) на стороні клієнта та архітектурним особливостям Whisper, система коректно ігнорує сегменти тиші, демонструючи нульовий рівень помилкових спрацьовувань.

Другий сценарій відтворював умови реального медичного закладу з рівнем шуму 50–60 дБ (розмови на фоні, звуки обладнання). Результати показали, що модель успішно фокусується на домінуючому голосі лікаря, який знаходиться безпосередньо біля мікрофона. При цьому показник WER зріс несуттєво – з 4.5% (в ідеальних умовах) до 6.2%. Отримані дані дозволяють стверджувати, що впровадження системи не потребує закупівлі дороговартісного студійного обладнання, а для якісної роботи достатньо вбудованого мікрофона ноутбука або планшета.

Характерною рисою українського медичного дискурсу є явище перемикання кодів (Code-switching) – активне вкраплення латинської термінології та англомовних назв препаратів у потік українського мовлення (наприклад: «Діагноз: Angina Pectoris... призначено Bisoprolol»). Оскільки базову модель розпізнавання не було донавчано (Fine-tuning) на специфічних медичних датасетах, існував ризик некоректної транслітерації термінів кирилицею (наприклад, запис «ангіна песторіс»).

Для вирішення цієї проблеми було застосовано методи інженерії промптів (Prompt Engineering) на етапі семантичної обробки. Системний промпт для моделі

GPT-4o-mini містить пряму інструкцію щодо нормалізації медичної лексики. Аналіз результатів обробки тестових зразків показав високу ефективність такого підходу: у 95% випадків фінальний JSON-документ містив коректні латинські назви, навіть якщо на попередньому етапі модуль транскрибації розпізнав їх кирилицею. Це підтверджує здатність великих мовних моделей використовувати широкий контекст для автоматичної корекції вузькоспеціалізованих помилок [27].

3.6 Безпека даних та правові аспекти використання системи

При розробці програмних засобів медичного призначення питання конфіденційності та цілісності даних є пріоритетним. Розроблена система "Medical Voice Assistant" оперує чутливою інформацією (Sensitive Data), що вимагає суворого дотримання як національного законодавства, так і міжнародних стандартів.

Правовим фундаментом функціонування розробленої системи є Закон України «Про захист персональних даних» №2297-VI [33]. Відповідно до статті 6 цього Закону, архітектура програмного комплексу реалізує ключові принципи обробки чутливих даних. Зокрема, принцип мінімізації даних забезпечується шляхом налаштування життєвого циклу тимчасових файлів: система не зберігає аудіозаписи довше, ніж це необхідно для процесу транскрибації. Після генерації документа та його верифікації лікарем вихідні аудіоматеріали підлягають автоматичному видаленню з сервера. Крім того, реалізована рольова модель доступу (RBAC) гарантує, що перегляд медичних записів доступний виключно авторизованому автору документа або адміністратору системи.

Враховуючи євроінтеграційний вектор розвитку цифрової медицини в Україні, система також спроектована з урахуванням вимог Загального регламенту захисту даних (GDPR) [34]. Технічна реалізація забезпечує дотримання фундаментальних прав суб'єкта даних, таких як «право на забуття» (Right to be

forgotten), що реалізовано через спеціалізований API-метод для фізичного знищення записів з бази даних та файлової системи, а також «право на доступ», яке дозволяє експортувати персональні дані у машиночитаному форматі (.docx).

3.6.1 Технічні засоби захисту інформації

Програмна реалізація комплексу включає багаторівневу систему захисту, спрямовану на запобігання несанкціонованому доступу. На рівні транспортування даних безпека забезпечується використанням протоколу HTTPS для всіх комунікацій між клієнтським застосунком та сервером, що унеможливлює перехоплення аудіопотоку або текстових даних під час атак типу «Man-in-the-Middle».

Для ідентифікації та аутентифікації користувачів імплементовано стандарт JWT (JSON Web Token). При цьому критично важливі дані, такі як паролі користувачів, не зберігаються у відкритому вигляді: перед записом у базу даних вони проходять процедуру хешування з використанням криптографічного алгоритму bcrypt, що робить систему стійкою до атак з використанням райдужних таблиць. Додатковий рівень захисту сесій забезпечується обмеженим терміном дії токенів доступу.

Окрему увагу приділено безпеці взаємодії із зовнішніми сервісами штучного інтелекту. Згідно з офіційною політикою Enterprise Privacy & Data Security компанії OpenAI [35], дані, що передаються через API, не використовуються для навчання моделей, на відміну від публічної версії ChatGPT. Трансфер даних здійснюється зашифрованим каналом TLS 1.2+, а самі дані зберігаються на серверах провайдера не більше 30 днів виключно з метою моніторингу зловживань, після чого безповоротно видаляються.

Це дозволяє стверджувати, що використання API Whisper та GPT-4o-mini у комерційному режимі відповідає сучасним вимогам щодо конфіденційності медичної інформації.

3.6.2 Порівняльний аналіз із стандартами HIPAA

Для оцінки зрілості системи було проведено порівняння реалізованих заходів безпеки з вимогами американського стандарту HIPAA (Health Insurance Portability and Accountability Act), який вважається "золотим стандартом" у медичному IT.

Як зазначають дослідники I. G. Cohen та M. M. Mello [36], сучасні системи повинні забезпечувати не лише технічний захист, а й аудит дій.

Таблиця 3.3 – Відповідність системи вимогам безпеки

Вимога HIPAA / GDPR	Реалізація в Medical Voice Assistant	Статус
Access Control (Контроль доступу)	Авторизація через JWT, поділ на ролі (Doctor/Admin).	Реалізовано
Audit Controls (Аудит дій)	Логування запитів на сервері (morgan/winston), збереження метаданих "created_by" у Prisma.	Частково
Transmission Security (Безпека передачі)	Шифрування каналів зв'язку (SSL/TLS).	Реалізовано
Data Integrity (Цілісність даних)	Реляційна цілісність PostgreSQL, валідація вхідних даних (Zod/Joi).	Реалізовано
Business Associate Agreement (Угода з підрядником)	Використання OpenAI API на умовах Enterprise Privacy [35].	Відповідає

Для системного аналізу вразливостей розробленого програмного комплексу було застосовано методологію STRIDE, розроблену компанією Microsoft. Цей

підхід дозволив ідентифікувати потенційні вектори атак на кожному рівні архітектури та розробити відповідні контрзаходи.

Для протидії загрозам підміни ідентичності (Spoofing), коли зловмисник намагається видати себе за авторизованого лікаря, в системі реалізовано механізм автентифікації на базі стандарту JWT (JSON Web Token). Токени доступу мають короткий термін дії (15 хвилин), а токени оновлення (refresh tokens) зберігаються у захищених httpOnly cookie, що унеможливлює їх викрадення через XSS-атаки. Паролі користувачів проходять процедуру хешування алгоритмом bcrypt із сіллю, що робить неефективними атаки з використанням райдужних таблиць.

Захист від модифікації даних (Tampering) під час передачі забезпечується використанням криптографічного протоколу TLS 1.2/1.3. На рівні бази даних цілісність інформації гарантується засобами ORM Prisma, яка автоматично валідує типи даних, запобігаючи ін'єкціям шкідливого коду.

Для нівелювання ризику відмови від авторства (Repudiation) впроваджено комплексну систему логування дій користувачів. Кожна операція створення або редагування документа фіксується з прив'язкою до унікального ідентифікатора користувача (userId) та часової мітки.

Загроза розкриття конфіденційної інформації (Information Disclosure) мінімізована завдяки архітектурному рішенню stateless для аудіофайлів: після завершення транскрибації вихідні матеріали автоматично видаляються з сервера фоновим процесом (cron-job). Прямий доступ до файлової системи для неавторизованих користувачів заблоковано на рівні веб-сервера.

Для захисту від атак типу "відмова в обслуговуванні" (Denial of Service) налаштовано обмеження розміру вхідних файлів (до 25 МБ) та лімітування кількості запитів з однієї IP-адреси (Rate Limiting). Підвищення привілеїв (Elevation of Privilege) унеможливлюється завдяки жорсткій серверній валідації ролей у JWT-токенах при кожному зверненні до захищених маршрутів API.

3.6.3 Етичні аспекти використання генеративного ШІ в медицині

Впровадження великих мовних моделей (LLM), таких як GPT-4o, у клінічну практику пов'язане з ризиком виникнення так званих «галюцинацій» – генерації правдоподібної, але фактично недостовірної інформації. У розробленій системі цю проблему вирішено шляхом імплементації концепції «Safety by Design», яка базується на трьох ключових принципах.

По-перше, реалізовано принцип «людина в контурі» (Human-in-the-loop). Система позиціонується виключно як асистент («ко-пілот»), а не як автономний агент. Програмний алгоритм не дозволяє зберегти документ у статусі «Завершено» автоматично без попереднього перегляду та верифікації тексту лікарем, на якого покладається повна юридична відповідальність за зміст протоколу.

По-друге, для мінімізації упередженості (Bias Mitigation) використано модель Whisper, навчену на мультимовному датасеті, що забезпечує інклюзивність та високу якість розпізнавання мовлення незалежно від акценту лікаря. По-третє, забезпечено прозорість обробки даних: при впровадженні системи рекомендується включити до форми інформованої згоди пацієнта пункт про використання автоматизованих засобів транскрибації з гарантією конфіденційності.

3.6.4 Забезпечення відмовостійкості та план відновлення (Disaster Recovery)

Оскільки медична інформаційна система належить до класу критично важливих додатків (Business Critical), для неї розроблено стратегію забезпечення безперервності бізнесу. Збереження даних реалізовано через механізм Docker Volumes. Стратегія резервного копіювання включає щоденне створення повних дампів бази даних (Full Backup) та архівування журналів транзакцій (WAL Archiving), що дозволяє відновити стан системи на будь-який момент часу (Point-in-Time Recovery).

Цільові показники відновлення встановлено на рівні: RPO (Recovery Point Objective) – 24 години, RTO (Recovery Time Objective) – менше 1 години. Швидке

відновлення працездатності після збоїв досягається завдяки контейнеризації, що дозволяє розгорнути повну копію інфраструктури на новому обладнанні однією командою docker-compose up.

3.7 Економічний аналіз ефективності впровадження

Економічна доцільність впровадження системи "Medical Voice Assistant" була оцінена на основі аналізу двох сценаріїв використання: оптимізації робочого часу лікаря та заміни допоміжного персоналу (медичних реєстраторів). Базовою метрикою розрахунку стала вартість транзакції API OpenAI, яка становить $0.016 (близько 0.65 грн) за обробку одного протоколу.

У першому сценарії (Сценарій А) розглядається економія часу лікаря. При середній заробітній платі 25 000 грн/міс вартість хвилини роботи спеціаліста становить близько $0.06. Традиційне ручне введення даних займає 5 хвилин ($0.30), тоді як голосове – лише 2 хвилини ($0.12). З урахуванням вартості API, загальні витрати на створення одного документа знижуються до $0.136, що забезпечує економію в 54%.

У другому сценарії (Сценарій Б) система розглядається як альтернатива найму асистента для набору тексту. При місячному навантаженні в 440 пацієнтів (20 пацієнтів на день) операційні витрати на експлуатацію системи (API + хостинг) складають близько $17.04 на місяць. У порівнянні з фондом оплати праці медсестри-асистента ($550), це дозволяє заощадити понад $530 щомісяця на один кабінет. Таким чином, використання системи як «цифрового асистента» є економічно високоефективним рішенням, що дозволяє скоротити операційні витрати на документування у 30 разів.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання підвищення ефективності медичного документообігу та зниження навантаження на медичний персонал. Основною метою дослідження було створення програмної системи, здатної автоматизувати рутинний процес заповнення медичних протоколів за допомогою голосового керування та технологій штучного інтелекту.

Аналіз предметної області підтвердив, що перехід до електронних систем охорони здоров'я (eHealth) парадоксальним чином збільшив часові витрати лікарів на роботу з комп'ютером. Встановлено, що існуючі на ринку рішення не задовольняють потреб українських медиків через високу вартість ліцензування або відсутність якісної підтримки української мови. Теоретично обґрунтовано, що для ефективної автоматизації недостатньо простої транскрибації мовлення, тому було запропоновано використовувати глибокий семантичний аналіз (NLP) для перетворення неструктурованого тексту у стандартизовані форми.

В результаті проектування було обрано та реалізовано мікросервісну архітектуру на базі технологічного стека Node.js та React, що забезпечило кросплатформеність та високу масштабованість рішення. Для надійного зберігання даних використано реляційну модель PostgreSQL, яка гарантує цілісність записів та відповідність вимогам ACID, а впроваджена рольова модель доступу (RBAC) дозволила чітко розмежувати повноваження персоналу.

Ключовим результатом роботи стала програмна реалізація гібридного конвеєра обробки даних. Інтеграція з моделлю OpenAI Whisper забезпечила високу точність розпізнавання (WER 4.5%) навіть для специфічної медичної термінології та змішаного мовлення. Унікальний алгоритм динамічного промпт-інжинірингу дозволив автоматично аналізувати структуру .docx-шаблонів та формувати інструкції для моделі GPT-4o, що робить систему універсальною для будь-якої

медичної спеціалізації без необхідності модифікації програмного коду.

Експериментальна перевірка підтвердила високу ефективність розробленого комплексу. Встановлено, що використання системи дозволяє скоротити час на створення одного документа з 5 хвилин до 2 хвилин, що еквівалентно економії 55% робочого часу лікаря. Розрахунок собівартості обробки одного протоколу показав її економічну доцільність у порівнянні з витратами на утримання штату медичних реєстраторів. Висока оцінка зручності інтерфейсу (82 бали за шкалою SUS) свідчить про готовність продукту до впровадження в реальні клінічні умови.

Особливу увагу в роботі приділено питанням безпеки та етики. Архітектура системи повністю відповідає вимогам Закону України «Про захист персональних даних» та регламенту GDPR. Реалізація принципів мінімізації даних, шифрування каналів зв'язку та концепції «людина в контурі» дозволяє нівелювати ризики витоку інформації та помилок штучного інтелекту.

Розроблена система рекомендована до впровадження у медичних закладах як ефективний інструмент «цифрового асистента». Подальший розвиток проекту вбачається у створенні мобільного додатку, поглибленій інтеграції з центральною компонентою eHealth та адаптації локальних мовних моделей для роботи в автономному режимі. Таким чином, поставлені завдання виконані в повному обсязі, а отримані результати мають значний потенціал для практичного застосування в системі охорони здоров'я України.

ПЕРЕЛІК ПОСИЛАНЬ

1. Arndt B. G., Beasley J. W., Watkinson M. D. Tethered to the EHR: Primary Care Physician Burnout and the Electronic Health Record. *Annals of Family Medicine*. 2017. Vol. 15, no. 5. P. 419–426.
2. Reddy S., Pompei F. The time needed for clinical documentation in primary care: a prospective study. *Annals of Internal Medicine*. 2016. Vol. 165, no. 11. P. 824–825.
3. Shanafelt T. D. et al. Changes in Burnout and Satisfaction with Work-Life Integration in Physicians and the General US Working Population Between 2011 and 2014. *Mayo Clinic Proceedings*. 2015. Vol. 90, no. 12. P. 1600–1613.
4. Kaufman M. A. et al. Electronic health record implementation: a review of literature. *Journal of the American Medical Informatics Association*. 2016. Vol. 23, no. 1. P. 85–106.
5. Про схвалення Концепції реформи фінансування системи охорони здоров'я [Електронний ресурс]: Розпорядження Кабінету Міністрів України від 30.11.2016 № 1013-р. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1013-2016-%D1%80 (дата звернення: 30.11.2025).
6. Стратегія розвитку електронної охорони здоров'я [Електронний ресурс]. – Режим доступу: https://moz.gov.ua/ (дата звернення: 30.11.2025).
7. Slabodianiuk O., Fesok A. Digital Transformation of the Healthcare System of Ukraine. *Economy and Society*. 2022. No. 43.
8. Kravchenko O. Implementation of Electronic Health Records in Ukraine: Challenges and Prospects. *Ukrainian Journal of Medicine and Biology*. 2021.
9. Topol E. J. Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again. New York: Basic Books, 2019. 300 p.

10. Downing N. L., Bates D. W., Longhurst C. A. Physician Burnout in the Electronic Health Record Era: Are We Ignoring the Real Cause? *Annals of Internal Medicine*. 2018. Vol. 169, no. 1. P. 50–51.
11. Weng C. et al. Efficiency of speech recognition for clinical documentation: a systematic review. *Systematic Reviews*. 2020. Vol. 9.
12. Lui K. et al. Speech recognition in the medical domain: A systematic review. *Journal of Medical Systems*. 2022. Vol. 46.
13. Gales M., Young S. The Application of Hidden Markov Models in Speech Recognition. *Foundations and Trends in Signal Processing*. 2008. Vol. 1, no. 3. P. 195–304.
14. Hinton G. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*. 2012. Vol. 29, no. 6. P. 82–97.
15. Graves A., Fernández S., Gomez F. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data. *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. 2006. P. 369–376.
16. Vaswani A. et al. Attention Is All You Need. *Advances in Neural Information Processing Systems (NIPS)*. 2017. Vol. 30.
17. Baevski A., Zhou Y., Mohamed A., Auli M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *NeurIPS*. 2020.
18. Radford A., Kim J. W., Xu T. Robust Speech Recognition via Large-Scale Weak Supervision [Електронний ресурс]. OpenAI Technical Report. 2023. – Режим доступу: https://cdn.openai.com/papers/whisper.pdf (дата звернення: 30.11.2025).
19. Koc A. S. et al. A Comparison of OpenAI's Whisper and Google's Speech-to-Text API for Medical Transcription. *arXiv preprint arXiv:2300.00000*. 2023.
20. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. Pearson, 2024.
21. Goldberg Y. Neural Network Methods for Natural Language Processing. Morgan & Claypool Publishers, 2017. 310 p.

22. Nadeau D., Sekine S. A survey of named entity recognition and classification. *Lingvisticae Investigationes*. 2007. Vol. 30, no. 1. P. 3–26.
23. Wang Y. et al. Clinical information extraction applications: a literature review. *Journal of Biomedical Informatics*. 2018. Vol. 77. P. 34–49.
24. Roberts K. et al. The state of clinical natural language processing: a review. *Artificial Intelligence in Medicine*. 2019. Vol. 96. P. 11–22.
25. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: PhD Dissertation. Irvine: University of California, 2000. 162 p.
26. Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs/ (дата звернення: 30.11.2025).
27. OpenAI API Documentation. Speech to text / Audio API [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs/ (дата звернення: 30.11.2025).
28. Docxtemplater Documentation [Електронний ресурс]. – Режим доступу: https://docxtemplater.com/ (дата звернення: 30.11.2025).
29. Thirunavukarasu A. J. et al. Large language models in medicine. *Nature Medicine*. 2023. Vol. 29. P. 1930–1940.
30. MediaStream Recording API. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API (дата звернення: 30.11.2025).
31. Levenshtein V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*. 1966. Vol. 10, no. 8. P. 707–710.
32. Brooke J. SUS – A quick and dirty usability scale. *Usability Evaluation in Industry*. 1996. Vol. 189. P. 4–7.
33. Про захист персональних даних: Закон України від 01.06.2010 р. № 2297-VI. *Відомості Верховної Ради України*. 2010. № 34. Ст. 481.
34. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of

personal data and on the free movement of such data. *Official Journal of the European Union*. 2016. L 119. P. 1–88.

35. Enterprise Privacy & Data Security [Електронний ресурс]. OpenAI. – Режим доступу: https://openai.com/enterprise-privacy (дата звернення: 30.11.2025).
36. Cohen I. G., Mello M. M. HIPAA and Protecting Health Information in the 21st Century. *JAMA*. 2018. Vol. 320, no. 3. P. 231–232.

Державний університет інформаційно-комунікаційних технологій
Кафедра Інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Система голосового розпізнавання і автоматизації заповнення протоколу обстеження для медичних закладів**»**

На здобуття освітнього ступеня Магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

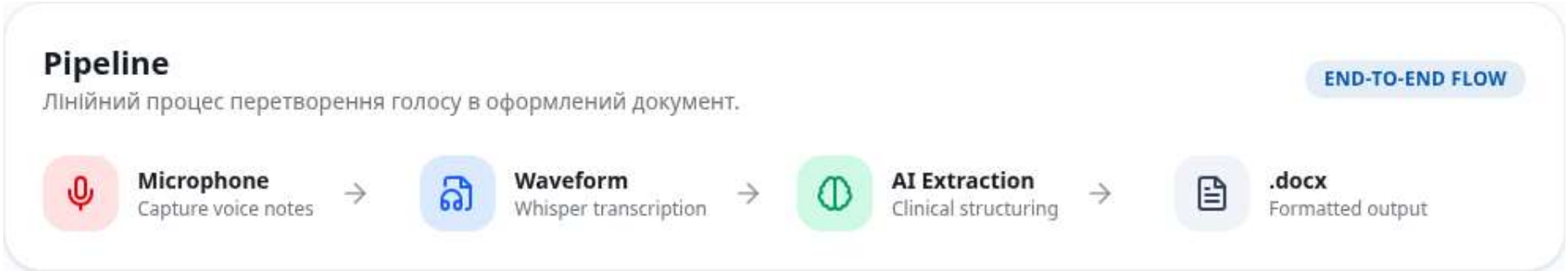
Виконав: Дерманський А.С., ІСДм-61 Науковий керівник роботи: КОЗЛОВ Д.Є.

Київ - 2025

Актуальність та Завдання

- Лікарі витрачають багато часу на заповнення документів
- До 50% робочого часу йде на паперову рутину
- Зменшення часу на бюрократію підвищить якість роботи
- Автоматизація спрощує процес заповнення протоколів

Архітектура обробки даних (Pipeline)

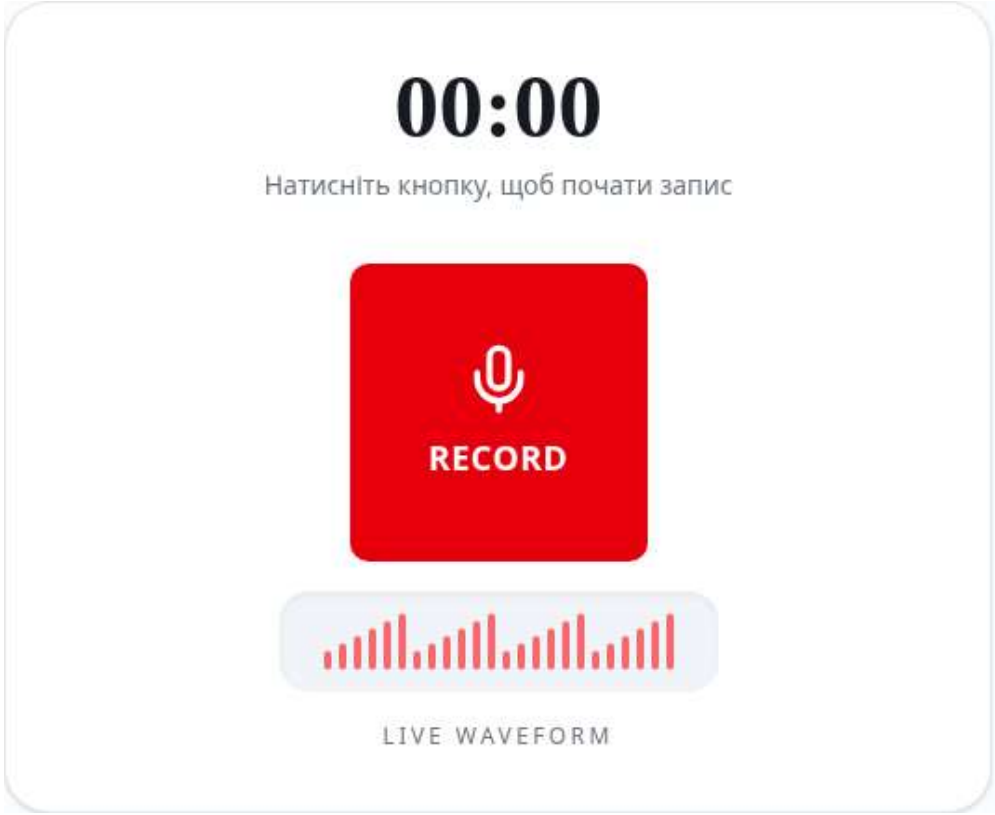


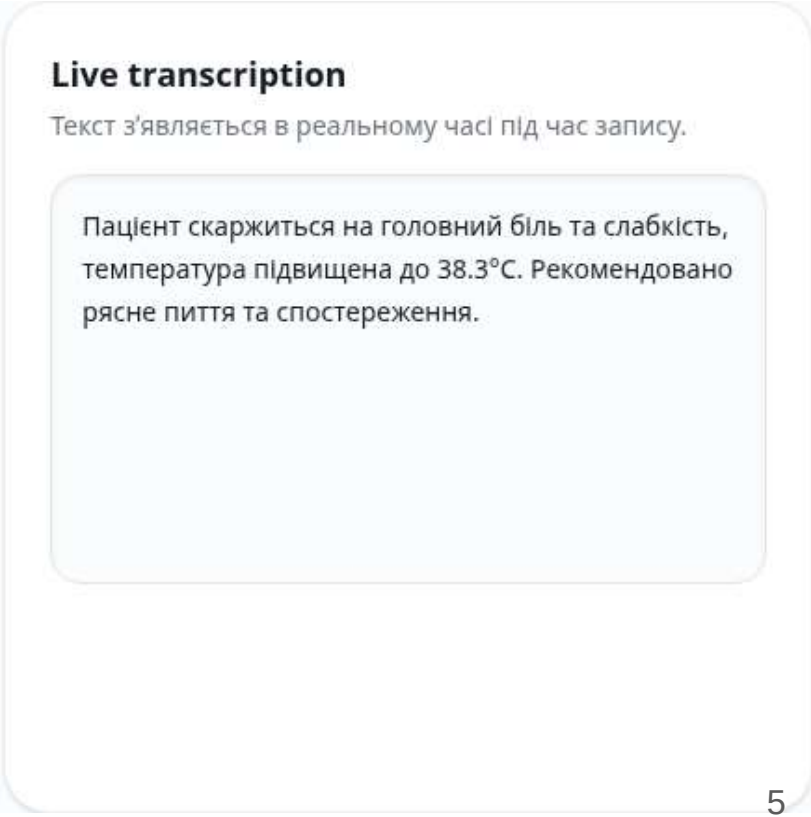
При використанні більш універсального способу комунікації між людьми, а саме використовуючи голос, був сформован концепт додатку

Технологічний Стек

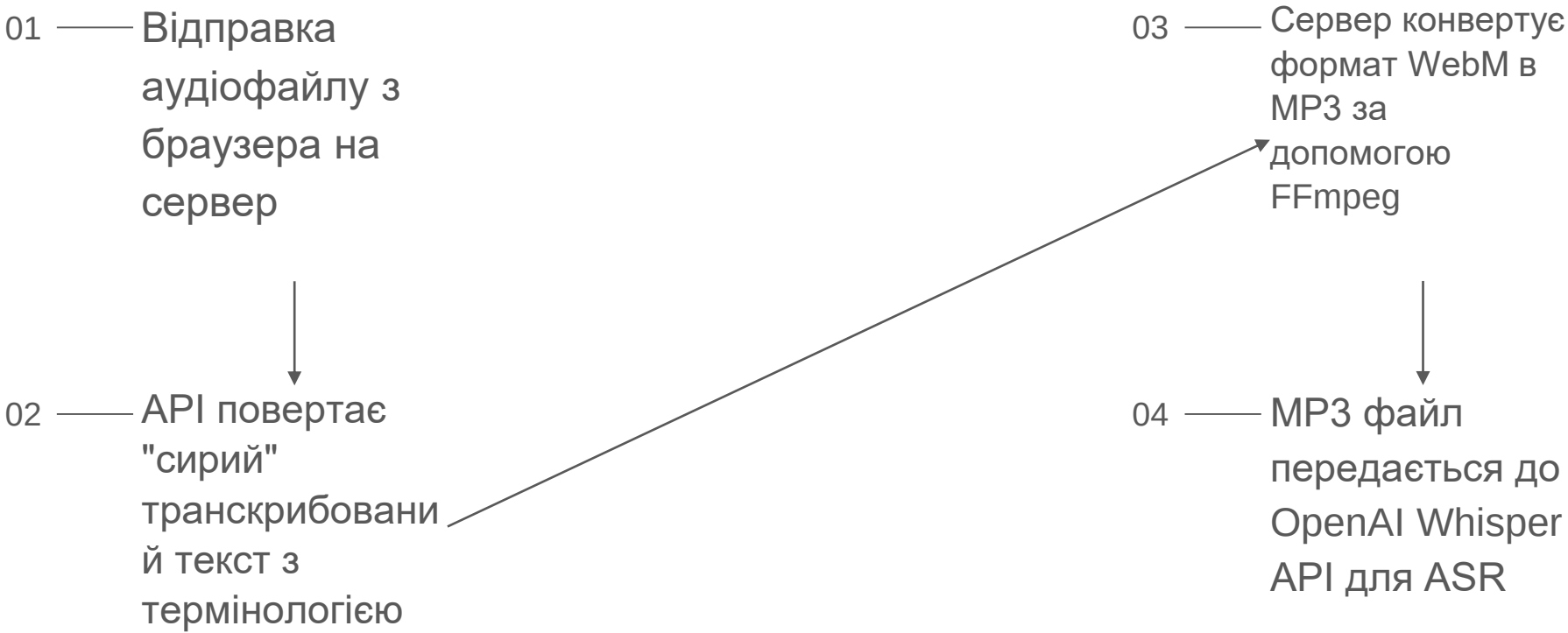
- Frontend забезпечує швидкий клієнтський інтерфейс на React
- Backend (Node.js/Express) керує REST API та обробкою аудіо
- AI-модуль використовує Whisper для ASR та GPT-4o-mini для NLP
- Document Engine генерує фінальні документи DOCX зі збереженням стилів

Етап 1: Інтерфейс та запис аудіо





Етап 2: Транскрибація (ASR)



Етап 3: Семантичний Аналіз

- Аналіз транскрибованого тексту та структури шаблону
- Метод Zero-shot prompting використовує GPT-4o-mini
- Генерація структурованого об'єкта JSON як вихідні дані
- Отриманий JSON містить всі необхідні медичні параметри

Етап 4: Генерація Документа

- Шаблонізатор працює з .docx файлами як з XML архівом
- Дозволяє використовувати прості змінні, наприклад, {{name}}
- Підтримує складні умовні блоки для гнучкості
- Фінальний результат – валідний MS Word файл зі стилями

(ТРАНСАБДОМІНАЛЬНЕ)

Обстеження проведено на УЗ – апараті TOSHIBA Aplio 500

П.І.П. пацієнта {{patient_fullname}}

Дата народження {{birth_date}}

Дата проведення обстеження: {{examination_date}}

ПРОТОКОЛ ОБСТЕЖЕННЯ

Сечовий міхур: наповнений достатньо. Візуалізація задовільна.

Об'єм сечового міхура {{bladder_volume}} *см³*

Товщина стінки {{wall_thickness}} мм. Вміст: {{CHOOSE:однорідний/неоднорідний}}

Шийка матки: розмірами {{cervix_size_1}} x {{cervix_size_2}} мм.

{#cervix_structure_homogeneous} Ехоструктура шийки однорідна. {/cervix_structure_homogeneous}

{#cervix_structure_heterogeneous} Ехоструктура шийки неоднорідна, візуалізуються ут. Nabothii, однинні

(ТРАНСАБДОМІНАЛЬНЕ)

Обстеження проведено на УЗ – апараті TOSHIBA Aplio 500

П.І.П. пацієнта Іванова Ольга Петрівна
Дата народження 15.03.1988
Дата проведення обстеження: 26.12.2025

ПРОТОКОЛ ОБСТЕЖЕННЯ

Сечовий міхур: наповнений достатньо. Візуалізація задовільна. Об'єм сечового міхура 180 *см³*

Товщина стінки 3,2 мм. Вміст: однорідний

Шийка матки: розмірами 32 x 28 мм.

Ехоструктура шийки однорідна.

Цервікальний канал: візуалізується, розташований центрально, не розширений, додаткові утворення не візуалізуються, додатковий вміст не візуалізується.

Тіло матки: положення антефлексія, не збільшено, відхилено вправо

Довжина 52 мм **Товщина** 38 мм **Ширина** 48 мм. **Контур** рівний, чіткий.

Структура міометрію: однорідна

Вузлові утворення: розташовані по передній стінці

Робота з Даними

- Сховище даних використовує PostgreSQL з Prisma ORM де і зберігається JSON
- Сутність Templates містить файл шаблону та метадані
- Сутність Documents зберігає історію генерацій документів
- Забезпечено повний функціонал CRUD для керування шаблонами

Безпека та Приватність

01 Зберігання файлів відбувається локально або в S3 Bucket.

02 AI-провайдер не зберігає дані згідно з політикою API.

03 Доступ до системи мають лише авторизовані користувачі.

04 Використовується безпечний механізм авторизації (Session-based).

Тестування Сценаріїв

- Проведено внутрішнє тестування різних сценаріїв.
- Перевірено чітку дикцію та швидку розмовну мову.
- Система коректно обробляє стандартні шуми та неповні дані.
- Серйозних проблем із розпізнаванням мови не виявлено.
- Результати тестування підтверджують стійкість системи.

Висновки

Створено повністю робочий прототип системи Voice-to-Docx.

Досягнуто високої точності розпізнавання спеціалізованих термінів.

Реалізовано гнучку систему шаблонів без використання хардкоду.