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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення для обробки звернень пацієнтів стоматологічної клініки з використанням LLM

Виконала: студентка IV курсу, групи СП-41

спеціальності 121 Інженерія прогламного забезпечення
(шифр і назва спеціальності)

_____ Гнецько В.М.
(підпис) (прізвище та ініціали)

Керівник _____ Михалик Д.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

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

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

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
«__» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 121 Інженерія прогламного забезпечення
(шифр і назва спеціальності)
Студентці Гнецько Вікторія Михайлівна
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка програмного забезпечення для обробки звернень пацієнтів стоматологічної клініки з використанням LLM

Керівник роботи Михалик Дмитро Михайлович, к.т.н., доцент кафедри ПІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «__» _____ 2026 року № ___
2. Термін подання студентом завершеної роботи 2026 р.
3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз вимог до системи. 2 Проєктування та розробка програмної системи. 3. Тестування, впровадження та підтримка 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Слайди презентації

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	13.04.2026	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір	13.04.2026-20.04.2026	*Виконано*
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	21.04.2026-03.05.2026	*Виконано*
	роботи		
4.	Оформлення розділу «Аналіз вимог до системи»	04.05.2026-10.05.2026	*Виконано*
5.	Оформлення розділу «Проєктування та розробка	04.05.2026-10.05.2026	*Виконано*
	програмної системи»		
6.	Оформлення розділу «Тестування, впровадження та	04.05.2026-10.05.2026	*Виконано*
	підтримка»		
7.	Виконання завдання до підрозділу «Охорона праці»	27.04.2026-10.05.2026	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	27.04.2026-10.05.2026	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	11.05.2026-13.05.2026	*Виконано*
10.	Нормоконтроль		*Виконано*
11.	Перевірка на плагіат		*Виконано*
12.	Попередній захист кваліфікаційної роботи		*Виконано*
13.	Захист кваліфікаційної роботи		

Студентка _____ Гнецько В.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Михалик Д.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення централізованого управління комунікаціями стоматологічної клініки з використанням LLM // Кваліфікаційна робота освітнього рівня «Бакалавр» // Гнецько Вікторія Михайлівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-41 // Тернопіль, 2026 // С. ___, рис. – ___, табл. – ___, додат. – ___, бібліогр. – ___.

Ключові слова: стоматологічна клініка; LLM; штучний інтелект; автоматизація комунікацій; Node.js; Python; PostgreSQL; Groq API; LLaMA 3.

Кваліфікаційна робота присвячена розробці програмного забезпечення централізованого управління комунікаціями стоматологічної клініки з використанням великої мовної моделі. Проаналізовано предметну область стоматологічних послуг, сучасні CRM-системи та месенджери. Сформовано функціональні й нефункціональні вимоги до системи. Спроєктовано архітектуру програмного забезпечення, структуру бази даних, REST API та механізм інтеграції великої мовної моделі. Реалізовано серверну частину на Node.js та Express.js, модуль обробки запитів на Python і інтеграцію з моделлю LLaMA 3 через Groq API. Для зберігання даних використано PostgreSQL.

Об'єкт дослідження – процес комунікації між пацієнтами та стоматологічною клінікою під час надання інформаційних послуг і запису на прийом.

Предмет дослідження – методи, моделі та програмні засоби автоматизації комунікацій із використанням великих мовних моделей та вебтехнологій.

Практичним результатом роботи є програмна система для автоматизованої обробки звернень пацієнтів, генерації відповідей і підтримки запису на прийом. Вона забезпечує централізоване управління комунікаціями стоматологічної клініки.

ABSTRACT

Development of Software for Centralized Communication Management in a Dental Clinic Using LLM // Bachelor's Qualification Thesis // Viktoriia Hnetsko // Ternopil Ivan Puluj national technical university, faculty of computer information systems and software engineering, department of software engineering, group SP-41 // Ternopil, 2026 // P. ___, fig. – ___, tabl. – ___, annexes – ___, references – ___.

Keywords: dental clinic; LLM; artificial intelligence; communication automation; Node.js; Python; PostgreSQL; Groq API; LLaMA 3.

The qualification thesis is devoted to the development of software for centralized communication management in a dental clinic using a large language model.

The problem domain of dental services, modern CRM systems, and messaging platforms was analyzed. Functional and non-functional requirements for the system were defined.

The software architecture, database structure, REST API, and integration mechanism of the large language model were designed.

The server-side application was implemented using Node.js and Express.js. A Python-based request processing module and integration with the LLaMA 3 model through the Groq API were developed. PostgreSQL was used for data storage.

The object of research is the communication process between patients and a dental clinic during information service provision and appointment scheduling.

The subject of research is methods, models, and software tools for communication automation using large language models and web technologies.

The practical result of the thesis is a software system for automated processing of patient requests, response generation, and appointment scheduling support. The system provides centralized management of communications within a dental clinic.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

LLM – Large Language Model (велика мовна модель)

AI – Artificial Intelligence (штучний інтелект)

API – Application Programming Interface (інтерфейс прикладного програмування)

REST – Representational State Transfer (архітектурний стиль взаємодії вебсервісів)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

JSON – JavaScript Object Notation (формат обміну даними)

CRM – Customer Relationship Management (система управління взаємовідносинами з клієнтами)

UI – User Interface (інтерфейс користувача)

UX – User Experience (користувацький досвід)

UML – Unified Modeling Language (уніфікована мова моделювання)

ERD – Entity Relationship Diagram (діаграма «сутність–зв’язок»)

DB – Database (база даних)

SQL – Structured Query Language (мова структурованих запитів)

JWT – JSON Web Token (маркер автентифікації користувача)

SDK – Software Development Kit (набір засобів для розробки програмного забезпечення)

QA – Quality Assurance (забезпечення якості програмного забезпечення)

RAM – Random Access Memory (оперативна пам’ять)

CPU – Central Processing Unit (центральний процесор)

URL – Uniform Resource Locator (уніфікований покажчик ресурсу)

LLM API – програмний інтерфейс доступу до великої мовної моделі

Groq API – програмний інтерфейс платформи Groq для доступу до мовних моделей

ВСТУП

Актуальність цієї роботи обумовлена зростаючою потребою в покращенні комунікаційних процесів у стоматологічних клініках. У більшості випадків запити пацієнтів все ще обробляються адміністраторами вручну, що призводить до затримок, пропущених повідомлень та непослідовних відповідей. Зі збільшенням кількості пацієнтів цей підхід стає менш ефективним та складнішим у підтримці. Існує явна потреба в автоматизації комунікаційних процесів, яка може забезпечити швидку, послідовну та масштабовану взаємодію з пацієнтами. У цьому контексті сучасні технології, зокрема моделі великих мов (LLM), відіграють важливу роль, оскільки вони дозволяють системам розуміти та генерувати текст, подібний до людського, у режимі реального часу.

Проблема стає більш очевидною при аналізі поточних комунікаційних робочих процесів. Пацієнти часто стикаються з повільним часом реагування, особливо в години пік, що негативно впливає на взаємодію з користувачем. Водночас адміністратори перевантажені повторюваними завданнями, такими як відповіді на поширені запитання, планування зустрічей та надання базових консультацій. Традиційні рішення, включаючи CRM-системи та чат-боти на основі правил, обмежені в гнучкості та вимагають попередньо визначених скриптів, що робить їх нездатними обробляти складніші або нестандартні запити.

Використання LLM забезпечує більш просунутий підхід до вирішення цих проблем. Такі моделі здатні розуміти природну мову, визначати наміри користувача та генерувати контекстно-залежні відповіді без використання жорстких правил. На відміну від традиційних систем на основі правил, рішення на основі LLM можуть адаптуватися до різних типів запитів та підтримувати більш природний потік комунікації, що робить їх більш придатними для реальних застосувань у системах зв'язку в охороні здоров'я.

Метою цієї кваліфікаційної роботи є розробка програмної системи для централізованого управління комунікацією в стоматологічній клініці з використанням LLM.

Для досягнення цієї мети були визначені такі завдання: аналіз предметної області, проектування архітектури системи, реалізація бекенду за допомогою Node.js, інтеграція модуля LLM за допомогою Python, встановлення взаємодії між компонентами системи на основі API, проектування структури бази даних, а також проведення тестування та оцінки розробленої системи.

Об'єктом дослідження є процес комунікації між пацієнтами та стоматологічною клінікою, включаючи обмін повідомленнями, пов'язаними з консультаціями, запитами на обслуговування та записом на прийом.

Предметом дослідження є методи та інструменти автоматизації комунікації з використанням моделей великих мов, зосереджуючись на інтеграції LLM у програмну систему для обробки запитів пацієнтів.

Методи дослідження включають аналіз предметної області для визначення ключових вимог та проблем, проектування програмного забезпечення для побудови архітектури системи, експериментальне тестування для оцінки поведінки системи в різних сценаріях та аналіз якості відповіді з точки зору точності та релевантності.

Практична цінність роботи полягає в можливості застосування розробленої системи в стоматологічних клініках для автоматизації процесів комунікації, зменшення ручного навантаження та підвищення загальної ефективності взаємодії з пацієнтами.

Структура роботи складається з трьох основних розділів: перший розділ зосереджений на аналізі вимог та предметної області, другий описує проектування та впровадження системи, а третій представляє результати тестування, розгортання та оцінки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Предметом цього дослідження є організація та управління комунікаційними процесами в стоматологічній клініці. Сучасні стоматологічні клініки функціонують як складні сервісно-орієнтовані системи, що передбачають постійну взаємодію між пацієнтами, медичним персоналом та адміністративним персоналом. Ефективна комунікація відіграє вирішальну роль у забезпеченні високоякісного обслуговування, задоволеності пацієнтів та ефективного управління робочим процесом [1].

1.1 Аналіз предметної області стоматологічної клініки

Типова стоматологічна клініка включає кількох ключових учасників: пацієнтів, стоматологів, адміністраторів та допоміжний персонал. Пацієнти взаємодіють з клінікою через різні канали, такі як телефонні дзвінки, платформи обміну повідомленнями, веб-сайти та мобільні додатки. Ці взаємодії включають запис на прийом, запити на консультації, запити щодо лікування, нагадування та спілкування після лікування. Адміністративний персонал відповідає за обробку цих запитів, планування прийомів та координацію між пацієнтами та лікарями.

Однією з основних проблем у цій галузі є великий обсяг повторюваних комунікаційних завдань. До них належать відповіді на поширені запитання, підтвердження прийомів, надсилання нагадувань та надання базових консультаційних відповідей. Ручне виконання таких завдань збільшує навантаження на персонал, призводить до затримок та може знизити загальну ефективність клініки [2].

Крім того, комунікація в стоматологічних клініках повинна враховувати конкретні обмеження, такі як конфіденційність даних, точність медичної інформації та своєчасне реагування. Затримки або невірна інформація можуть негативно вплинути на довіру пацієнтів та результати лікування. Тому

автоматизація та інтелектуальні системи підтримки стають дедалі актуальнішими в цій галузі (рис. 1.1).

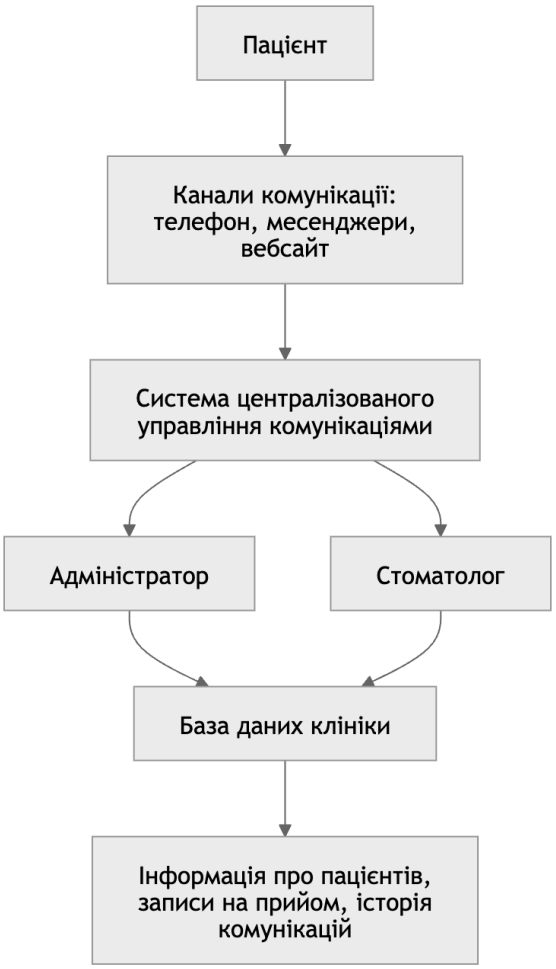

Рисунок 1.1 – Загальна структура комунікаційних процесів у стоматологічній клініці

Ще одним важливим аспектом є інтеграція комунікаційних процесів із внутрішніми системами клініки, такими як записи пацієнтів, системи планування та управління лікуванням. Централізована система зв'язку дозволяє зберігати, відстежувати та аналізувати всі взаємодії, що покращує прийняття рішень та якість обслуговування [3].

Нещодавні досягнення в галузі штучного інтелекту, зокрема моделі великих мов (LLM), надають нові можливості для автоматизації комунікаційних процесів. Системи на основі LLM можуть генерувати відповіді природною мовою,

обробляти запити пацієнтів та допомагати в управлінні комунікаційними робочими процесами. Це дозволяє частково автоматизувати рутинні завдання, зберігаючи при цьому досвід взаємодії, подібний до людського.

Таким чином, предметна область характеризується:

- високою інтенсивністю комунікаційних процесів;
- необхідністю взаємодії в режимі реального часу;
- повторюваними та структурованими типами запитів;
- суворими вимогами до точності та надійності;
- необхідністю інтеграції з існуючими інформаційними системами.

Аналіз цієї області показує, що існує явна потреба в розробці централізованої програмної системи, здатної ефективно керувати комунікаційними процесами, зменшувати навантаження на персонал та покращувати досвід пацієнтів завдяки використанню інтелектуальних технологій.

1.2 Аналіз існуючих програмних рішень для управління комунікаціями

Сучасний ландшафт програмних рішень для управління комунікаціями в охороні здоров'я, і зокрема в стоматологічних клініках, досить різноманітний. На перший погляд може здатися, що проблема вже вирішена, оскільки доступно багато інструментів. Однак, глибший погляд показує, що більшість цих систем вирішують лише конкретні аспекти комунікації, а не забезпечують повне, інтегроване рішення.

Значна частина клінік покладається на універсальні CRM-системи, такі як HubSpot CRM або Zoho CRM. Ці платформи розроблені для організації взаємодії з клієнтами, зберігання історії комунікації та автоматизації певних процесів, таких як відповіді на електронні листи або нагадування. Вони гнучкі та відносно легко адаптуються, що пояснює їхню популярність. Водночас вони не побудовані з урахуванням медичних робочих процесів. В результаті клінікам часто доводиться

адаптувати свої процеси до системи, а не до того, щоб система адаптувалася до клініки [4, 5].

З іншого боку, існують спеціалізовані системи управління стоматологією, такі як Dentrix та Open Dental. Ці рішення набагато ближчі до реальних потреб стоматологічних клінік. Зазвичай вони включають записи пацієнтів, планування прийомів, виставлення рахунків, а іноді й базові функції комунікації, такі як SMS-нагадування. Однак, коли справа доходить до фактичної взаємодії з пацієнтами, їхні можливості залишаються досить обмеженими. Більшість відповідей базуються на шаблонах, а будь-який нестандартний запит вимагає ручної обробки.

Існує також дуже поширена практика, яка не завжди формально вважається частиною системи — використання месенджерів. Такі інструменти, як WhatsApp Business або Telegram, широко використовуються для прямого спілкування з пацієнтами. Вони забезпечують швидку та зручну взаємодію, якої пацієнти очікують сьогодні. Однак такий підхід створює фрагментацію. Повідомлення розкидані по платформах, немає централізованого сховища, і персоналу доводиться вручну відстежувати та відповідати на запити.

Нещодавно з'явився новий напрямок — використання систем на основі штучного інтелекту, особливо великих мовних моделей, таких як ChatGPT. Ці системи здатні розуміти природну мову, генерувати відповіді та навіть обробляти відносно складні розмови. Порівняно з традиційними рішеннями, вони значно зменшують потребу в ручній роботі. Водночас вони рідко використовуються як окремі продукти в реальних клініках, здебільшого тому, що вони не пов'язані безпосередньо з внутрішніми системами, такими як система планування або бази даних пацієнтів.

Якщо ми відступимо назад і подивимося на всі ці рішення разом, то стане видною певна закономірність. Кожна категорія досить добре вирішує одну частину проблеми, але жодна з них не охоплює весь робочий процес комунікації від початку до кінця. CRM-системи керують даними, стоматологічні системи

керують процесами, месенджери обробляють комунікацію, а LLM надають аналітичні дані — але вони не уніфіковані.

Щоб зробити це порівняння зрозумілішим, основні характеристики аналізованих систем наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих рішень для управління комунікаціями

Система	**Тип**	**Канали комунікації**	**Інтеграція з клінічними системами**	**Обмеження**
HubSpot CRM	CRM-система	Електронна пошта, чат, форми	Обмежена	Не орієнтована безпосередньо на медичну сферу
Zoho CRM	CRM-система	Електронна пошта, чат, телефон	Обмежена	Потребує додаткового налаштування
Dentrix	Стоматологічна інформаційна система	SMS, внутрішні засоби комунікації	Висока	Обмежені можливості комунікації з пацієнтами
Open Dental	Стоматологічна інформаційна система	SMS, електронна пошта	Висока	Використання шаблонних відповідей
WhatsApp Business	Месенджер	Чат	Відсутня	Відсутність централізованого управління
Telegram	Месенджер	Чат	Відсутня	Переважно ручна обробка повідомлень

Порівняння підкреслює важливий момент. Існуючі системи зосереджені або на управлінні, або на комунікації, але рідко поєднують обидва інтелектуальним способом. На практиці це змушує клініки використовувати кілька інструментів одночасно, що ускладнює робочі процеси та збільшує ризик помилок.

Через це ідея централізованої системи стає не просто актуальною, а необхідною. Така система повинна поєднувати структуроване управління даними, комунікацію в режимі реального часу та інтелектуальні процеси.

1.3 Особливості використання моделей великих мов програмування в медичних інформаційних системах

В останні роки використання моделей великих мов програмування (LLM) у медичних інформаційних системах значно зросло. Ця тенденція зумовлена зростаючою потребою в автоматизації, покращенні взаємодії з пацієнтами та ефективнішій обробці великих обсягів текстових даних. LLM, засновані на архітектурах глибокого навчання, таких як трансформатори, здатні розуміти та генерувати природну мову з високим рівнем точності (рис. 1.2).

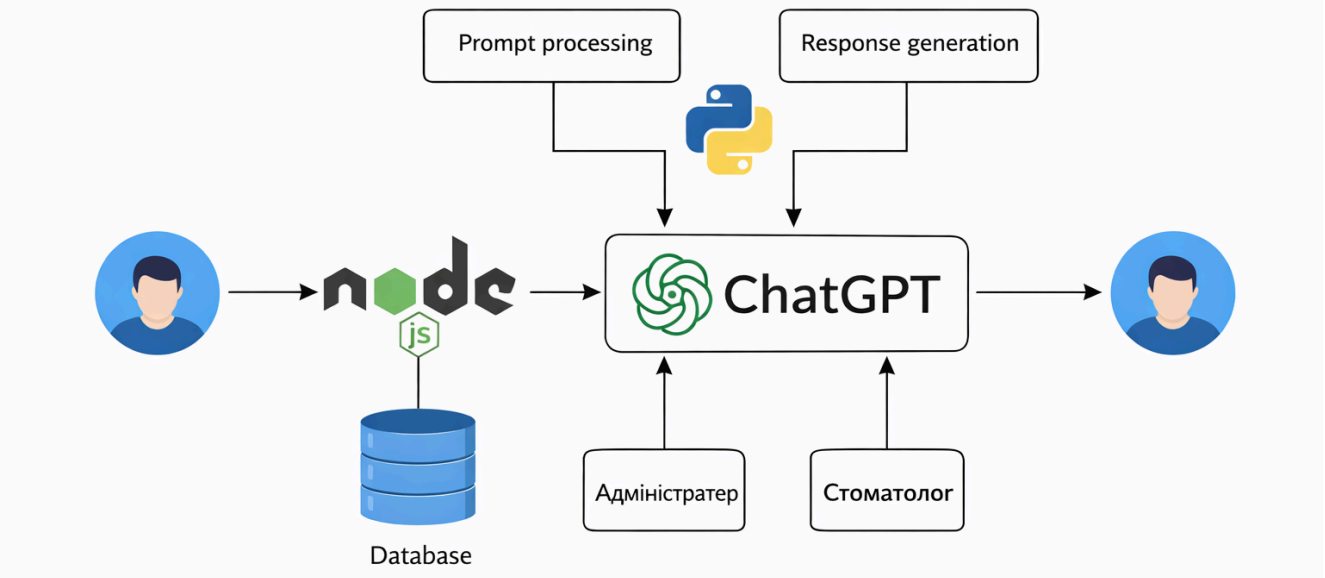

Рисунок 1.2 – Концептуальна схема обробки запитів пацієнтів із використанням LLM

Однією з ключових переваг LLM є їхня здатність обробляти неструктуровані дані. У системах охорони здоров'я до 80% клінічних даних є неструктурованими, включаючи повідомлення пацієнтів, медичні записки та записи консультацій. Традиційні системи мають труднощі з ефективною обробкою таких даних, тоді як LLM можуть витягувати значення, класифікувати запити та генерувати відповіді в режимі реального часу [6].

Ще одним важливим аспектом є автоматизація спілкування з пацієнтами. Дослідження показують, що 60–70% запитів пацієнтів є повторюваними,

включаючи запис на прийом, години роботи клініки, вартість лікування та основні консультації. Інтегруючи LLM у системи зв'язку, можна автоматизувати значну частину цих взаємодій, зменшуючи навантаження на адміністративний персонал приблизно на 30–50%.

Якість відповідей сучасних LLM також досягла рівня, придатного для практичного використання. Наприклад, у контрольованих оцінках системи на основі LLM демонструють точність 85–92% під час відповіді на загальні медичні запитання, за умови, що відповіді обмежені недіагностичною інформацією. Однак точність може знижуватися при роботі з вузькоспеціалізованими або неоднозначними запитами, що вимагає додаткових механізмів перевірки.

З точки зору продуктивності, LLM забезпечують взаємодію майже в режимі реального часу. Середній час генерації відповідей для сучасних моделей на основі API зазвичай коливається від 0,5 до 2 секунд, залежно від розміру моделі та інфраструктури. Цього достатньо для підтримки природного розмовного досвіду для користувачів [7].

Інтеграція є ще одним критичним фактором. LLM рідко використовуються як автономні системи. Натомість вони інтегруються в існуюче медичне програмне забезпечення через API. Це дозволяє їм отримувати доступ до структурованих даних, таких як записи пацієнтів, графіки прийомів та інформація про послуги. Водночас це створює проблеми, пов'язані з конфіденційністю та безпекою даних. Медичні системи повинні відповідати суворим правилам (таким як GDPR або HIPAA), що означає, що конфіденційні дані повинні бути належним чином анонімізовані або захищені під час обробки.

Також є обмеження, які необхідно враховувати. LLM можуть генерувати неправильну або оманливу інформацію, явище, відоме як галюцинації. Дослідження показують, що рівень галюцинацій у сценаріях відкритого домену може досягати 10–20%, що є неприйнятним у критичних медичних контекстах. Тому LLM у системах охорони здоров'я зазвичай обмежуються адміністративними та інформаційними завданнями, а не прийняттям клінічних рішень.

Ще однією проблемою є обчислювальні витрати. Запуск великих моделей вимагає значних ресурсів. Наприклад, витрати на виведення для хмарних API LLM можуть коливатися від $0,002 до $0,02 за запит, залежно від моделі та використання токенів. Для систем, які обробляють тисячі щоденних взаємодій, це стає важливим фактором у проектуванні системи [8].

Незважаючи на ці проблеми, переваги використання LLM є суттєвими. Вони покращують ефективність комунікації, зменшують експлуатаційні витрати та покращують взаємодію з користувачем. У контексті стоматологічних клінік LLM особливо корисні для обробки запитів пацієнтів, управління комунікацією, пов'язаною з прийомом пацієнтів, та надання стандартизованої інформації про послуги.

Таким чином, інтеграція LLM у медичні інформаційні системи є перспективним напрямком за умови впровадження відповідних обмежень, механізмів валідації та системних архітектур для забезпечення надійності та безпеки.

1.4 Постановка проблеми та цілі розробки системи

Проблема, що розглядається в цій роботі, полягає в відсутності централізованої та інтелектуальної системи управління комунікаційними процесами в стоматологічних клініках. Існуючі рішення або не забезпечують достатньої автоматизації, або не інтегровані з клінічними робочими процесами, що призводить до неефективної обробки запитів пацієнтів та збільшення навантаження на персонал [9].

Основною метою цієї роботи є розробка програмної системи для централізованого управління комунікаціями в стоматологічній клініці з використанням моделей великих мов програмування (LLM). Система повинна автоматизувати взаємодію з пацієнтами, інтегрувати кілька каналів зв'язку та забезпечити ефективну обробку запитів.

Для досягнення цієї мети визначено такі завдання:

- аналіз предметної області та існуючих рішень;
- проектування архітектури системи з інтеграцією LLM;
- розробка backend API за допомогою Node.js;
- реалізація модуля обробки на основі LLM за допомогою Python;
- забезпечення інтеграції між компонентами системи;
- проектування та реалізація структури бази даних;
- розробка інтерфейсу користувача;
- проведення тестування та оцінки системи.

Розроблена система має покращити ефективність комунікації, зменшити ручне робоче навантаження та підвищити загальну якість обслуговування пацієнтів.

1.5 Визначення зацікавлених сторін

Зацікавлені сторони – це особи або групи, які взаємодіють із системою або на які впливає її робота. У контексті системи управління комунікаціями стоматологічної клініки зацікавлені сторони визначаються на основі їхньої ролі в процесах комунікації та використанні системи.

Ключові зацікавлені сторони представлені в таблиці 1.2.

Таблиця 1.2 – Зацікавлені сторони системи

Зацікавлена сторона	**Роль у системі**	**Потреби**	**Рівень взаємодії**
1	**2**	**3**	**4**
Пацієнти	Ініціюють комунікацію та подають запити на отримання послуг	Швидкі відповіді, запис на прийом, зрозуміла інформація	Високий
Адміністратори	Керують комунікацією та розкладом прийому	Ефективна обробка запитів, зменшення навантаження	Високий
Стоматологи	Надають медичні послуги та отримують інформацію про пацієнтів	Точна та своєчасна інформація	Середній

Продовження таблиці 1.2

1	2	3	4
Керівництво клініки	Контролює роботу клініки та ефективність процесів	Звітність, контроль системи, підвищення ефективності роботи	Середній
Розробники системи	Розробляють та супроводжують систему	Стабільність системи, масштабованість, можливість інтеграції	Низький
Фахівці технічної підтримки	Підтримують працездатність інфраструктури	Надійність, безпека та безперервність роботи системи	Низький

Визначені зацікавлені сторони визначають функціональні та нефункціональні вимоги до системи та впливають на її архітектуру та реалізацію.

1.6 Ідентифікація системних акторів

Системні актори – це сутності, які безпосередньо взаємодіють із системою для досягнення певних цілей. Вони представляють зовнішніх користувачів або системи, які ініціюють дії або отримують результати від системи.

У розробленій системі управління комунікаціями для стоматологічної клініки визначено таких акторів:

- Пацієнт – ініціює спілкування із системою, подає запити, записується на прийом та отримує відповіді;
- Адміністратор – обробляє запити, керує прийомами та контролює робочі процеси комунікації;
- Стоматолог – отримує інформацію, пов’язану з пацієнтом, та взаємодіє із системою для планування та оновлень;
- Зовнішні служби обміну повідомленнями – надають канали зв’язку (месенджери, веб-форми тощо), через які запити надходять до системи;
- Модуль LLM (сервіс Python) – обробляє запити користувачів та генерує відповіді, використовуючи розуміння природної мови.

Ці актори взаємодіють із системою на різних рівнях та визначають основні сценарії взаємодії. Їх ідентифікація є важливою для побудови діаграм варіантів використання та проектування функціональності системи.

1.7 Діаграма варіантів використання

Діаграма варіантів використання представляє функціональну взаємодію між учасниками системи та основними службами, які надає програмна система. Він візуалізує, як різні користувачі та зовнішні компоненти взаємодіють із системою, і визначає межі її функціональності [10].

У розробленій системі централізованого управління комунікаціями в стоматологічній клініці визначено п’ять основних акторів: Пацієнт, Адміністратор, Стоматолог, Зовнішні служби обміну повідомленнями та Модуль LLM (Python). Кожен із цих учасників взаємодіє з системою відповідно до своєї ролі та обов’язків (рис. 1.3).

Актор Пацієнт ініціює зв'язок із системою. Основні випадки використання, пов’язані з цим актором, включають Надсилання запиту, Отримання відповіді, Запис на прийом і Скасування / перенесення. Ці випадки використання відображають типові взаємодії, коли пацієнт запитує інформацію, отримує відповіді та керує призначенням.

Операційне керівництво відповідає актор Адміністратор. Цей актор взаємодіє з такими варіантами використання, як Обробка звернення, Керування розкладом і Перегляд даних пацієнта. Адміністратор забезпечує належну обробку вхідних запитів і чітке дотримання розкладу.

Актор Стоматолог взаємодіє із системою за допомогою таких варіантів використання, як Перегляд даних пацієнта та Оновлення статусу прийому. Ці взаємодії пов’язані з клінічним процесом і відстеженням лікування пацієнтів.

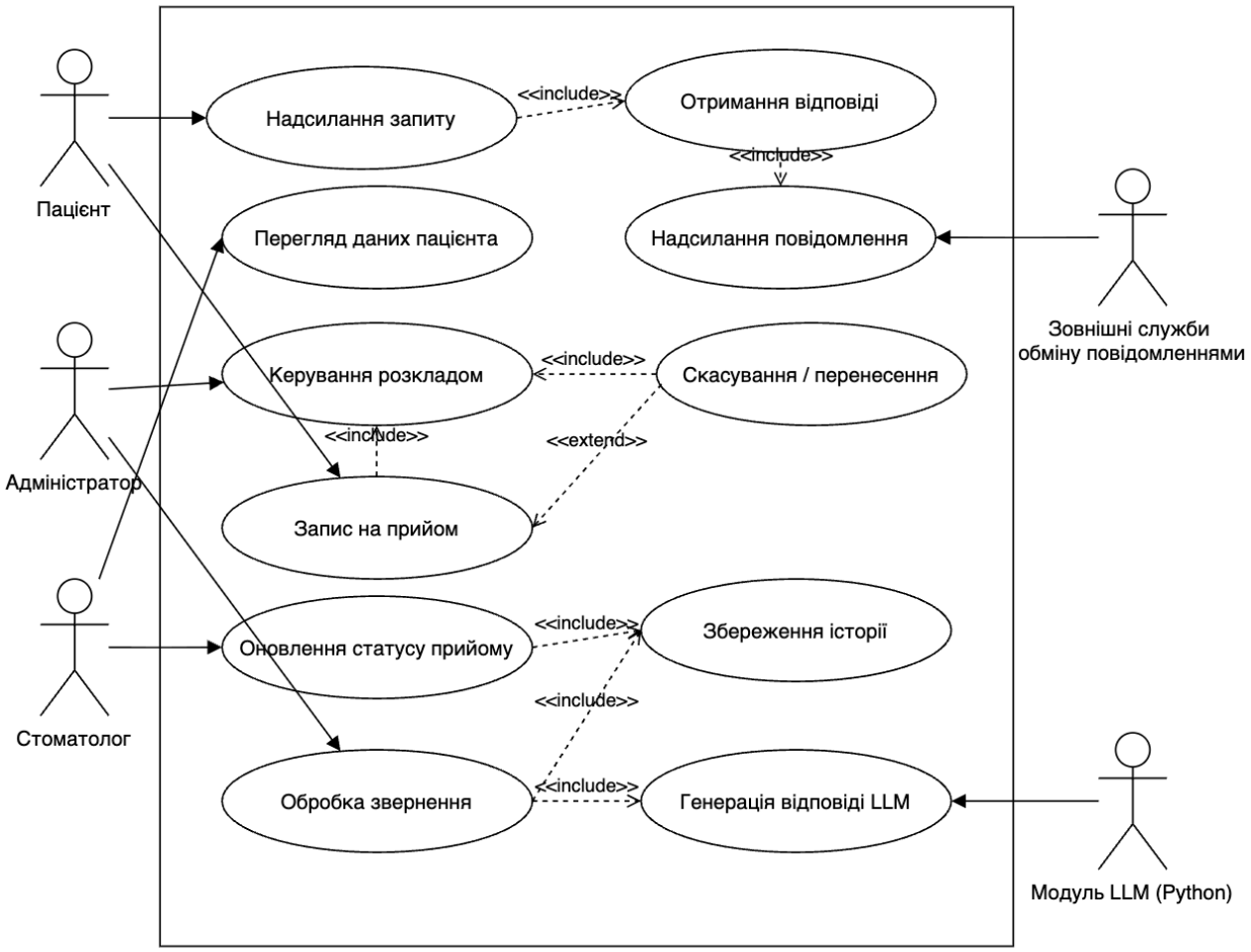

Рисунок 1.3 – Діаграма варіантів використання

Актор Зовнішні служби обміну повідомленнями відповідає за доставку повідомлень між системою та користувачами. Він безпосередньо пов’язаний із варіантом використання Надсилання повідомлення, який забезпечує зв’язок через зовнішні платформи, такі як месенджери чи веб-інтерфейси.

За інтелектуальну обробку запитів відповідає актор Модуль LLM (Python). Він взаємодіє з варіантом використання Генерація відповіді LLM, що дозволяє автоматично генерувати відповіді на основі введення природною мовою.

Зв’язки між варіантами використання визначаються за допомогою залежностей <<include>> і <<extend>>. Варіант використання Надсилання запиту включає Отримання відповіді, вказуючи, що подання запиту безпосередньо пов’язане з обробкою відповіді. Варіант використання Отримання відповіді

включає Надсилання повідомлення, оскільки відповідь має бути доставлена користувачеві.

Варіант використання Обробка звернення включає як Генерацію відповіді LLM, так і Збереження історії, вказуючи на те, що кожен оброблений запит аналізується модулем LLM і зберігається для використання в майбутньому. Аналогічно, Оновлення статусу прийому включає Збереження історії, гарантуючи, що всі зміни записуються.

Варіант використання Керування розкладом включає Запис на прийом, оскільки планування записів є основною частиною керування розкладом клініки. Крім того, Скасування / перенесення розширює Запис на прийом, оскільки представляє необов'язкову модифікацію наявної зустрічі.

Таким чином, діаграма варіантів використання чітко визначає функціональні можливості системи, обов'язки учасників і зв'язки між різними процесами. Він надає структуроване уявлення про те, як реалізуються комунікаційні робочі процеси, і підкреслює роль інтеграції LLM в автоматизації формування відповідей у системі.

1.8 Аналіз функціональних та нефункціональних вимог (FURPS+)

Аналіз системних вимог проводиться за допомогою моделі FURPS+, яка дозволяє структурувати вимоги на функціональні та нефункціональні категорії. Такий підхід забезпечує чітке розуміння поведінки системи, очікуваної продуктивності та операційних обмежень [11, 12].

Функціональні вимоги визначають, що повинна робити система. У розробленій системі основна функціональність зосереджена на управлінні комунікаційними процесами в стоматологічній клініці. Система повинна підтримувати обробку запитів користувачів, генерування відповідей, управління записами на прийом та зберігання комунікаційних даних. Основні функціональні вимоги представлені в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги

ID	**Вимога**
FR1	Система повинна дозволяти Пацієнт надсилати запити
FR2	Система повинна надавати відповіді на запити користувачів
FR3	Система повинна підтримувати запис на прийом
FR4	Система повинна дозволяти скасування або перенесення записів
FR5	Система повинна дозволяти Адміністратору керувати запитами
FR6	Система повинна підтримувати управління розкладом
FR7	Система повинна дозволяти переглядати дані пацієнтів
FR8	Система повинна оновлювати статус записів
FR9	Система повинна генерувати відповіді за допомогою LLM
FR10	Система повинна зберігати історію зв'язку
FR11	Система повинна надсилати повідомлення через зовнішні служби

Нефункціональні вимоги описують, як система виконує свої функції, та визначають такі атрибути якості, як зручність використання, надійність, продуктивність та можливість підтримки.

З точки зору зручності використання, система повинна забезпечувати простий та інтуїтивно зрозумілий інтерфейс як для пацієнтів, так і для персоналу. Процес взаємодії має бути зрозумілим та вимагати мінімального навчання.

Щодо надійності, система повинна забезпечувати стабільну роботу та правильну обробку запитів. Рівень збоїв має бути мінімальним, а цілісність даних має бути збережена за будь-яких умов.

Вимоги до продуктивності особливо важливі через зв'язок у режимі реального часу. Система повинна обробляти та відповідати на запити користувачів протягом 1–2 секунд за нормальних умов навантаження.

Підтримуваність включає ремонтопридатність, масштабованість та можливості інтеграції. Система повинна підтримувати модульну архітектуру, що дозволяє незалежне оновлення таких компонентів, як бекенд Node.js та модуль LLM на основі Python [13].

Компонент "+" FURPS+ включає додаткові обмеження та вимоги, пов'язані з реалізацією, інтеграцією та зовнішніми системами. До них належать:

- використання Node.js для розробки бекенд API;

- використання Python для інтеграції LLM;
- взаємодія із зовнішніми службами обміну повідомленнями через API;
- дотримання вимог захисту даних;
- розгортання в хмарному або локальному серверному середовищі.

Таким чином, застосування моделі FURPS+ дозволяє комплексно визначити системні вимоги та гарантує, що як функціональні можливості, так і атрибути якості належним чином враховуються під час проектування та впровадження системи.

1.9 Висновки до першого розділу

У першому розділі проведено аналіз предметної області прогнозування попиту та особливостей використання часових рядів для розв'язання даної задачі. Розглянуто основні підходи до прогнозування, сучасні моделі машинного навчання та існуючі програмні рішення. Виконано аналіз доступних наборів даних і обґрунтовано вибір датасетів Store Sales – Time Series Forecasting та Walmart Recruiting – Store Sales Forecasting для проведення дослідження.

На основі проведеного аналізу сформовано функціональні та нефункціональні вимоги до програмної системи, побудовано діаграму варіантів використання та визначено основні сценарії взаємодії користувача із системою. Сформульовано задачу дослідження, яка полягає у розробці програмної системи прогнозування попиту та порівнянні моделей ARIMA, LSTM, GRU і Transformer за показниками якості прогнозування [13, 14].

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

Етапи проєктування та розробки програмної системи є основним у процесі виконання кваліфікаційної роботи, тому тут варто чітко підготувати майбутню архітектуру та правильно реалізувати на її основі готову систему.

2.1 Вибір технологій та засобів реалізації

Для розробки системи централізованого управління комунікаціями стоматологічної клініки обрано гнучку методологію Agile, з використанням підходу Scrum. Це обумовлено необхідністю швидкої ітеративної розробки, можливістю внесення змін у вимоги та поетапної перевірки результатів.

Scrum дозволяє розбити процес розробки на короткі спринти тривалістю 1–2 тижні, в межах яких реалізується частина функціональності системи. Після кожного спринту проводиться оцінка результатів і уточнення вимог.

Такий підхід є доцільним для даного проєкту, оскільки система включає інтеграцію декількох компонентів (Node.js backend, Python LLM-модуль, база даних), а також потребує тестування взаємодії між ними [15].

Обрана методологія забезпечує гнучкість, контроль якості та можливість поступового нарощування функціональності системи.

2.2 Загальна архітектура системи (Frontend / Backend / LLM / API)

Розроблена система дотримується модульної архітектури, яка розділяє основні компоненти на фронтенд, бекенд, модуль LLM та зовнішні взаємодії API. Такий підхід забезпечує масштабованість, зручність обслуговування та чіткий розподіл відповідальності між частинами системи (рис. 2.1).

Фронтенд являє собою інтерфейс користувача, через який такі учасники, як Пацієнт та Адміністратор, взаємодіють із системою. Він забезпечує

функціональність для надсилання запитів, перегляду відповідей, керування зустрічами та доступу до відповідної інформації. Фронтенд взаємодіє з бекендом через HTTP-запити.

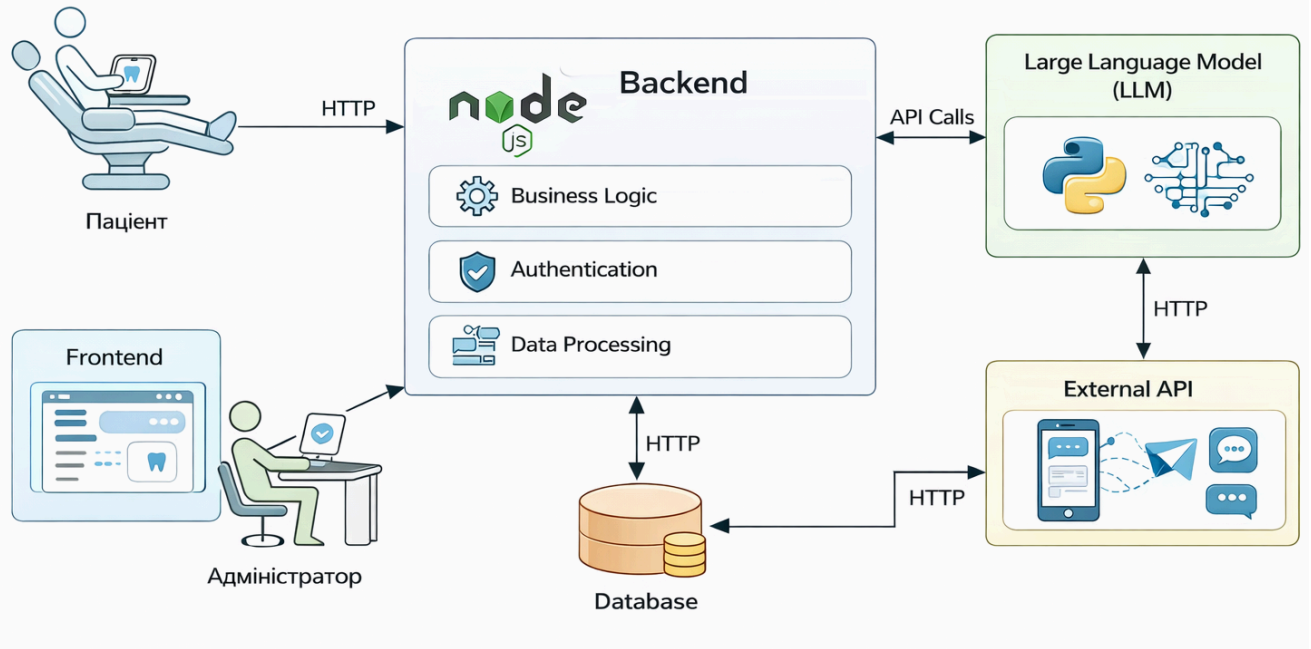

Рисунок 2.1 – Загальна архітектура програмної системи

Бекенд реалізовано за допомогою Node.js та діє як центральний координаційний рівень. Він обробляє вхідні запити, обробляє бізнес-логіку, керує автентифікацією та взаємодіє як з базою даних, так і з модулем LLM. Бекенд також забезпечує перевірку даних та координує зв'язок між компонентами системи.

Модуль LLM реалізовано як окремий сервіс за допомогою Python. Він відповідає за обробку введених даних природною мовою, аналіз запитів користувачів та генерацію відповідей. Бекенд взаємодіє з модулем LLM через виклики API, надсилаючи дані користувача та отримуючи згенеровані відповіді.

Рівень API забезпечує інтеграцію між системою та зовнішніми сервісами, включаючи платформи обміну повідомленнями. Він використовується для надсилання та отримання повідомлень, що дозволяє здійснювати зв'язок через різні канали, такі як месенджери або веб-інтерфейси [16, 17].

База даних зберігає структуровані дані, включаючи інформацію про пацієнтів, розклади прийомів та історію спілкування. Доступ до неї отримує бекенд, що забезпечує збереження та узгодженість даних.

Така архітектура дозволяє незалежну розробку та масштабування кожного компонента, спрощує обслуговування системи та підтримує ефективну інтеграцію інтелектуальної обробки через модуль LLM.

2.3 Проектування бекенду (Node.js API)

Бекенд системи реалізовано за допомогою Node.js та слугує центральним компонентом, відповідальним за обробку запитів, виконання бізнес-логіки та координацію взаємодії між системними модулями (рис. 2.2).

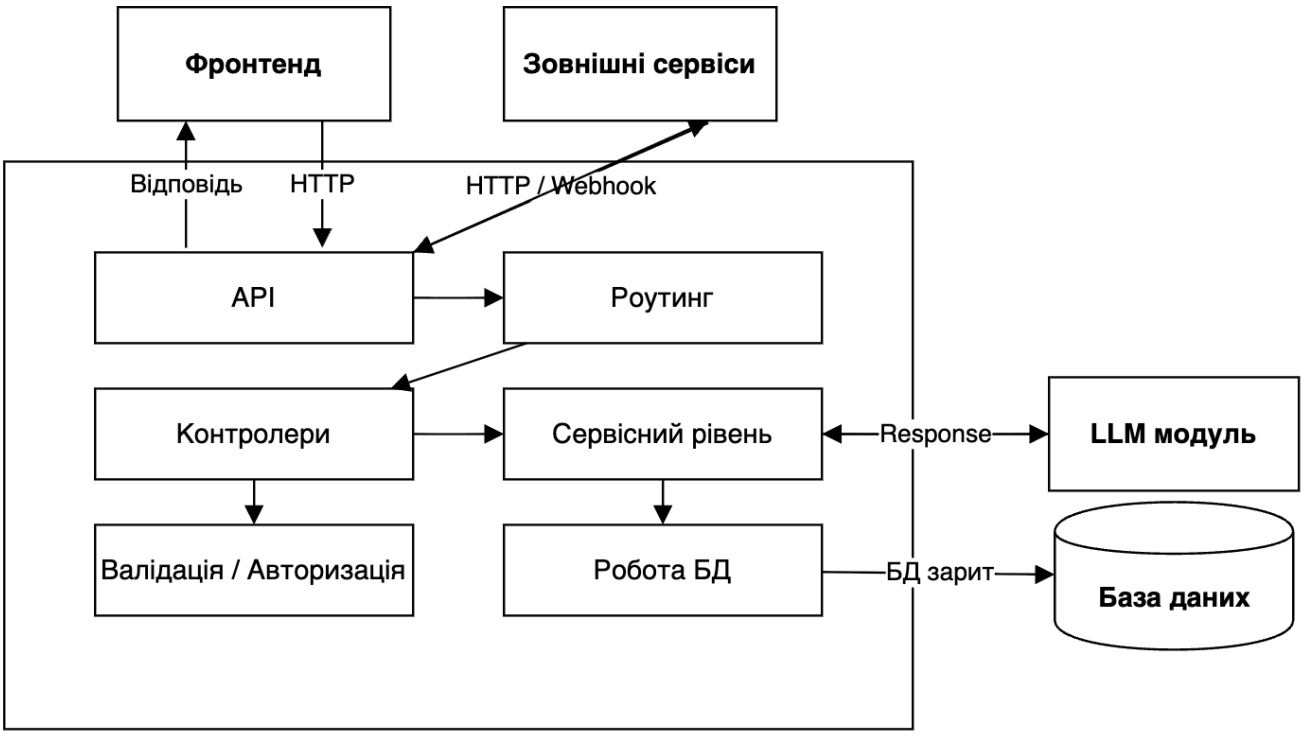

Рисунок 2.2 – Архітектура серверної частини системи на основі Node.js API

Діаграма ілюструє потік взаємодії та внутрішню структуру бекенд-системи на основі API Node.js. Вхідні запити від фронтенд- та зовнішніх сервісів передаються через механізми HTTP або Webhook до рівня API, який виступає

точкою входу в систему. API пересилає запити до компонента маршрутизації, звідки вони спрямовуються до відповідних контролерів. Контролери обробляють запити, включаючи перевірку та авторизацію, і передають логіку до рівня сервісу. Рівень сервісу реалізує основну бізнес-логіку та координує взаємодію як з базою даних, так і з модулем LLM. Операції з базою даних виконуються через спеціалізований компонент доступу до даних, що забезпечує структуроване та ефективне зберігання системних даних. Для інтелектуальної генерації відповідей рівень сервісу взаємодіє з модулем LLM, який повертає оброблені результати. Нарешті, сформована відповідь поширюється назад через API до фронтенд- або зовнішніх сервісів, завершуючи цикл запит-відповідь [18].

Node.js API обробляє вхідні HTTP-запити від фронтенду та зовнішніх сервісів. Він обробляє дані користувача, перевіряє дані та направляє запити до відповідних компонентів, включаючи базу даних та модуль LLM.

Бекенд розроблено з використанням модульної структури, де окремі компоненти обробляють маршрутизацію, бізнес-логіку та доступ до даних. Це покращує зручність обслуговування та дозволяє незалежне оновлення частин системи.

Зв'язок з модулем LLM здійснюється через виклики API. Бекенд надсилає запити користувача до сервісу на основі Python та отримує згенеровані відповіді, які потім повертаються користувачеві.

API також інтегрується із зовнішніми службами обміну повідомленнями, що дозволяє системі надсилати та отримувати повідомлення через різні канали зв'язку.

Таким чином, бекенд Node.js забезпечує централізоване керування роботою системи, надійну обробку даних та інтеграцію всіх компонентів системи.

2.4 Проєктування модуля обробки запитів на основі LLM

Проектування модуля обробки запитів на основі LLM (Python) зосереджено на інтеграції попередньо навченої мовної моделі для автоматизації обробки

запитів користувачів та генерації контекстних відповідей. Замість розробки моделі з нуля було обрано готове до використання рішення — модель LLaMA 3 (через Groq API), яка забезпечує високоякісні відповіді, низьку затримку та доступний безкоштовний тариф для тестування. Модуль реалізовано як окремий сервіс Python, що взаємодіє з бекендом Node.js через REST API. Він отримує структуровані запити, виконує формування запитів, надсилає їх до LLM та обробляє згенеровану відповідь. Застосовуються додаткові кроки попередньої та постобробки для забезпечення релевантності відповідей, фільтрації та форматування відповідно до системних вимог. Такий підхід дозволяє ефективно інтегрувати інтелектуальні можливості, мінімізуючи складність розробки та обчислювальні витрати (рис. 2.3).

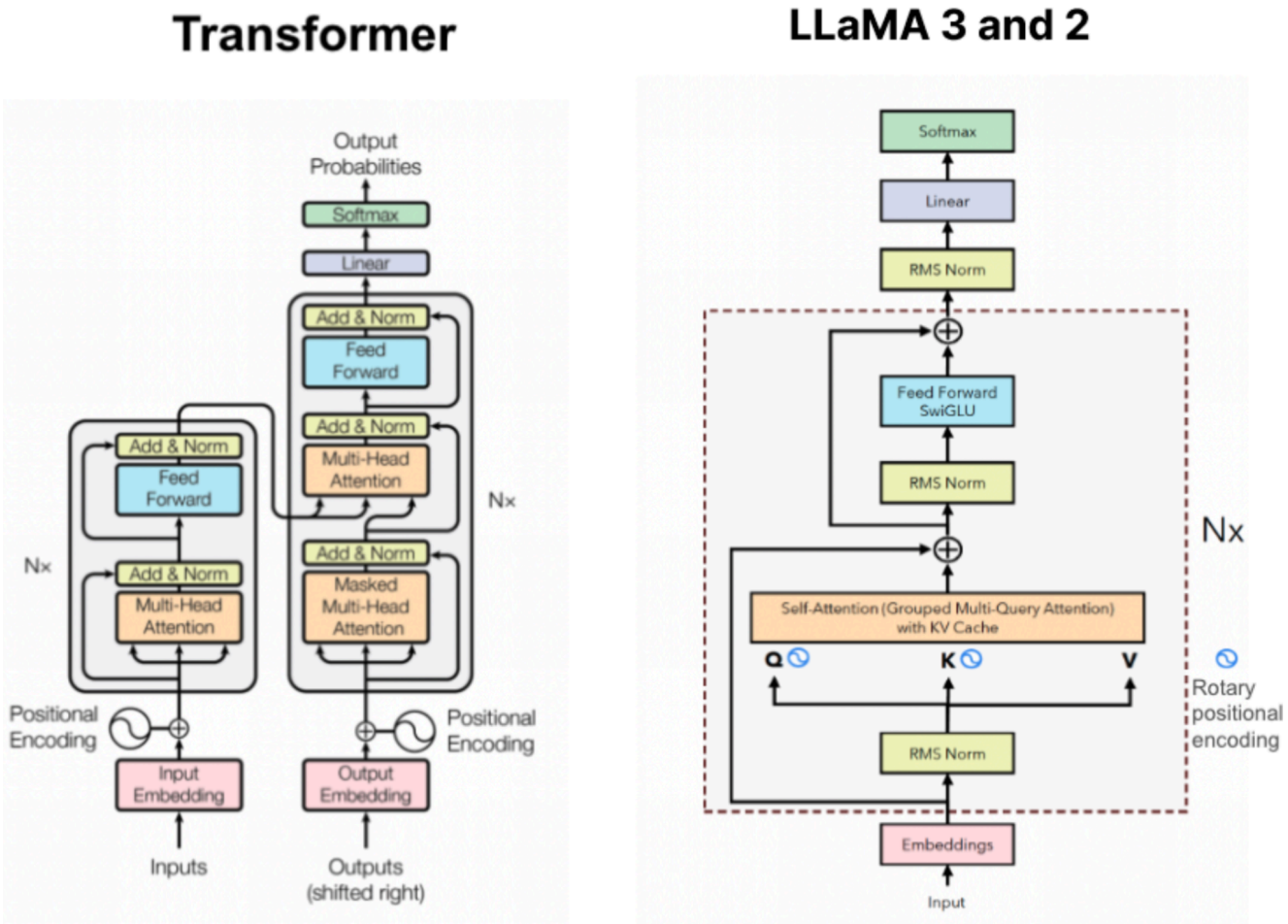

Рисунок 2.3 – Архітектури LLM рішень

Вибір архітектури на основі трансформаторів, зокрема таких моделей, як LLaMA 3, забезпечує кілька важливих переваг для розробленої системи. По-перше, такі моделі демонструють високу точність та контекстне розуміння природної мови, що дозволяє генерувати релевантні та змістовні відповіді на запити користувачів. По-друге, використання попередньо навченої моделі значно скорочує час розробки та усуває необхідність масштабного навчання з нуля. По-третє, сучасні реалізації трансформаторів, включаючи оптимізації, такі як нормалізація RMS та ефективні механізми уваги, забезпечують хорошу продуктивність та масштабованість навіть за обмежень реального часу. Крім того, наявність API та гнучких варіантів інтеграції дозволяє легко інтегрувати модуль LLM в існуючу архітектуру бекенду. Загалом, цей вибір забезпечує баланс між якістю, швидкістю та складністю реалізації, що є критично важливим для практичного розгортання інтелектуальних систем зв'язку [19].

Додатковою перевагою такого підходу є чіткий розподіл відповідальності між компонентами системи, що покращує зручність обслуговування та масштабованість. Виділяючи функціональність LLM у спеціалізований сервіс Python, стає можливим незалежно оновлювати, замінювати або налаштовувати модель, не впливаючи на логіку основного сервера. Крім того, така архітектура підтримує горизонтальне масштабування, де сервіс LLM може бути розгорнутий окремо та масштабований на основі навантаження запитів. Ще одним важливим аспектом є можливість реалізації централізованого контролю над запитами та обробкою відповідей, що покращує узгодженість, відстежуваність та загальну надійність системи в реальних сценаріях використання.

2.5 Проєктування структури бази даних

База даних для розробленої системи реалізована за допомогою PostgreSQL, який був обраний завдяки своїй надійності, підтримці реляційних моделей даних та високій продуктивності в обробці структурованих даних. PostgreSQL надає надійні механізми для забезпечення цілісності даних, транзакційної узгодженості

та масштабованості, що є важливими для системи, яка керує інформацією про пацієнтів та процесами комунікації [20].

Структура бази даних розроблена відповідно до реляційної моделі та включає такі основні сутності: пацієнти, зустрічі, повідомлення, користувачі та llm_responses. Таблиця patients зберігає основну інформацію про клієнтів клініки. Таблиця appointments містить записи про заплановані візити, включаючи дату, час, статус та посилання як на пацієнта, так і на стоматолога. Таблиця messages використовується для зберігання всіх вхідних та вихідних повідомлень, включаючи вміст повідомлень, позначки часу, канал зв'язку та пов'язаного пацієнта. Таблиця users зберігає користувачів системи, таких як адміністратори та стоматологи, включаючи їхні ролі та дані автентифікації. Таблиця llm_responses зберігає згенеровані виходи мовної моделі, що дозволяє відстежувати та аналізувати відповіді системи (рис. 2.4).

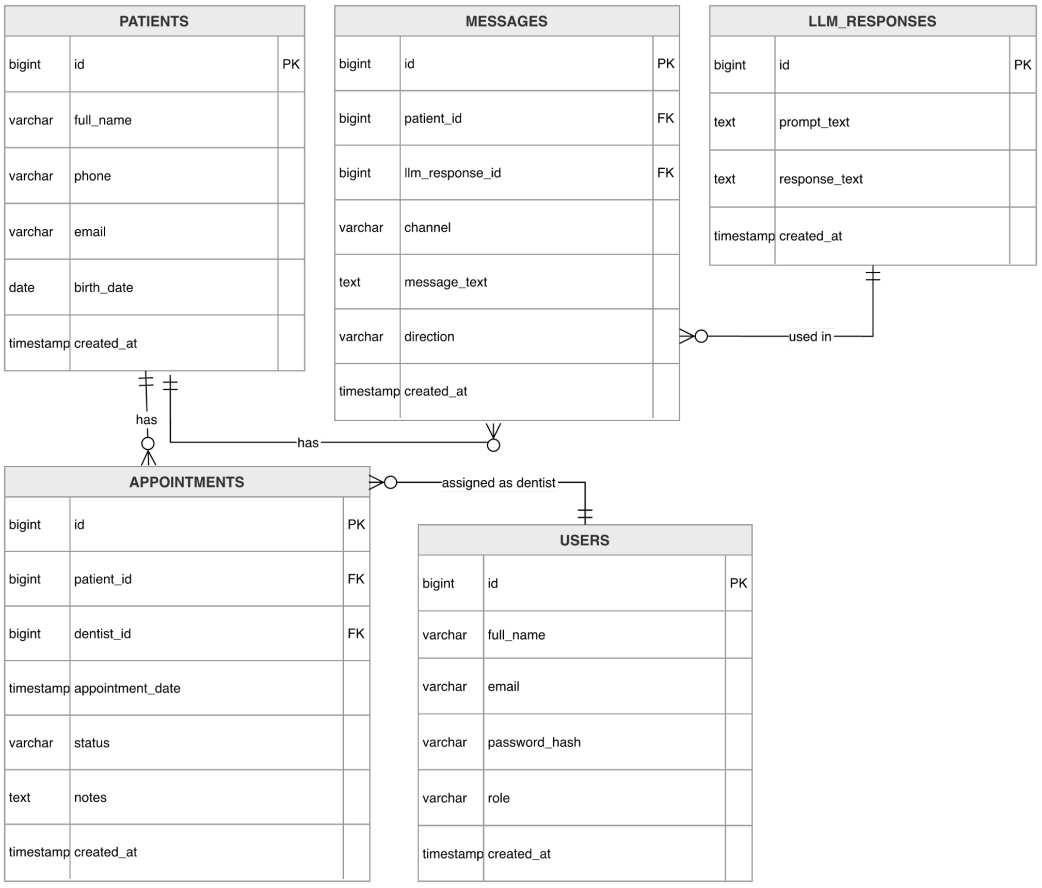

Рисунок 2.4 – ER-діаграма бази даних системи централізованого управління комунікаціями стоматологічної клініки

Зв'язки між таблицями визначаються за допомогою зовнішніх ключів. Один пацієнт може мати кілька записів на прийом та повідомлень. Кожен запис на прийом пов'язаний з одним пацієнтом та певним стоматологом (користувачем). Повідомлення пов'язані з пацієнтами та можуть за бажанням посилатися на згенеровані LLM відповіді. Така структура забезпечує нормалізацію даних, мінімізує надмірність та спрощує виконання запитів.

PostgreSQL також дозволяє використовувати індексацію та обмеження для оптимізації продуктивності запитів та забезпечення узгодженості даних. Наприклад, індекси можна застосовувати до часто запитуваних полів, таких як ідентифікатори пацієнтів або дати записів. Обмеження, такі як NOT NULL, UNIQUE та FOREIGN KEY, забезпечують правильність збережених даних.

Таким чином, обрана база даних та її структура забезпечують стабільну та ефективну основу для зберігання операційних даних, підтримки робочих процесів зв'язку та інтеграції контенту, згенерованого LLM, у систему.

2.6 Побудова структурної моделі програмної системи

На відміну від класичної діаграми класів, зосередженої на об'єктно-орієнтованих ієрархіях, ця модель відображає архітектурну структуру системи, включаючи шари бекенду, модулі інтеграції та сутності даних (рис. 2.5).

Діаграма демонструє багаторівневу організацію бекенду Node.js. Рівень контролерів служить точкою входу для вхідних запитів і відповідає за обробку зв'язку з фронтендом та зовнішніми службами. Ці компоненти делегують обробку рівню сервісів, де реалізується основна бізнес-логіка. Рівень сервісів координує такі операції, як обробка повідомлень, управління призначеннями та обробка даних пацієнтів [21].

Компоненти інтеграції виділені окремо, щоб підкреслити взаємодію із зовнішніми системами. LLMClient відповідає за зв'язок із сервісом LLM на базі Python, тоді як MessagingGateway обробляє інтеграцію із зовнішніми платформами

обміну повідомленнями. Такий поділ забезпечує модульність та спрощує майбутні розширення системи.

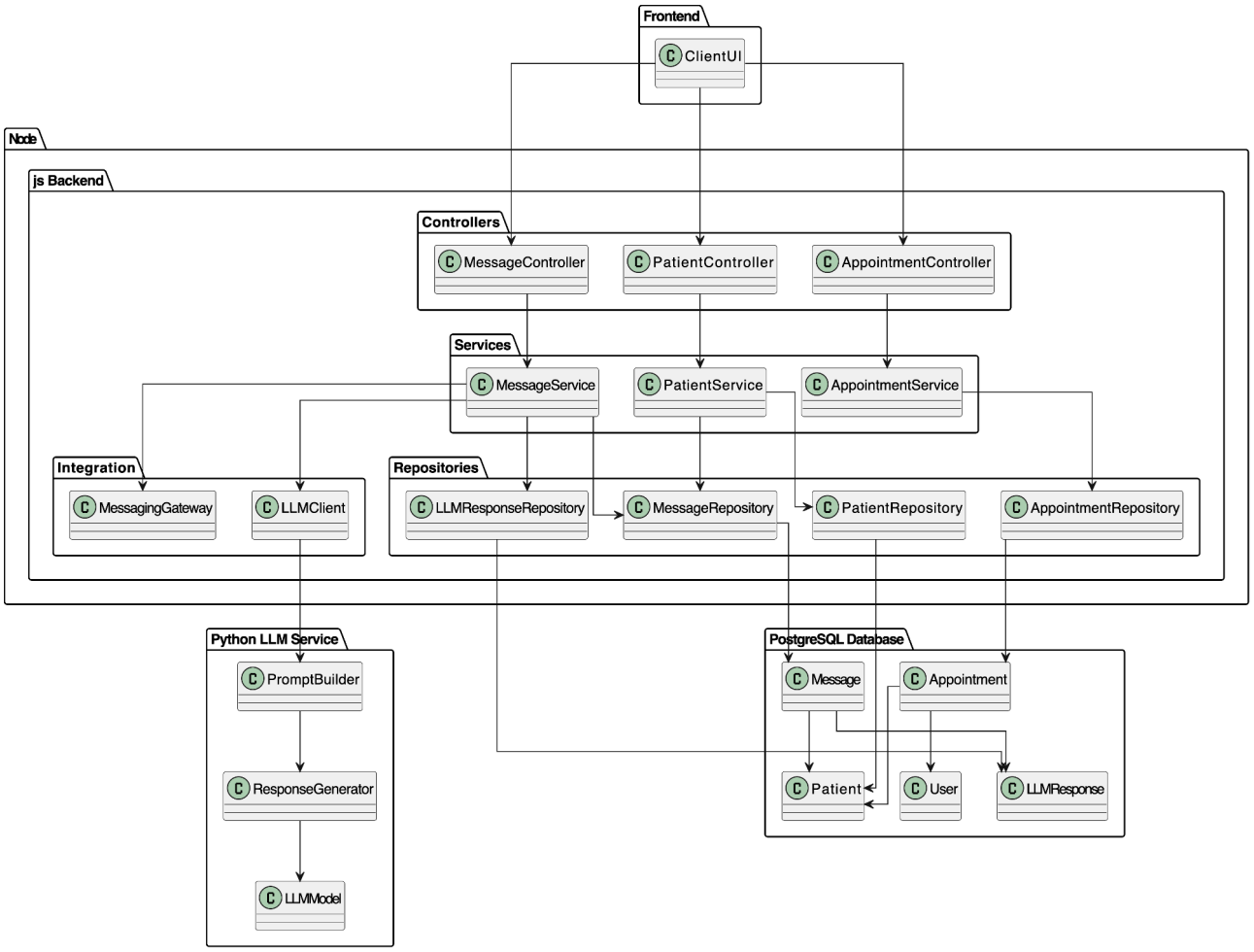

Рисунок 2.5 – Структурна діаграма програмної системи централізованого управління комунікаціями стоматологічної клініки

Доступ до даних організовано через рівень Repository, який інкапсулює взаємодію з базою даних PostgreSQL. Кожен репозиторій відповідає певній сутності та надає структурований інтерфейс для операцій з даними. Такий підхід ізолює логіку бази даних від бізнес-логіки та покращує зручність обслуговування.

Сервіс Python LLM змодельовано як незалежна підсистема, що складається з компонентів, відповідальних за PromptBuilder, ResponseGenerator та взаємодію з мовною моделлю (LLMModel). Це підкреслює відокремлення інтелектуальної обробки від основного бекенду.

На рівні даних система включає основні сутності, такі як Patient, Appointment, Message, User та LLMResponse. Ці сутності представляють основні структури даних, необхідні для підтримки робочих процесів комунікації та управління призначеннями [22].

Загалом, розроблена структурна діаграма UML забезпечує чітке представлення архітектури системи, демонструє розподіл відповідальності між компонентами та ілюструє інтеграцію модуля LLM у логіку бекенду.

2.7 Побудова діаграм взаємодії (Sequence Diagram)

На відміну від структурних діаграм, які представляють статичні зв'язки, діаграми послідовностей зосереджені на порядку операцій та обміні повідомленнями між елементами системи з часом.

Перша діаграма представляє процес обробки повідомлення користувача та генерації відповіді за допомогою модуля LLM. Взаємодія починається, коли Пацієнт надсилає повідомлення через Frontend, який пересилає запит до MessageController. Контролер делегує обробку MessageService, де виконується основна логіка. Сервіс створює запит за допомогою PromptBuilder та надсилає його до LLMClient. Потім запит обробляється LLMModel, який генерує відповідь. Згенерований результат повертається через той самий ланцюжок і доставляється назад Пацієнту через Frontend. Ця діаграма чітко демонструє інтеграцію модуля LLM у робочий процес комунікації та підкреслює розподіл відповідальності між компонентами (рис. 2.6).

Друга діаграма ілюструє процес планування зустрічей. Пацієнт ініціює запит через фронтенд, вибираючи потрібну дату та час. Запит передається до AppointmentController, а потім до AppointmentService, де застосовується валідація та бізнес-логіка. Сервіс взаємодіє з AppointmentRepository, який зберігає дані про зустрічі в PostgreSQL. Після успішного створення підтвердження повертається користувачеві. Цей сценарій відображає типовий транзакційний потік і показує, як система забезпечує збереження та узгодженість даних (рис. 2.7).

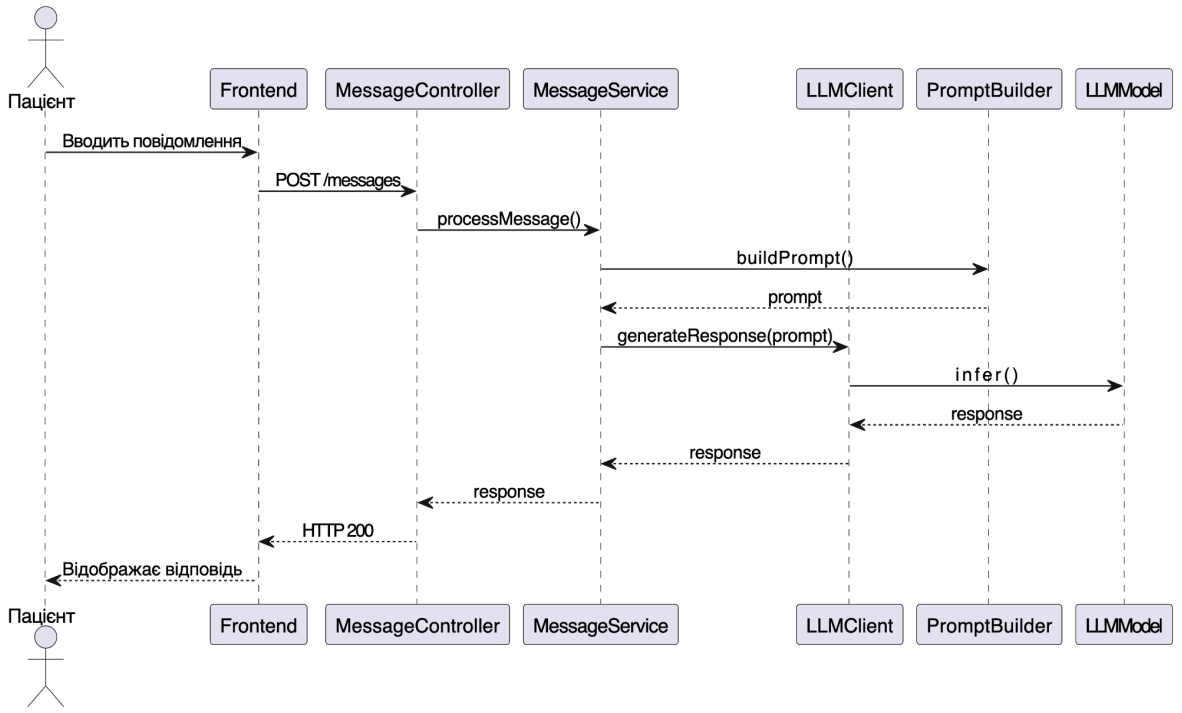

Рисунок 2.6 – Діаграма послідовності обробки повідомлення та генерації відповіді LLM

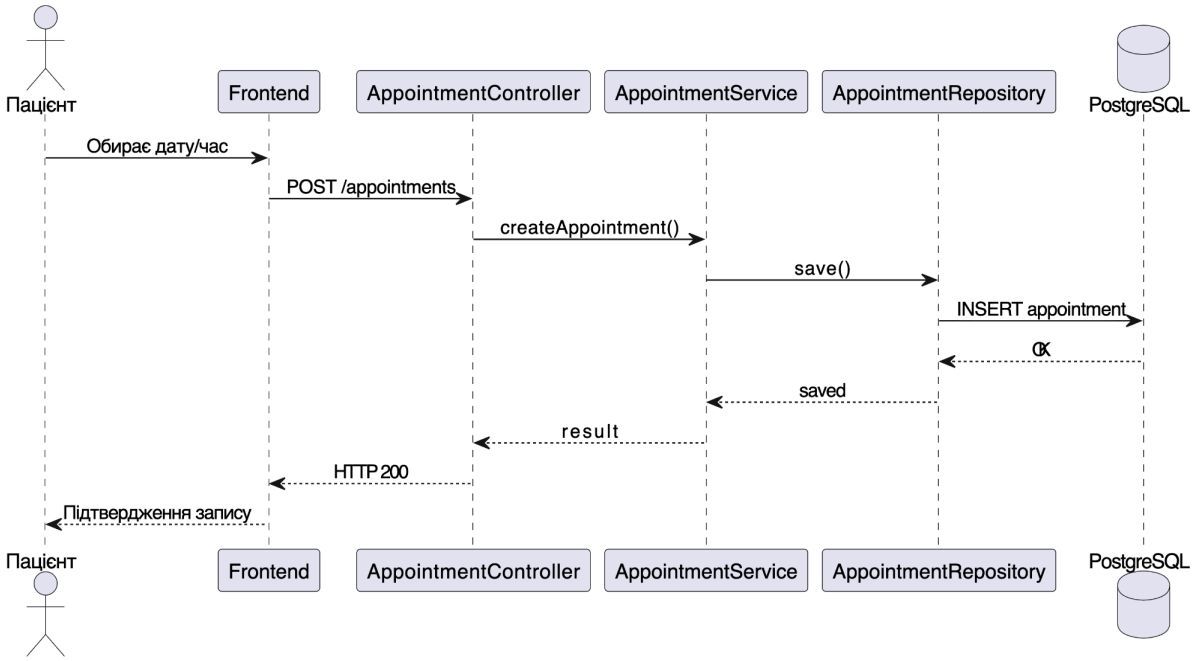

Рисунок 2.7 – Діаграма послідовності запису пацієнта на прийом

Третя діаграма представляє взаємодію із зовнішніми службами обміну повідомленнями. У цьому випадку MessageService надсилає згенеровану відповідь

до MessagingGateway, який діє як посередник між серверною частиною та зовнішніми комунікаційними платформами. Шлюз пересилає повідомлення через зовнішній API та отримує у відповідь статус доставки. Результат потім поширюється назад до системи. Ця діаграма висвітлює рівень інтеграції та демонструє, як система взаємодіє зі сторонніми службами (рис. 2.8).

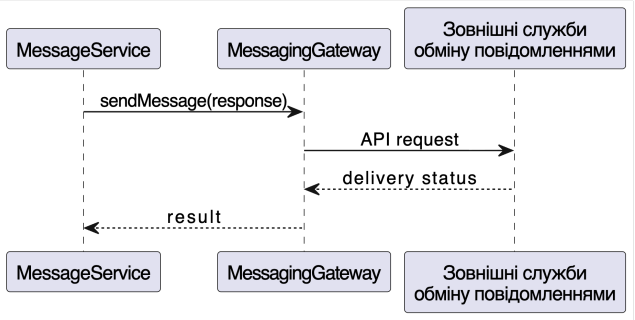

Рисунок 2.8 – Діаграма послідовності взаємодії із зовнішніми сервісами обміну повідомленнями

Разом ці діаграми послідовностей надають повне уявлення про поведінку системи під час виконання. Вони дозволяють визначити ролі окремих компонентів, зрозуміти потік даних та перевірити правильність взаємодії між модулями, включаючи інтеграцію сервісу LLM та зовнішніх каналів зв'язку.

2.8 Вибір технологій та інструментів розробки

Вибір технологій базується на системних вимогах, масштабованості, можливостях інтеграції та ефективності розробки. Система реалізована за допомогою комбінації JavaScript (Node.js) для бекенду та Python для модуля LLM, що дозволяє відокремити бізнес-логіку від інтелектуальної обробки [23].

Node.js використовується для реалізації бекенду API завдяки його високій продуктивності в обробці асинхронних операцій та придатності для побудови RESTful сервісів. Python обраний для модуля LLM через його сильну екосистему для машинного навчання та безшовну інтеграцію із сучасними мовними

моделями. PostgreSQL використовується як основна база даних завдяки своїй надійності, підтримці реляційних даних та транзакційній узгодженості.

Зв'язок між компонентами реалізовано через REST API, що забезпечує слабкий зв'язок між сервісами. Зовнішні сервіси обміну повідомленнями інтегровані через шлюзи API. Для взаємодії з мовною моделлю використовується Groq API для доступу до моделі LLaMA 3, яка забезпечує швидкий висновок та доступний тариф для тестування.

Основні технології та інструменти, що використовуються в системі, наведені в таблиці 2.1.

Таблиця 2.1 – Використані технології та їх призначення

Технологія / Інструмент	**Призначення**
Node.js	Розробка серверної частини системи (REST API, обробка запитів)
Express.js	Маршрутизація запитів та організація структури API
Python	Реалізація модуля обробки запитів на основі LLM
Groq API	Доступ до моделі LLaMA 3 для генерації відповідей
PostgreSQL	Реляційна база даних для зберігання інформації системи
REST API	Забезпечення взаємодії між бекендом та LLM-модулем
Messaging APIs	Інтеграція із зовнішніми сервісами обміну повідомленнями
Git	Контроль версій програмного коду
Postman	Тестування та налагодження API

Цей набір технологій забезпечує модульну архітектуру, ефективну обробку запитів та можливість інтеграції інтелектуальних компонентів у систему.

2.9 Реалізація основних модулів системи

У цьому підрозділі наведено результати реалізації основних модулів системи, а також продемонстровано їх практичну роботу на прикладах взаємодії через API, обробки запитів LLM та збереження даних у базі даних [24].

Бекенд системи реалізовано з використанням Node.js та фреймворку Express.js. Основними компонентами є контролери, сервіси та репозиторії.

Контролери відповідають за прийом HTTP-запитів від клієнтської частини, їх первинну обробку та передачу до відповідних сервісів. Сервісний рівень реалізує бізнес-логіку системи, включаючи обробку повідомлень користувачів, взаємодію з LLM-модулем та управління записами на прийом. Репозиторії забезпечують ізоляцію доступу до бази даних PostgreSQL та виконують операції збереження і отримання даних [25].

На рисунку 2.9 наведено приклад тестування API-ендпоінта обробки повідомлень користувача. Демонструється формування HTTP-запиту типу POST /api/messages, передача JSON-даних, а також отримання відповіді від системи у вигляді структурованого об’єкта, який містить як текстову відповідь, так і додаткову інформацію про обробку запиту.

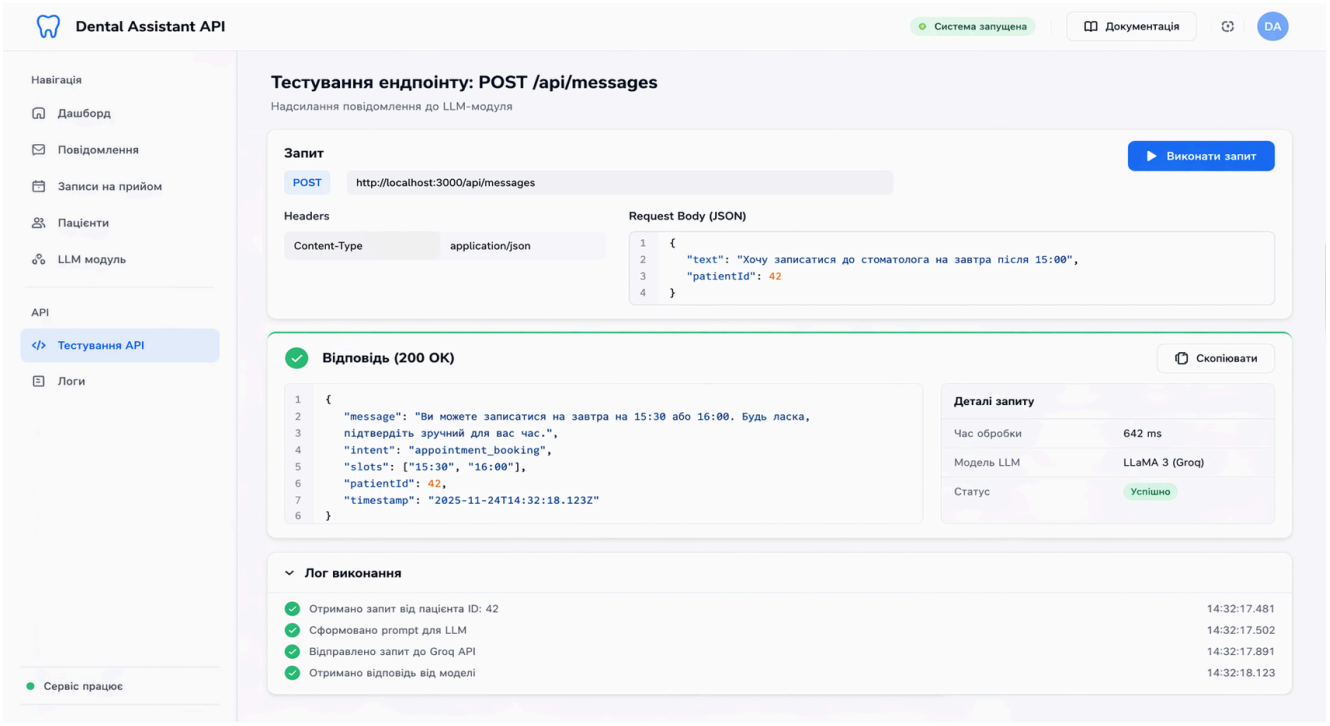

Рисунок 2.9 – Приклад тестування API ендпоінта обробки повідомлень

Модуль обробки запитів на основі LLM реалізовано як окремий сервіс на Python. Він відповідає за формування prompt-запитів, виклик мовної моделі через Groq API та обробку отриманих результатів. Взаємодія з бекендом здійснюється через REST API, що забезпечує слабке зв’язування компонентів системи. У

процесі обробки запиту система формує контекст, передає його до моделі та отримує відповідь, яка далі використовується для формування кінцевого результату для користувача.

На рисунку 2.10 представлено приклад роботи інтерфейсу системи, який демонструє повний цикл обробки запиту: від введення повідомлення користувачем до формування відповіді моделлю. Також відображено сформований prompt (у скороченому вигляді) та відповідь LLM, що дозволяє оцінити якість інтеграції інтелектуального модуля.

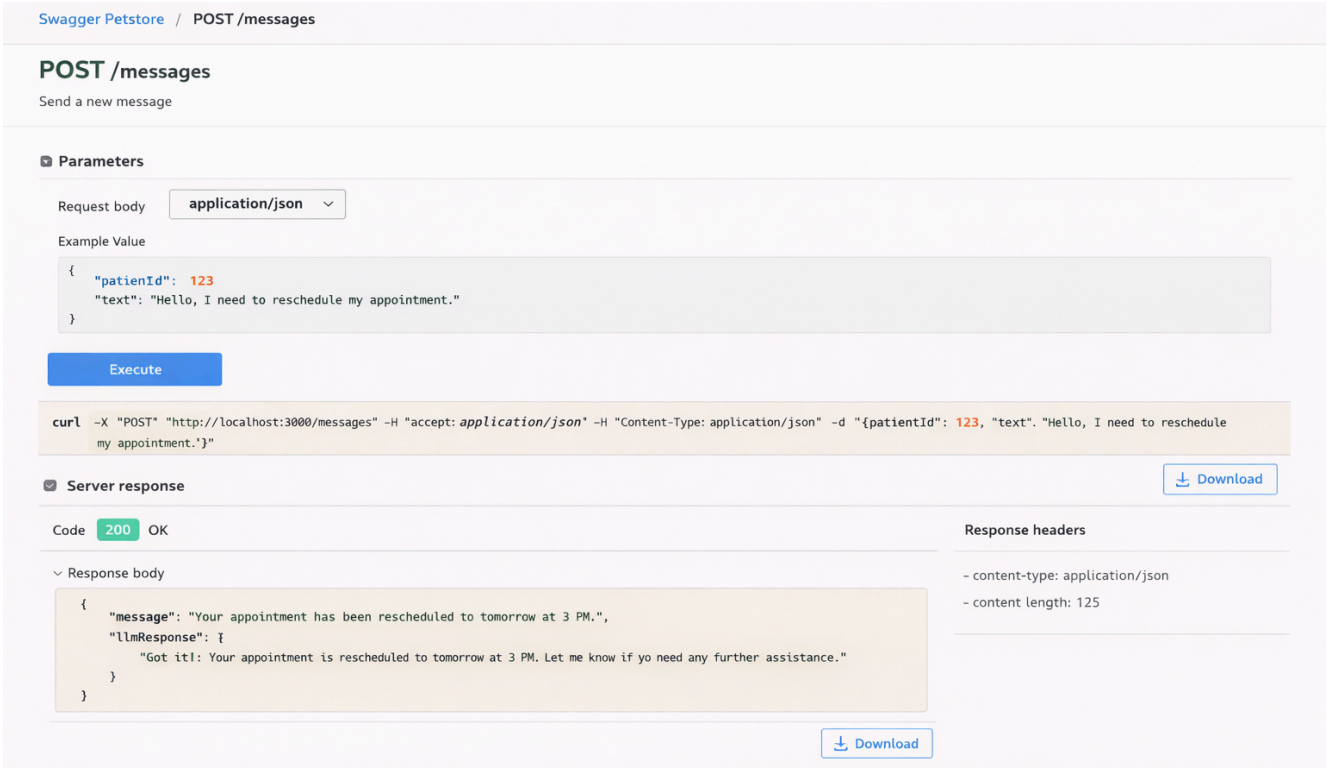

Рисунок 2.10 – Приклад роботи модуля обробки запитів LLM у системі

Зберігання даних реалізовано з використанням реляційної бази даних PostgreSQL. Основними таблицями є messages, appointments та llm_responses. Таблиця messages містить інформацію про вхідні та вихідні повідомлення користувачів, appointments — дані про записи на прийом, а llm_responses — результати обробки запитів мовною моделлю, включаючи сформовані prompt-запити та відповіді.

На рисунку 2.11 наведено приклад структури та вмісту основних таблиць бази даних. Це демонструє, як система зберігає історію взаємодії з користувачами, результати роботи LLM та інформацію про записи, що забезпечує цілісність і відтворюваність процесів.

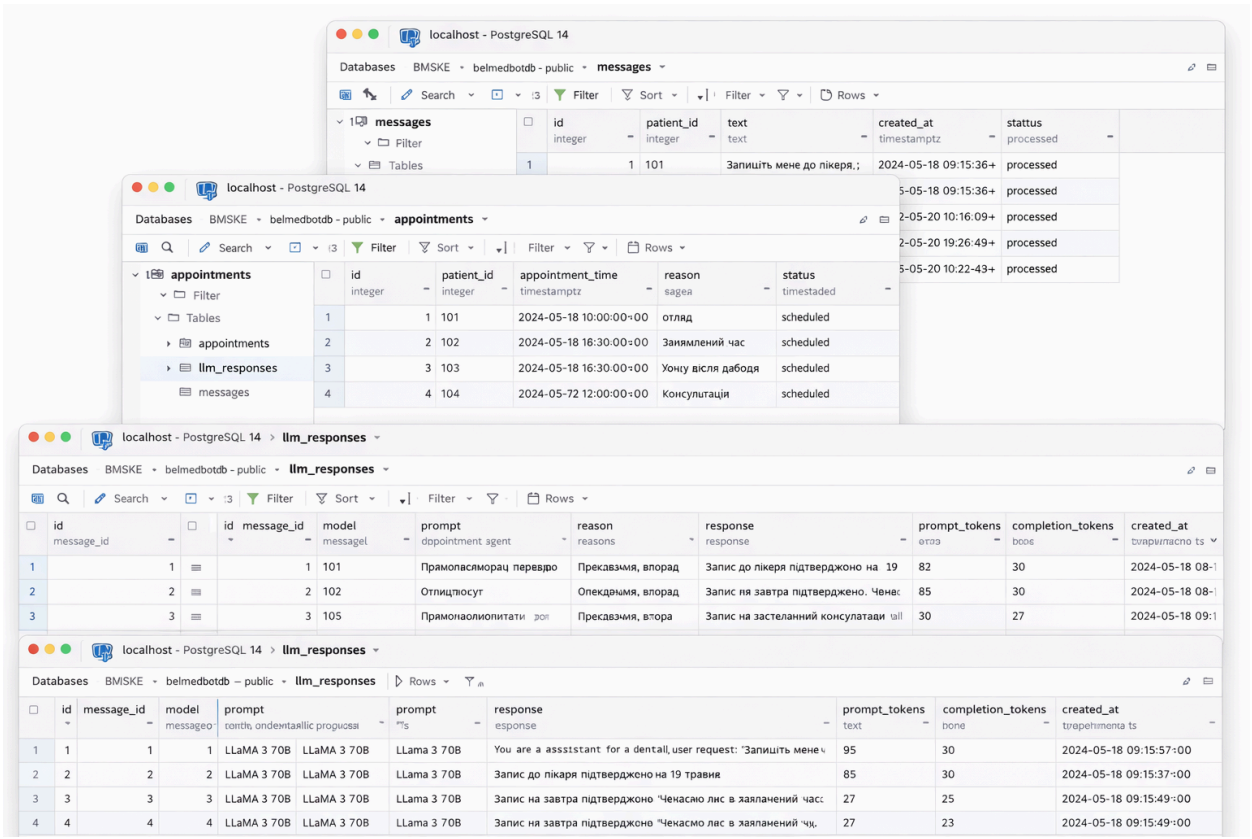

Рисунок 2.11 – Структура та вміст таблиць бази даних системи

Таким чином, реалізація основних модулів підтверджує працездатність розробленої системи, демонструє коректну інтеграцію всіх компонентів та забезпечує виконання поставлених функціональних вимог.

2.10 Реалізація обробки запитів пацієнтів

У межах розробленої системи обробка запитів пацієнтів реалізується через механізм обміну повідомленнями, який інтегрується з зовнішніми каналами комунікації. Як було зазначено раніше, система не має власного користувацького

інтерфейсу, тому взаємодія з користувачем здійснюється через API, що дозволяє підключати різні месенджери та соціальні мережі [26].

Серед найбільш поширених платформ, які можуть бути інтегровані з системою, варто виділити Telegram, Viber, WhatsApp, Facebook Messenger та інші сервіси, що надають відкриті API для обробки повідомлень. Такі платформи дозволяють отримувати текстові повідомлення користувачів, передавати їх до серверної частини системи, обробляти за допомогою LLM та повертати сформовану відповідь назад користувачу.

У рамках даного проєкту пряма інтеграція з реальними API месенджерів не була реалізована. Це пов’язано з необхідністю проходження авторизації, використання бізнес-акаунтів та дотриманням політик відповідних платформ. Замість цього було використано тестове середовище, яке дозволяє імітувати процес обміну повідомленнями та зосередитися безпосередньо на логіці роботи системи та обробці запитів.

На рисунку 2.12 представлено загальну схему взаємодії між зовнішніми месенджерами та розробленою системою.

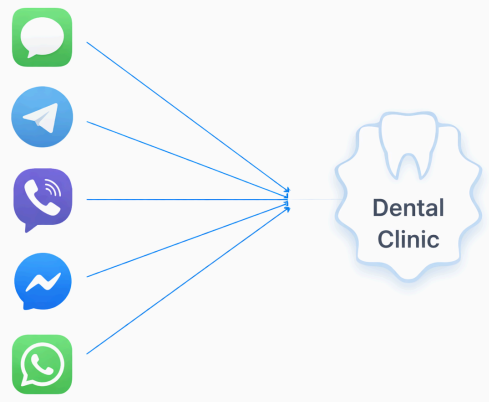

Рисунок 2.12 – Сервіси для API взаємодії

Процес обробки запиту включає декілька основних етапів. Спочатку повідомлення користувача надходить через API-запит до серверної частини системи. Далі воно зберігається у базі даних та передається до модуля обробки LLM. Модель аналізує зміст повідомлення, визначає намір користувача та формує

відповідь, яка може включати як текстову консультацію, так і пропозицію щодо запису на прийом. Після цього відповідь повертається користувачу через відповідний канал комунікації.

Для демонстрації роботи системи було змодельовано приклади взаємодії користувача з чат-ботом у різних месенджерах. На рисунках 2.13 та 2.14 наведено приклади діалогів, у яких користувач звертається із запитом щодо стоматологічної допомоги, а система автоматично формує відповідь та пропонує запис на прийом.

Рисунок 2.13 – Приклад взаємодії у Telegram та Viber

Як видно з наведених прикладів, система здатна коректно обробляти різні типи запитів, зокрема запити на консультацію, уточнення вартості послуг та запис на прийом. Важливою особливістю є те, що незалежно від формулювання запиту, система приводить діалог до логічного завершення — успішного запису пацієнта на прийом, що є ключовою бізнес-ціллю даного рішення (рис. 2.14).

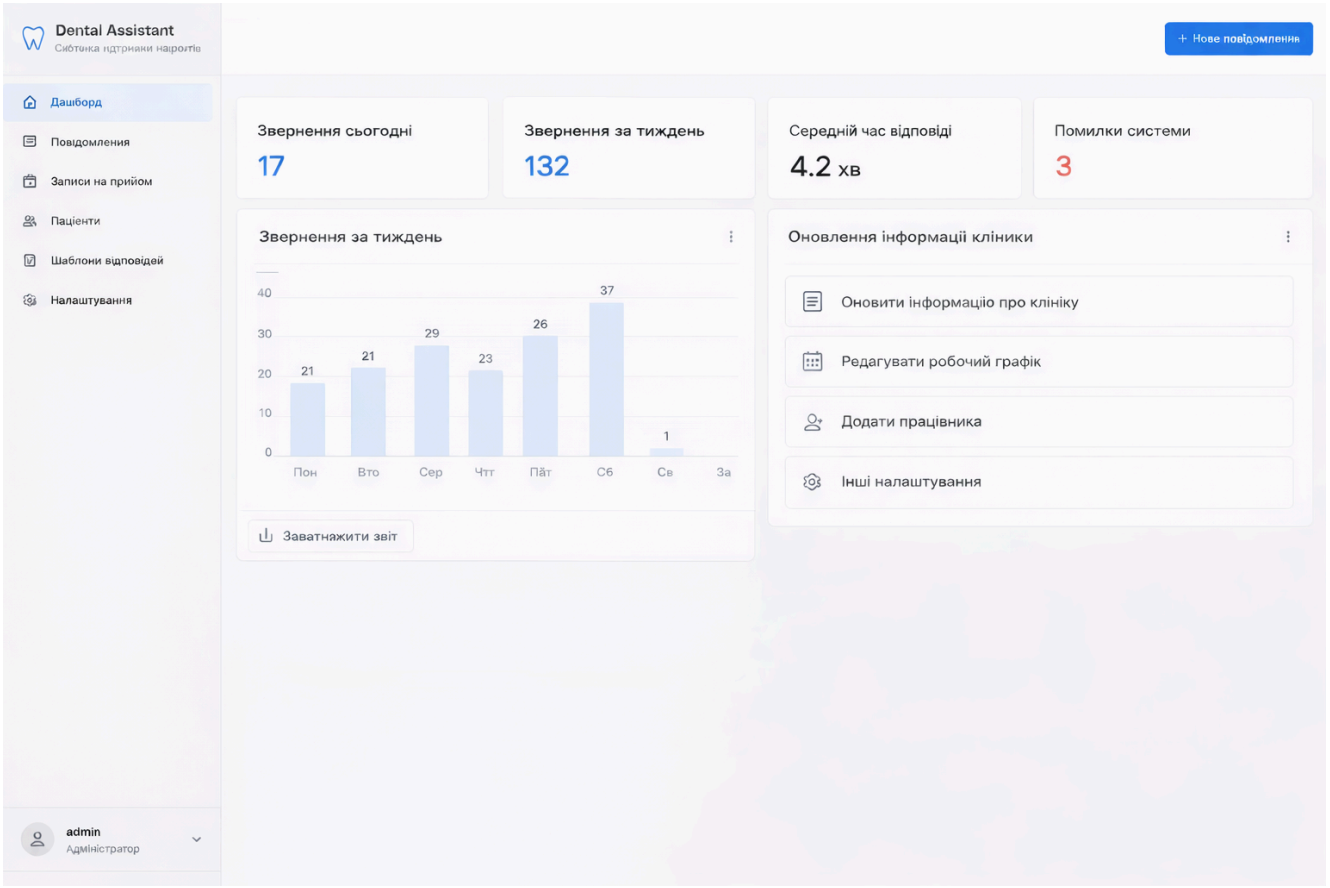

Рисунок 2.14 – Інтерфейс панелі інструментів із системними показниками та інструментами управління клінікою

Панель інструментів розробленої системи забезпечує централізований інтерфейс для моніторингу взаємодії з пацієнтами та управління роботою клініки. Вона відображає ключові показники ефективності, включаючи кількість запитів пацієнтів, отриманих протягом поточного дня та тижня, середній час відповіді та кількість системних помилок. Це дозволяє адміністраторам швидко оцінювати навантаження системи та загальну продуктивність.

Окрім аналітичних даних, панель інструментів містить функції управління. Користувач може оновлювати інформацію про клініку, змінювати графік роботи, додавати нових співробітників та отримувати доступ до додаткових налаштувань системи. Також надається візуальне представлення щотижневих запитів у вигляді діаграми, яка допомагає аналізувати тенденції та визначати періоди пікової активності.

Такий інтерфейс забезпечує як операційний контроль, так і аналітичну інформацію, що робить систему практичною для реального використання в стоматологічних клініках.

2.11 Висновки до розділу 2

У другому розділі було виконано проєктування та реалізацію програмної системи централізованого управління комунікаціями стоматологічної клініки з використанням великої мовної моделі. Розроблено архітектуру системи, яка включає Node.js бекенд, Python-модуль для роботи з LLM, базу даних PostgreSQL та механізми інтеграції через REST API.

Було спроєктовано структуру бази даних, визначено основні сутності та зв’язки між ними, а також розроблено UML-діаграми, що відображають структуру та взаємодію компонентів системи. Реалізовано основні програмні модулі, включаючи обробку повідомлень, управління записами на прийом та взаємодію з мовною моделлю через Groq API.

Крім того, було продемонстровано роботу системи на прикладах обробки звернень пацієнтів та автоматичного формування відповідей. Отримані результати підтверджують можливість використання розробленого рішення для автоматизації комунікаційних процесів стоматологічної клініки.

3 ТЕСТУВАННЯ, ОЦІНЮВАННЯ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ

Для повноцінної роботи системи та чіткого виконання всіх функцій проведено комплексне тестування, яке було розбито на різні етапи відповідно до вимог, поставлених до проєкту програмної системи.

3.1 Планування тестування програмної системи

Тестування розробленої системи спрямоване на перевірку коректності всіх компонентів, стабільності обробки запитів та якості відповідей, що генеруються мовною моделлю. Завдяки архітектурі системи, тестування включає як класичні підходи до тестування програмного забезпечення, так і специфічні перевірки, пов'язані з використанням LLM.

Процес тестування був організований у кілька етапів, з поділом на різні рівні тестування. Це дає можливість локалізувати помилки та оцінювати поведінку кожного модуля незалежно.

У таблиці 3.1 представлені основні типи тестування, що застосовуються в розробленій системі.

Таблиця 3.1 – Типи системного тестування

Вид тестування	**Опис**	**Об’єкт тестування**	**Інструменти / підхід**
Модульне тестування (Unit Testing)	Перевірка коректності роботи окремих функцій та програмних модулів	Сервіс обробки повідомлень, модуль роботи з БД, LLM-модуль	Jest, PyTest, локальні виклики
Інтеграційне тестування (Integration Testing)	Перевірка взаємодії між компонентами системи	REST API, LLM-модуль, база даних PostgreSQL	Postman, інтеграційні сценарії
Тестування API (API Testing)	Перевірка коректності обробки HTTP-запитів та відповідей	REST API endpoint-и	Postman, Swagger
Тестування LLM (LLM Testing)	Оцінка якості відповідей мовної моделі	Генерація відповідей на запити пацієнтів	Тестові сценарії, експертна оцінка

Модульне тестування використовується для перевірки окремих функцій, таких як обробка тексту, генерація запитів та операції збереження даних. Інтеграційне тестування перевіряє взаємодію між API, базою даних та модулем LLM. Тестування API виконується за допомогою таких інструментів, як Postman та Swagger, для перевірки HTTP-запитів та відповідей. LLM-тестування проводиться окремо, оскільки якість відповіді залежить не лише від реалізації коду, але й від того, як модель інтерпретує вхідні запити.

Для перевірки системи було підготовлено набір тестових сценаріїв для імітації реальних запитів пацієнтів. Основна мета — перевірити, чи правильно система обробляє запити та спрямовує розмову до успішного запису на прийом.

Таблиця 3.2 – Приклади тестових сценаріїв

Вхідний запит користувача	**Дія системи**	**Очікуваний результат**
«У мене болить зуб»	Аналіз звернення та визначення наміру користувача	Рекомендація записатися на консультацію до стоматолога-терапевта
«Скільки коштує видалення зуба мудрості?»	Пошук інформації про послугу	Надання інформації про вартість та пропозиція запису на прийом
«Хочу записатися на прийом»	Ініціація процедури запису	Відображення доступних часових слотів
«Можна на завтра після 15:00?»	Перевірка доступного розкладу	Пропозиція найближчих доступних часових слотів
«Запишіть мене на 16:00»	Створення запису на прийом	Підтвердження успішного бронювання та повідомлення деталей запису

Кожен тестовий сценарій включає три ключові елементи: вхідний запит, очікувану логіку обробки та кінцевий очікуваний результат. Важливою вимогою є те, що навіть за різних формулювань запитів система повинна призводити до послідовного та правильного виводу.

3.2 Тестування взаємодії між Python (LLM) та Node.js API

Взаємодія між Node.js та сервісом LLM відбувається за моделлю запит-відповідь. Бекенд надсилає структурований HTTP-запит, що містить введені користувачем дані та контекстні дані, до сервісу Python. Модуль LLM обробляє запит, генерує відповідь, використовуючи мовну модель, та повертає результат у форматі JSON. Потім серверна частина інтерпретує відповідь та пересилає її користувачеві.

Тестування цієї взаємодії включає кілька ключових аспектів. По-перше, перевіряється правильність форматування запиту, включаючи структуру JSON, обов'язкові поля та кодування. По-друге, виконується перевірка відповіді, щоб гарантувати, що сервіс LLM повертає коректні та належним чином структуровані дані. По-третє, тестуються механізми обробки помилок для перевірки поведінки системи у таких випадках, як недоступність сервісу, недійсні відповіді або тайм-аути. Для перевірки взаємодії використовувалися інструменти тестування API, такі як Postman, для моделювання запитів від серверної частини до сервісу LLM. Тестові випадки включали надсилання типових запитів користувачів, перевірку вмісту відповідей та вимірювання часу відгуку. Особлива увага приділялася забезпеченню того, щоб система правильно обробляла багатомовний ввід та генерувала відповіді потрібною мовою.

Приклад тестового запиту від бекенду Node.js до сервісу Python LLM наведено нижче в лістингу 3.1, а результат виконання на рисунку 3.1.

Лістинг 3.1 – Приклад тестового запиту

```
POST /llm/process
Content-Type: application/json
{
  "message": "I have a toothache, what should I do?",
  "context": {
    "language": "uk",
    "intent": "consultation"
  }
}
```

```
{
  "response": "Рекомендуємо записатися на
    консультацію до стоматолога. Доступні слоти: 15
    :30, 16:00.",
  "status": "success"
}
```


Рисунок 3.1 – Очікувана відповідь від сервісу LLM

Під час тестування було підтверджено, що система правильно обробляє передачу запитів, розбір відповідей та подальшу обробку всередині бекенд. Зв'язок між Node.js та сервісом Python залишається стабільним за нормальних умов та підтримує необхідну функціональність для обробки повідомлень у режимі реального часу.

3.3 Оцінка якості відповіді LLM

Оцінка модуля LLM зосереджена на якості згенерованих відповідей з точки зору точності та релевантності запитам користувачів. Оскільки система працює в розмовній області, традиційних метрик класифікації недостатньо. Натомість використовується комбінація якісних та напівкількісних підходів до оцінки.

Точність у цьому контексті визначається як ступінь, до якої згенерована відповідь правильно відображає намір користувача та надає фактично доречну інформацію. Відповідь вважається точною, якщо вона не містить оманливих або неправильних рекомендацій та відповідає очікуваній поведінці системи.

Релевантність відображає, наскільки добре відповідь відповідає запиту користувача та чи вирішує вона фактичну проблему. Релевантна відповідь повинна безпосередньо відповідати на запитання, уникати зайвої інформації та спрямовувати користувача до змістовного результату, такого як запис на прийом.

Для оцінки цих критеріїв було підготовлено набір тестових запитів, що охоплюють різні типи намірів користувачів, включаючи запити на консультації, запити щодо вартості послуг та запис на прийом. Кожна відповідь, згенерована LLM, оцінювалася вручну на основі попередньо визначених критеріїв.

У таблиці 3.3 представлено результати оцінювання.

Таблиця 3.3 – Оцінювання відповідей LLM

№	Запит користувача	Точність (0–1)	Релевантність (0–1)	Коментар
1	«У мене болить зуб»	1	1	Коректне визначення проблеми та пропозиція консультації
2	«Скільки коштує видалення зуба мудрості?»	0,9	1	Коректна відповідь із незначним узагальненням
3	«Хочу записатися на прийом»	1	1	Правильне визначення наміру користувача та запуск процедури запису
4	«Можна на завтра після 15:00?»	1	0,9	Запропоновано відповідний час, однак відповідь є дещо узагальненою
5	«У мене болить ясна після лікування»	0,9	0,9	Надано релевантну рекомендацію, але відсутнє детальне уточнення симптомів

Оцінювання показує, що LLM стабільно видає високоякісні відповіді, зі значеннями точності та релевантності, близькими до 1,0 у більшості випадків. Незначні відхилення в основному пов'язані з узагальненням або відсутністю детального контексту, що типово для мовних моделей.

Важливим спостереженням є те, що система успішно підтримує потік розмови та веде користувача до цільової дії, такої як запис на прийом. Це підтверджує, що інтеграція модуля LLM відповідає функціональним вимогам системи та підтримує ефективну комунікацію з користувачами.

3.4 Розгортання системи та системні вимоги

Розгортання розробленої системи базується на розподіленій архітектурі, де основні компоненти розгортаються як окремі сервіси. Бекенд Node.js, модуль LLM на основі Python та база даних PostgreSQL можуть розміщуватися як на локальному сервері, так і в хмарному середовищі. Такий підхід забезпечує

гнучкість, масштабованість та можливість незалежного керування кожним компонентом.

Бекенд-додаток розгортається як сервіс REST API за допомогою Node.js. Він обробляє вхідні запити, керує бізнес-логікою та взаємодіє як з базою даних, так і з модулем LLM. Сервіс LLM, реалізований на Python, розгортається як окремий сервіс, який надає кінцеву точку API для обробки запитів. Зв'язок між цими сервісами здійснюється через HTTP з використанням формату JSON.

База даних PostgreSQL розгортається як окремий сервіс та зберігає всі операційні дані, включаючи повідомлення, зустрічі та відповіді LLM. Для забезпечення стабільної роботи необхідна правильна конфігурація підключень до бази даних, індексації та контролю доступу. Розгортання може бути виконане за допомогою технологій контейнеризації, таких як Docker, що спрощує налаштування та забезпечує узгодженість у різних середовищах. Кожен компонент (бекенд, сервіс LLM, база даних) може бути упакований в окремі контейнери та оркестрований за допомогою Docker Compose або подібних інструментів.

Система може працювати у двох основних середовищах: локальна розробка та продакшн. У локальному середовищі всі компоненти працюють на одній машині для тестування та налагодження. У продакшн-середовищі сервіси можуть бути розподілені між кількома серверами або хмарними екземплярами для забезпечення вищої доступності та продуктивності.

У таблиці 3.4 представлено системні вимоги для розгортання.

Таблиця 3.4 – Системні вимоги

Компонент	**Вимога**
Операційна система	Windows / Linux / macOS
Середовище виконання Backend	Node.js 18+
Середовище виконання LLM-сервісу	Python 3.10+
База даних	PostgreSQL 13+
Оперативна пам’ять	Не менше 8 ГБ
Процесор	Від 4 ядер
Мережа	Стабільне підключення до Інтернету
Додаткові інструменти	Docker (за потреби), Postman

Окрім технічних вимог, інтеграція системи в реальну стоматологічну клініку вимагає надання структурованих даних, специфічних для предметної області, модулю LLM. Якість відповідей безпосередньо залежить від повноти та правильності цієї інформації.

Клініка, яка планує інтегрувати систему, повинна надати такі дані:

- загальну інформацію про клініку (назва, адреса, години роботи);
- перелік доступних послуг (консультації, лікування, хірургічне втручання, діагностика);
- інформацію про ціни або діапазони цін на послуги;
- доступний графік прийому (інтервали, робочі дні, доступність лікаря);
- перелік спеціалістів (ролі, спеціальності);
- правила бронювання, скасування та перенесення запису;
- типові відповіді або правила спілкування (необов'язково).

Ці дані слід надавати у структурованому форматі, такому як JSON, для забезпечення належної інтеграції з модулем LLM. Приклад структури наведено нижче в лістингу 3.2.

Лістинг 3.2 – Приклад тестового запиту

```
{
  "clinic": {
    "name": "Dental Clinic",
    "working_hours": "09:00-18:00"
  },
  "services": [
    {
      "name": "Consultation",
      "price": "from 500 UAH"
    },
    {
      "name": "Wisdom tooth removal",
      "price": "from 2000 UAH"
    }
  ],
  "schedule": {
    "available_slots": ["15:30", "16:00"]
  }
}
```

Надання цієї інформації дозволяє LLM генерувати точні, контекстно-залежні та персоналізовані відповіді. Це гарантує, що систему можна адаптувати до різних клінік без зміни основної логіки, що робить її масштабованою та придатною для повторного використання в кількох розгортаннях.

3.5 Аналіз продуктивності та масштабованості системи

Для оцінки ефективності роботи розробленої системи було проведено аналіз її продуктивності та масштабованості. Під час тестування оцінювалися час обробки запитів, використання апаратних ресурсів та здатність системи працювати за одночасного надходження великої кількості звернень. Основні результати тестування наведено в таблиці 3.5.

Таблиця 3.5 – Основні показники продуктивності системи

Показник	**Значення**
Час відповіді API (без LLM)	80–120 мс
Час відповіді API (з LLM)	650–1200 мс
Максимальний час відповіді	до 1,8 с
Кількість одночасних користувачів	120–150
Частка помилок	< 1 %
Використання RAM бекенду	150–250 МБ
Використання RAM LLM-модуля	300–500 МБ

Як видно з таблиці 3.5, система забезпечує стабільну роботу при обробці запитів користувачів та демонструє прийнятні показники продуктивності. Отримані результати свідчать про можливість подальшого масштабування системи шляхом збільшення обчислювальних ресурсів або розгортання додаткових екземплярів сервісів.

3.6 Висновки до розділу 3

У третьому розділі проведено тестування, оцінювання та аналіз ефективності розробленої системи. Було сформовано план тестування, який охоплює модульне, інтеграційне, API-тестування та оцінювання якості відповідей мовної моделі. Для перевірки працездатності системи розроблено набір тестових сценаріїв, що відображають типові звернення пацієнтів стоматологічної клініки.

Результати тестування підтвердили коректність взаємодії між Node.js бекендом, базою даних PostgreSQL та Python-модулем, який забезпечує доступ до LLM. Проведена оцінка якості відповідей показала високий рівень точності та релевантності під час обробки інформаційних запитів і сценаріїв запису на прийом.

Також було виконано аналіз продуктивності та масштабованості системи. Отримані показники свідчать про стабільну роботу програмного забезпечення та можливість його подальшого використання в умовах реальної експлуатації. Результати верифікації та валідації підтвердили відповідність реалізованої системи поставленим вимогам і її готовність до подальшого розвитку та інтеграції з реальними комунікаційними платформами.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

У даному розділі розглядаються основні вимоги охорони праці та безпеки життєдіяльності, яких необхідно дотримуватися під час виконання робіт з використанням комп'ютерної техніки. Особливу увагу приділено організації робочого місця та забезпеченню безпечних умов праці відповідно до чинних нормативних документів.

4.1 Значення адаптації в трудовому процесі

Адаптація працівника до умов праці є одним із важливих чинників забезпечення безпеки життєдіяльності, ефективності трудової діяльності та збереження здоров'я працівників. Під адаптацією розуміють процес пристосування людини до умов виробничого середовища, особливостей трудового процесу, технічних засобів праці та соціально-психологічних умов колективу [27]. Успішна адаптація сприяє формуванню професійних навичок, підвищенню продуктивності праці та зменшенню ймовірності виникнення виробничого травматизму.

Згідно з положеннями охорони праці, одним із головних завдань організації трудового процесу є створення таких умов, за яких працівник може швидко та безпечно пристосуватися до виконання своїх функціональних обов'язків [28]. Особливого значення це набуває під час роботи з комп'ютерною технікою та інформаційними системами, де основне навантаження припадає на органи зору, нервову систему та психоемоційний стан працівника.

Відповідно до вимог ДСТУ ISO 6385:2005 «Ергономічні принципи проєктування робочих систем», під час створення робочих місць необхідно враховувати фізіологічні та психологічні можливості людини, особливості сприйняття інформації та взаємодії з технічними системами [29]. Дотримання

ергономічних принципів дозволяє скоротити період адаптації працівника до нових умов праці та зменшити рівень втоми під час виконання професійних завдань.

Важливим аспектом адаптації є психофізіологічна адаптація, яка передбачає пристосування організму людини до інтенсивності трудового процесу, рівня інформаційного навантаження та режиму праці й відпочинку [30]. Недостатній рівень адаптації може призводити до розвитку професійної втоми, зниження концентрації уваги, погіршення якості виконання роботи та збільшення кількості помилок під час прийняття рішень.

Для працівників, діяльність яких пов’язана з використанням персональних комп’ютерів, важливими є також вимоги щодо організації робочого місця. Відповідно до ДСанПіН 3.3.2.007-98, робочі місця користувачів комп’ютерної техніки повинні забезпечувати безпечні умови праці, належний рівень освітлення, оптимальні параметри мікроклімату та ергономічне розміщення обладнання [31]. Недотримання цих вимог може ускладнювати процес адаптації та негативно впливати на працездатність працівників.

У межах розробленої системи централізованого управління комунікаціями стоматологічної клініки питання адаптації має важливе практичне значення. Адміністратор клініки працює з веб-інтерфейсом системи, здійснює обробку звернень пацієнтів, контролює записи на прийом та аналізує відповіді, сформовані мовною моделлю. Для забезпечення швидкої адаптації користувача під час проєктування системи було враховано принципи ергономіки інтерфейсів, логічне структурування інформації та мінімізацію кількості дій, необхідних для виконання типових операцій. Важливим результатом успішної адаптації є зниження ризику виникнення професійних помилок під час роботи з комп’ютерними системами. На початкових етапах використання нового програмного забезпечення працівники витрачають більше часу на пошук необхідних функцій та виконання стандартних операцій, що може призводити до підвищеного психоемоційного навантаження. Після завершення адаптаційного періоду кількість помилок зменшується, підвищується швидкість обробки інформації та покращується якість виконання службових обов’язків [29]. Саме тому під час розроблення інформаційних систем

важливо враховувати вимоги ергономіки та принципи людино-орієнтованого проєктування.

Крім того, використання автоматизованої системи дозволяє зменшити інформаційне навантаження на адміністратора шляхом автоматичної обробки типових запитів пацієнтів. Це сприяє зниженню психоемоційної напруги та створює більш комфортні умови праці. Відповідно до рекомендацій Міжнародної організації праці, автоматизація рутинних процесів є одним із ефективних способів покращення умов праці та підвищення продуктивності персоналу [32]. Окремою складовою трудової адаптації є соціально-психологічна адаптація працівника, яка полягає у пристосуванні до організаційної структури підприємства, особливостей взаємодії з колегами та прийнятих правил виконання роботи [30]. В умовах цифровізації значна частина професійних обов'язків виконується із застосуванням інформаційних систем, тому ефективність роботи персоналу значною мірою залежить від рівня прийняття нових технологій та готовності до їх використання [30]. Наявність зрозумілого інтерфейсу, автоматизованих механізмів підтримки прийняття рішень та можливості швидкого доступу до необхідної інформації сприяє формуванню позитивного ставлення працівників до нових програмних засобів і скорочує період їх адаптації. Як зазначається у роботах з охорони праці, скорочення часу адаптації безпосередньо впливає на підвищення продуктивності праці, зменшення кількості помилок та покращення загального психологічного клімату в колективі [30].

Таким чином, адаптація працівника до умов праці є важливою складовою безпеки життєдіяльності та ефективної організації трудового процесу. Використання сучасних інформаційних систем, розроблених із дотриманням вимог ергономіки, охорони праці та психофізіологічних особливостей людини, дозволяє скоротити період адаптації, знизити рівень професійного навантаження та підвищити ефективність роботи персоналу.

4.2 Інженерно-технічні рішення з охорони праці

Одним із важливих завдань охорони праці є забезпечення безпечних та комфортних умов роботи працівників. Для фахівців, діяльність яких пов’язана з використанням комп’ютерної техніки та інформаційних систем, продуктивність праці значною мірою залежить від умов робочого середовища, рівня інформаційного навантаження та ергономічності робочого місця. Недотримання вимог охорони праці може призводити до зниження працездатності, виникнення професійної втоми, підвищення психоемоційного навантаження та погіршення стану здоров’я працівників [28].

У межах розробленої системи централізованого управління комунікаціями стоматологічної клініки основним користувачем є адміністратор, який здійснює обробку звернень пацієнтів, аналізує відповіді, сформовані мовною моделлю, та керує записами на прийом. Тому особливого значення набуває оцінка умов праці працівника, який виконує роботу із застосуванням персонального комп’ютера.

Відповідно до Постанови Кабінету Міністрів України №442 від 01.08.1992 р. атестація робочих місць за умовами праці є одним із основних інструментів оцінювання впливу виробничих факторів на працівника [34]. Результати такої оцінки використовуються для визначення рівня безпечності умов праці та розроблення заходів щодо їх покращення.

Для аналізу умов праці використовується «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу», затверджена наказом Міністерства охорони здоров’я України №248 від 08.04.2014 р. [35]. Згідно з даною класифікацією умови праці поділяються на чотири класи: оптимальні, допустимі, шкідливі та небезпечні.

Під час експлуатації розробленої інформаційної системи основними факторами, що можуть впливати на працівника, є тривала робота за комп’ютером, статичне навантаження під час сидіння, навантаження на органи зору, підвищена концентрація уваги та необхідність обробки значних обсягів інформації. На

відміну від виробничих підприємств, де переважають фізичні небезпеки, для операторів інформаційних систем найбільш характерними є психофізіологічні та ергономічні фактори ризику [27].

Важкість трудового процесу визначається рівнем фізичного навантаження на працівника. Для адміністратора стоматологічної клініки фізичне навантаження є незначним і пов'язане переважно зі статичним положенням тіла під час роботи за комп'ютером. Водночас напруженість праці є достатньо високою через необхідність постійної роботи з інформацією, контролю звернень пацієнтів, прийняття рішень щодо запису на прийом та аналізу відповідей, сформованих системою [29].

Відповідно до вимог ДСанПіН 3.3.2.007-98 робоче місце користувача персонального комп'ютера повинно забезпечувати належний рівень освітлення, оптимальні параметри мікроклімату, ергономічне розташування обладнання та можливість зміни робочої пози [31]. Дотримання цих вимог сприяє зниженню втомлюваності працівників та підвищує ефективність їх діяльності.

Проведений аналіз дозволяє зробити висновок, що умови праці адміністратора стоматологічної клініки, який використовує розроблену систему, можуть бути віднесені до допустимого класу умов праці за умови дотримання встановлених санітарно-гігієнічних вимог та правил організації робочого місця. Для подальшого покращення умов праці доцільно використовувати ергономічні меблі, забезпечувати регламентовані перерви під час роботи з комп'ютером та підтримувати оптимальні параметри виробничого середовища.

Шкідливі умови праці за ступенем перевищення гігієнічних нормативів та вираженості можливих змін в організмі працюючих поділяються на чотири ступені [34]. Проте під час експлуатації розробленої системи централізованого управління комунікаціями стоматологічної клініки більшість небезпечних виробничих факторів, характерних для промислових підприємств, відсутні. Основними чинниками, що можуть впливати на працівника, є тривала робота за персональним комп'ютером, статичне навантаження, напруження органів зору та підвищене інформаційне навантаження.

Ступінь шкідливості умов праці для адміністратора стоматологічної клініки визначається насамперед параметрами мікроклімату приміщення, рівнем освітленості робочого місця, тривалістю роботи з візуальними дисплейними терміналами та психоемоційним навантаженням під час обробки звернень пацієнтів. Особливу увагу необхідно приділяти організації робочого місця відповідно до вимог ДСанПіН 3.3.2.007-98 та ДСТУ ISO 6385:2005, які регламентують ергономічні вимоги до робочих систем та умов праці користувачів комп'ютерної техніки [29, 31].

Оцінка важкості трудового процесу для даного робочого місця здійснюється за показниками статичного навантаження, тривалості перебування працівника в сидячому положенні та характеру виконуваних операцій. Оскільки діяльність адміністратора не пов'язана зі значними фізичними навантаженнями, показники важкості праці можуть бути віднесені до допустимого рівня. Водночас напруженість трудового процесу є більш суттєвим фактором і визначається необхідністю постійного контролю інформаційних потоків, прийняття рішень щодо запису пацієнтів на прийом, перевірки коректності відповідей, сформованих мовною моделлю, та взаємодії з різними каналами комунікації.

Для наочного представлення результатів аналізу умов праці проведено оцінку основних виробничих факторів, що впливають на користувача системи. Результати оцінювання наведено в таблиці 4.1.

Як видно з таблиці 4.1, більшість факторів виробничого середовища належать до допустимого класу умов праці. Найбільший вплив на працівника мають фактори напруженості трудового процесу, пов'язані з інтелектуальним та емоційним навантаженням. Водночас застосування розробленої системи дозволяє частково компенсувати вплив зазначених факторів завдяки автоматизації обробки типових запитів пацієнтів та централізованому управлінню інформацією.

Використання великої мовної моделі дозволяє автоматично формувати відповіді на поширені запити, що зменшує навантаження на адміністратора та скорочує час обробки звернень. Крім того, централізоване зберігання інформації

про пацієнтів, повідомлення та записи на прийом забезпечує швидкий доступ до необхідних даних і підвищує ефективність виконання службових обов’язків.

Таблиця 4.1 – Оцінка умов праці адміністратора стоматологічної клініки

Фактор	**Характеристика**	**Клас умов праці**
Фізичне навантаження	Незначне, відсутність переміщення вантажів	2
Статичне навантаження	Тривала робота у сидячому положенні	2
Навантаження на органи зору	Робота з монітором протягом робочого дня	2
Інтелектуальне навантаження	Аналіз звернень пацієнтів та контроль роботи системи	2
Емоційне навантаження	Взаємодія з пацієнтами та контроль записів	2
Мікроклімат та освітлення	Відповідають нормативним вимогам	2
Загальна оцінка умов праці	Допустимі умови праці	2

На підставі проведеного аналізу можна зробити висновок, що умови праці користувача розробленої системи належать до допустимого класу умов праці (2 клас) за показниками шкідливості та небезпечності виробничих факторів. За умови дотримання вимог чинних нормативних документів щодо організації робочого місця, режимів праці та відпочинку, освітлення та параметрів мікроклімату робота із системою не створює значного ризику для здоров’я працівника та забезпечує належний рівень працездатності протягом робочої зміни.

ВИСНОВКИ

У межах виконаної роботи було розроблено програмну систему для автоматизованої обробки запитів пацієнтів стоматологічної клініки на основі використання великої мовної моделі. Основне призначення системи полягає у забезпеченні швидкої та релевантної відповіді на звернення користувачів, а також у підтримці процесу запису на прийом без участі адміністратора. Розроблене рішення орієнтоване на задачу автоматизації комунікації з пацієнтами через текстові повідомлення, що дозволяє зменшити навантаження на персонал та підвищити якість обслуговування.

Поставлена мета роботи була досягнута. У процесі розробки реалізовано ключові компоненти системи, зокрема модуль обробки запитів на основі LLM, серверну частину на Node.js, окремий Python-сервіс для взаємодії з мовною моделлю, а також базу даних для збереження інформації про повідомлення, пацієнтів та відповіді системи. Забезпечено інтеграцію між усіма компонентами через API, що дозволяє організувати узгоджену роботу системи.

Основні результати роботи полягають у побудові багаторівневої архітектури, яка поєднує Node.js бекенд, Python-сервіс та LLM як інтелектуальний модуль обробки тексту. Реалізовано механізм прийому та аналізу запитів користувачів, формування prompt-запитів, генерації відповідей та їх подальшої обробки. Інтеграція компонентів здійснюється через REST API, що забезпечує гнучкість та масштабованість рішення. Для зберігання даних використовується реляційна база даних, у якій організовано структури для повідомлень, записів на прийом та результатів роботи моделі. Також визначено типові сценарії роботи системи, що відображають реальні взаємодії пацієнтів із клінікою.

У процесі дослідження було проведено тестування системи на різних рівнях, включаючи модульне, інтеграційне, API-тестування та оцінку якості відповідей LLM. Отримані результати показали високий рівень точності та релевантності відповідей у типових сценаріях використання. Система продемонструвала стабільну роботу при обробці запитів, коректну взаємодію між компонентами та

відповідність визначеним вимогам. Це підтверджує доцільність використання обраного підходу для побудови інтелектуальних систем підтримки комунікації.

Практична цінність розробленої системи полягає у можливості її застосування в реальних умовах стоматологічних клінік для автоматизації обробки запитів пацієнтів. Система може бути використана як інтелектуальний помічник для прийому звернень, надання консультацій та організації запису на прийом. Для бізнесу це означає зменшення навантаження на адміністративний персонал, скорочення часу обробки запитів та підвищення швидкості реагування на звернення клієнтів. Крім того, автоматизація комунікації дозволяє забезпечити безперервну підтримку пацієнтів, що позитивно впливає на рівень сервісу та задоволеність клієнтів.

Перспективи розвитку системи пов’язані з розширенням її функціональних можливостей та поглибленням інтеграції. Доцільним є впровадження прямої інтеграції з популярними платформами обміну повідомленнями, такими як Telegram, WhatsApp та іншими, що дозволить використовувати систему в реальних каналах комунікації. Важливим напрямом є донавчання мовної моделі на специфічних даних клініки для підвищення точності відповідей. Також можливе розширення системи за рахунок додавання CRM-функціоналу, аналітичних інструментів та підтримки декількох мов, що зробить рішення більш універсальним і придатним для різних типів медичних установ.

У підсумку, розроблена система демонструє ефективність поєднання сучасних підходів до обробки природної мови з класичною серверною архітектурою. Використання LLM у поєднанні з бекенд-сервісами дозволяє створити гнучке, масштабоване та інтелектуальне рішення для автоматизації комунікації. Отримані результати підтверджують доцільність використання такого підходу та створюють основу для подальшого розвитку подібних систем у сфері медичних інформаційних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Михалик Д. М., Цуприк Г. Б., Бревус В. М. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за освітньо-професійною програмою «Інженерія програмного забезпечення» спеціальності 121 – «Інженерія програмного забезпечення» всіх форм навчання. Тернопіль : ТНТУ ім. І. Пулюя, 2024. 45 с.

2. Volodymyr Semchyshyn; Dmytro Mykhalyk Data-driven decision-making methods and hierarchical analysis in cloud-based medical service management systems / ITTAP-2025: 5th International Workshop on Information Technologies: Theoretical and Applied Problems| (2025), CEUR Workshop Proceedings Volume 4146. P.478-486

3. Oleh Zaiats; Dmytro Mykhalyk; Vasyl Yatsyshyn; Oleh Pastukh; Dmytro Uhryn Methods for integrating large language models into requirements management in agile methodologies / ITTAP-2025: 5th International Workshop on Information Technologies: Theoretical and Applied Problems| (2025), CEUR Workshop Proceedings Volume 4146. P.379-397

4. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p.

5. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

6. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston : Addison-Wesley, 2004. 560 p.

7. Newman S. Building Microservices. 2nd ed. Sebastopol : O'Reilly Media, 2021. 616 p.

8. Brown S. Software Architecture for Developers. London : Leanpub, 2022. 284 p.

9. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 08.06.2026).

10. Node.js Foundation. Node.js Documentation. URL: https://nodejs.org/docs (дата звернення: 08.06.2026).

11. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 08.06.2026).

12. Python Software Foundation. Python Documentation. URL: https://docs.python.org/3/ (дата звернення: 08.06.2026).

13. OpenAPI Initiative. OpenAPI Specification. URL: https://swagger.io/specification/ (дата звернення: 08.06.2026).

14. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. Irvine : University of California, 2000. 162 p.

15. Richardson L., Amundsen M., Ruby S. RESTful Web APIs. Sebastopol : O'Reilly Media, 2013. 408 p.

16. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. Draft. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 08.06.2026).

17. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson, 2021. 1168 p.

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

19. Touvron H. et al. LLaMA: Open and Efficient Foundation Language Models. arXiv. 2023. URL: https://arxiv.org/abs/2302.13971 (дата звернення: 08.06.2026).

20. Touvron H. et al. Llama 3 Model Card. Meta AI. URL: https://ai.meta.com/llama/ (дата звернення: 08.06.2026).

21. Groq Documentation. URL: https://console.groq.com/docs (дата звернення: 08.06.2026).

22. Postman Learning Center. URL: https://learning.postman.com/ (дата звернення: 08.06.2026).

23. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

24. ISO/IEC/IEEE 29148:2018. Systems and Software Engineering – Life Cycle Processes – Requirements Engineering. Geneva : ISO, 2018.

25. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмних засобів. Вимоги та оцінювання якості систем і програмного забезпечення (SQuaRE). Моделі якості системи та програмного продукту. Київ : ДП «УкрНДНЦ», 2016.

26. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 775 p.

27. Желібо Є. П., Заверуха Н. М. Безпека життєдіяльності : навч. посіб. Київ : Каравела, 2020. 344 с.

28. Гандзюк М. П., Желібо Є. П., Халімовський М. О. Основи охорони праці : підручник. Київ : Каравела, 2021. 512 с.

29. ДСТУ ISO 6385:2005. Ергономічні принципи проєктування робочих систем. Київ : Держспоживстандарт України, 2007. 18 с.

30. Крушельницька О. В. Управління персоналом : навч. посіб. Київ : Кондор, 2018. 308 с.

31. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Київ : МОЗ України, 1998.

32. International Labour Organization. Occupational Safety and Health. URL: https://www.ilo.org/global/topics/safety-and-health-at-work (дата звернення: 01.06.2026).

33. Про порядок проведення атестації робочих місць за умовами праці : Постанова Кабінету Міністрів України від 01.08.1992 р. № 442. URL: https://zakon.rada.gov.ua/laws/show/442-92-п (дата звернення: 01.06.2026).

34. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу : наказ Міністерства охорони здоров'я України від 08.04.2014 р. № 248. URL: https://zakon.rada.gov.ua/laws/show/z0472-14 (дата звернення: 01.06.2026).

ДОДАТКИ

ДОДАТОК А

Основні файли взаємодії елементів системи

Лістинг коду А.1 - Приклад структурованого файлу з інформацією про клініку

```
{
  "clinic": {
    "name": "Dental Clinic SmileCare",
    "city": "Ternopil",
    "address": "15 Ruska St.",
    "phone": "+380671234567",
    "email": "info@smilecare.ua",
    "working_hours": {
      "monday_friday": "09:00-18:00",
      "saturday": "10:00-15:00",
      "sunday": "closed"
    }
  },
  "services": [
    {
      "id": 1,
      "name": "Initial consultation",
      "category": "consultation",
       "description": "Primary examination, complaint analysis,
and treatment plan discussion.",
      "price_uah": 500,
      "duration_min": 30
    },
    {
      "id": 2,
      "name": "Professional cleaning",
      "category": "hygiene",
       "description": "Removal of plaque and tartar, polishing,
hygiene recommendations.",
      "price_uah": 1200,
      "duration_min": 45
    },
    {
      "id": 3,
      "name": "Caries treatment",
      "category": "therapy",
            "description": "Treatment of dental caries with
restoration.",
      "price_uah": 1800,
      "duration_min": 60
    },
    {
```

```
            "id": 4,
            "name": "Wisdom tooth removal",
            "category": "surgery",
             "description": "Simple or complex extraction of a wisdom
tooth.",
            "price_uah": 2500,
            "duration_min": 60
          },
          {
            "id": 5,
            "name": "Orthodontic consultation",
            "category": "orthodontics",
                 "description": "Assessment of bite and treatment
options.",
            "price_uah": 700,
            "duration_min": 40
          }
        ],
        "doctors": [
          {
            "id": 1,
            "full_name": "Dr. Olena Melnyk",
            "specialization": "therapist",
            "services": [1, 2, 3]
          },
          {
            "id": 2,
            "full_name": "Dr. Andrii Kovalenko",
            "specialization": "surgeon",
            "services": [1, 4]
          },
          {
            "id": 3,
            "full_name": "Dr. Iryna Bondar",
            "specialization": "orthodontist",
            "services": [1, 5]
          }
        ],
        "schedule": {
          "available_slots": [
            {
              "doctor_id": 1,
              "date": "2026-04-21",
              "times": ["10:00", "11:00", "15:30", "16:30"]
            },
            {
              "doctor_id": 2,
              "date": "2026-04-21",
              "times": ["12:00", "14:00", "17:00"]
            },
            {
              "doctor_id": 3,
              "date": "2026-04-21",
```

```
            "times": ["09:30", "13:00"]
          }
        ]
      },
      "booking_rules": {
        "appointment_required": true,
        "reschedule_allowed_hours_before": 12,
        "cancellation_allowed_hours_before": 12,
        "online_booking_supported": true
      },
      "communication_rules": {
        "language": "Ukrainian",
        "tone": "polite, professional, concise",
        "ask_for_confirmation_before_booking": true,
        "do_not_provide_medical_diagnosis": true,
        "offer_consultation_if_symptoms_are_unclear": true
      },
      "faq": [
        {
          "question": "How much does wisdom tooth removal cost?",
             "answer": "Wisdom tooth removal costs from 2500 UAH
depending on complexity."
        },
        {
          "question": "Do I need an appointment?",
          "answer": "Yes, visits are by prior appointment."
        },
        {
          "question": "Can I cancel my appointment?",
           "answer": "Yes, cancellation or rescheduling is possible
at least 12 hours before the visit."
        }
      ]
    }
```

Лістинг коду А.2 - Приклад чтання Python-модулю JSON-файлу і використовувати його як контекст для формування prompt

```
    import json
    from pathlib import Path
    def load_clinic_data(file_path: str) -> dict:
        """Load structured clinic information from a JSON file."""
        path = Path(file_path)
        if not path.exists():
            raise FileNotFoundError(f"File not found: {file_path}")
        with path.open("r", encoding="utf-8") as file:
            return json.load(file)
    def build_prompt(user_message: str, clinic_data: dict) -> str:
         """Build a prompt for the LLM using user input and clinic
context."""
        clinic_name = clinic_data["clinic"]["name"]
```

```
        working_hours = clinic_data["clinic"]["working_hours"]
        services = clinic_data["services"]
        faq = clinic_data["faq"]
        service_list = "\n".join(
            [f"- {service['name']}: {service['price_uah']} UAH" for
service in services]
        )
        faq_list = "\n".join(
            [f"Q: {item['question']}\nA: {item['answer']}" for item
in faq]
        )
        prompt = f"""
    You are an assistant for {clinic_name}.
    Answer the user in Ukrainian.
    Use only the clinic information provided below.
    Do not invent services or prices.
    If the request is unclear, offer a consultation.
    Clinic working hours:
    {working_hours}
    Available services:
    {service_list}
    Frequently asked questions:
    {faq_list}
    User request:
    {user_message}
    """.strip()
        return prompt
    if __name__ == "__main__":
        clinic_data = load_clinic_data("clinic_data.json")
        user_message = "Скільки коштує видалення зуба мудрості?"
        prompt = build_prompt(user_message, clinic_data)
        print("Generated prompt:\n")
        print(prompt)
```

Лістинг коду А.2 - Python-модуль викликає Groq API (LLaMA 3) з уже сформованим prompt

```
    from groq import Groq
    def generate_llm_response(prompt: str) -> str:
        """Send prompt to Groq API and return model response."""
        client = Groq(
            api_key="YOUR_GROQ_API_KEY"  # заміни на свій ключ
        )
        response = client.chat.completions.create(
            model="llama3-70b-8192",
            messages=[
                {
                    "role": "system",
                      "content": "You are a helpful assistant for a
dental clinic. Answer in Ukrainian."
                },
                {
```

```
                "role": "user",
                "content": prompt
            }
        ],
        temperature=0.3,
        max_tokens=512
    )
    return response.choices[0].message.content
if __name__ == "__main__":
    prompt = "Скільки коштує видалення зуба мудрості?"
    result = generate_llm_response(prompt)
    print("LLM response:\n")
    print(result)
```

ДОДАТОК Б

Теза конференції

УДК 621.326
Гнецько В. – ст. гр. СП-41
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЦЕНТРАЛІЗОВАНОГО УПРАВЛІННЯ КОМУНІКАЦІЯМИ СТОМАТОЛОГІЧНОЇ КЛІНІКИ З ВИКОРИСТАННЯМ LLM

Науковий керівник: к.т.н., доцент Михалик Д. М.

Hnetsko V.
Ternopil Ivan Puluj National Technical University

DEVELOPMENT OF SOFTWARE FOR CENTRALIZED COMMUNICATION MANAGEMENT IN A DENTAL CLINIC USING LLM

Supervisor: PhD in Technical Sciences, Associate Professor Mykhalyk D.

Ключові слова: LLM, комунікації, стоматологічна клініка
Keywords: LLM, communication, dental clinic

Ефективне управління комунікаціями у стоматологічній клініці є важливим фактором забезпечення якісного обслуговування пацієнтів. Сучасні клініки взаємодіють із пацієнтами через різні канали, зокрема телефон, електронну пошту, месенджери та веб-форми, що створює потребу у централізованому підході до обробки запитів. Відсутність єдиної системи може призводити до втрати інформації, дублювання записів або затримок у відповідях. У цьому контексті використання великих мовних моделей відкриває нові можливості для автоматизації комунікаційних процесів.

Сучасні LLM дозволяють обробляти текстові запити користувачів, розпізнавати наміри та генерувати релевантні відповіді у природній формі. Це дає змогу автоматизувати типові сценарії, такі як запис на прийом, надання інформації про послуги, нагадування про візити та відповіді на поширені запитання. Інтеграція таких моделей у програмне забезпечення дозволяє значно зменшити навантаження на персонал клініки та підвищити швидкість обробки запитів.

Архітектурно система може бути побудована як багаторівнева платформа, що об’єднує різні канали комунікації у єдиному інтерфейсі. Використання API для інтеграції з месенджерами та CRM-системами дозволяє централізовано зберігати та обробляти інформацію про пацієнтів. LLM у такій системі виступає як інтелектуальний модуль, що аналізує вхідні повідомлення та формує відповіді або відповідні дії.

Окрім цього, важливим є забезпечення контекстності взаємодії, коли система враховує попередні звернення пацієнта, історію записів та індивідуальні особливості. Це дозволяє формувати більш персоналізовані відповіді та покращує користувацький досвід. Використання механізмів пам’яті або інтеграція з базами даних забезпечує збереження та повторне використання інформації.

Експериментальні дослідження показують, що впровадження LLM у системи комунікації дозволяє підвищити швидкість відповіді, зменшити кількість помилок та покращити задоволеність пацієнтів. Основними метриками оцінювання є точність розпізнавання намірів, релевантність відповідей та час обробки запитів [1].

Разом з тим, існують виклики, пов'язані з забезпеченням конфіденційності медичних даних, точністю відповідей та необхідністю контролю якості генерації тексту. Тому актуальними є задачі інтеграції механізмів перевірки відповідей, обмеження доступу до чутливої інформації та адаптації моделей до специфіки медичної галузі [2].

Додатково важливим є впровадження механізмів контролю якості відповідей, що генеруються LLM, особливо у медичному контексті. Система повинна забезпечувати фільтрацію потенційно некоректних або двозначних відповідей, а також можливість передачі складних або критичних запитів безпосередньо оператору. Для цього можуть використовуватись правила валідації, шаблони відповідей або гібридні підходи, де LLM працює разом із заздалегідь визначеними сценаріями. Це дозволяє зберегти баланс між автоматизацією та контролем, що є критично важливим для забезпечення безпеки та довіри з боку пацієнтів.

Окрему увагу слід приділити інтеграції системи з внутрішніми інформаційними ресурсами клініки, такими як електронні медичні записи, системи управління записами пацієнтів та фінансові модулі. Це дозволяє забезпечити повну централізацію комунікацій та уникнути дублювання даних. Крім того, використання аналітичних інструментів дає змогу відстежувати ефективність взаємодії з пацієнтами, визначати типові запити та оптимізувати бізнес-процеси клініки на основі зібраних даних.

Таким чином, поєднання великих мовних моделей, сучасних підходів до обробки даних та інтеграції інформаційних систем створює основу для розробки ефективних платформ управління комунікаціями у медичних установах. Такі системи здатні підвищити якість обслуговування, оптимізувати робочі процеси та забезпечити більш ефективну взаємодію між клінікою та пацієнтами.

Розробка програмного забезпечення централізованого управління комунікаціями стоматологічної клініки з використанням великих мовних моделей дозволяє створити сучасну, ефективну та адаптивну систему взаємодії з пацієнтами. Інтеграція LLM із внутрішніми сервісами клініки забезпечує автоматизацію рутинних процесів, підвищує швидкість обробки запитів та покращує якість обслуговування.

Важливим є не лише впровадження інтелектуальних моделей, а й забезпечення їх коректної роботи в умовах реальної експлуатації, з урахуванням вимог до безпеки, конфіденційності та точності інформації. Поєднання автоматизації з механізмами контролю та можливістю втручання оператора дозволяє досягти балансу між ефективністю та надійністю системи.

У результаті формується комплексна платформа, здатна масштабуватись, адаптуватись до змін потреб клініки та забезпечувати високий рівень сервісу. Такий підхід сприяє оптимізації внутрішніх процесів, зменшенню навантаження на персонал та створює передумови для подальшого розвитку цифрових медичних сервісів.

Література:

1. Brown, T. et al. (2021). Language Models are Few-Shot Learners. — NeurIPS.
2. Bommasani, R. et al. (2022). On the Opportunities and Risks of Foundation Models. — arXiv.
3. Kaddour, J., Harris, J., Mozes, M., et al. (2023). Challenges and Applications of Large Language Models in Real-World Systems. — arXiv.