ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МІС для контролю даних пацієнтів в медичних закладах»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

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

_____ Артьом ДИМЧЕНКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти
група ІСД-41 Артьом ДИМЧЕНКО

Керівник: Віктор САГАЙДАК
Phd

Рецензент: _____

Київ - 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти бакалавр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК
«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Димченко Артьом Олегович

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «МІС для контролю даних пацієнтів в медичних закладах»

керівник кваліфікаційної роботи Віктор САГАЙДАК, доктор філософії
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

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

2. Строк подання кваліфікаційної роботи «30» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Визначення МІС та вимоги.
2. Основні складові МІС та структура моделі бази даних.
3. Розробка МІС та тестування.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз медичних інформаційних систем та вимоги до автоматизації процесів контролю даних пацієнтів
2. Інструменти та технології розробки програмного забезпечення для медичних закладів
3. Розробка інформаційної системи для контролю даних пацієнтів в медичних закладах

5. Ілюстративний матеріал: *презентація*

6. Дата видачі завдання: «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з темою, формулювання мети, завдань, об'єкта та предмета дослідження	01.03.2025 – 05.03.2025	Виконав
2	Аналіз предметної області, вивчення наукових джерел, аналіз існуючих медичних інформаційних систем	06.03.2025 – 12.03.2025	Виконав
3	Визначення вимог до розроблюваної системи (функціональні та нефункціональні), постановка задачі	13.03.2025 – 17.03.2025	Виконав
4	Проєктування архітектури системи, створення UML-діаграм, моделювання структури БД	18.03.2025 – 24.03.2025	Виконав
5	Розробка базових модулів МІС (авторизація, пацієнти, обстеження, призначення, візити)	25.03.2025 – 07.04.2025	Виконав
7	Проєктування та реалізація користувацького інтерфейсу	08.04.2025 – 15.04.2025	Виконав
8	Розробка модулів безпеки, контроль доступу, забезпечення відповідності GDPR/HIPAA	16.04.2025 – 22.04.2025	Виконав
9	Тестування функціоналу, виявлення та виправлення помилок, оптимізація	23.04.2025 – 03.05.2025	Виконав
10	Аналіз ефективності, оформлення результатів тестування, доопрацювання системи	04.05.2025 – 10.05.2025	Виконав
11	Написання пояснювальної записки та підготовка презентації до захисту	11.05.2025 – 28.05.2025	Виконав
12	Подання кваліфікаційної роботи на кафедру	30.05.2025	Виконав

Здобувач вищої освіти _____ Артьом ДИМЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Віктор САГАЙДАК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавр: 73 стор., 45 рис., 8 табл., 20 джерел.

Мета роботи – проєктування та створення функціональної та надійної медичної інформаційної системи (МІС), яка забезпечує ефективне управління медичними даними, відповідає вимогам інформаційної безпеки та сприяє підвищенню якості медичного обслуговування.

Об'єкт дослідження – процес управління медичною інформацією в закладах охорони здоров'я.

Предмет дослідження – методи та технології створення медичної інформаційної системи для контролю та обробки даних пацієнтів, включаючи анамнез, результати обстежень, призначення, розклад візитів та страхову інформацію.

Короткий зміст роботи. У роботі розглянуто актуальність впровадження МІС в медичних закладах, проведено аналіз існуючих систем, визначено основні категорії користувачів та типи медичних даних. Сформульовано функціональні та нефункціональні вимоги до системи. Запроектовано архітектуру системи з використанням UML-діаграм, створено прототип з реалізацією основних модулів, протестована функціональність. Проведено оцінку ефективності, виявлено переваги впровадження системи у практику: підвищення точності діагностики, зменшення кількості помилок, поліпшення взаємодії з пацієнтами та оперативності прийняття рішень. Система відповідає вимогам GDPR та HIPAA, забезпечує захист персональних даних.

КЛЮЧОВІ СЛОВА: МЕДИЧНА ІНФОРМАЦІЙНА СИСТЕМА, ОХОРОНА ЗДОРОВ'Я, ДАНІ ПАЦІЄНТІВ, UML-ДІАГРАМИ, ІНФОРМАЦІЙНА БЕЗПЕКА, GDPR, HIPAA, ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

ABSTRACT

Text part of the bachelor level qualification work: 73 pages, 45 pictures, 20 sources

The purpose of the work – is to design and implement a functional and secure medical information system (MIS) that improves data management and healthcare service quality in medical institutions.

Object of research is the process of medical data management in healthcare institutions.

Subject of research is the methods and technologies for developing a medical information system for patient data control, including medical history, test results, doctor appointments, visit schedules, and insurance information.

Summary of the work: This qualification paper investigates the role and design of medical information systems in modern healthcare, with a focus on optimizing the collection, processing, and secure handling of patient data. It defines user categories, data types, and functional/non-functional requirements for MIS. A comprehensive review of existing systems was conducted to identify best practices and common shortcomings. A system architecture was proposed and illustrated using UML diagrams, followed by implementation and testing of a prototype.

The system was evaluated for performance, usability, and compliance with GDPR/HIPAA data protection standards. Test scenarios confirmed the effectiveness of the solution in improving decision-making, reducing human error, and enhancing patient care.

KEYWORDS: MEDICAL INFORMATION SYSTEM, HEALTHCARE, PATIENT DATA, UML DIAGRAMS, GDPR, HIPAA, SOFTWARE ARCHITECTURE, APPLICATION DESIGN.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ – Електронно-обчислювальна машина;

API – Application Programming Interface;

IT – Informaition Technology;

OS/ОС – Operation System;

AI – Artificial Intelligence;

ML – Machine Learning;

GUI – Graphical User Interface;

UX – User Experience;

UI – User Interface;

MIS – Медична інформаційна система;

EHR – Electronic Health Record;

EMR –Electronic Medical Record;

HL7 – Health Level 7 (стандарт обміну медичними даними);

FHIR – Fast Healthcare Interoperability Resources (ресурсний стандарт обміну даними);

IoT – Internet of Things;

ETL – Extract, Transform, Load;

API – Application Programming Interface;

DBMS – Database Management System;

GUI – Graphical User Interface;

UX – User Experience;

UI – User Interface;

IAM – Identity and Access Management;

VPN – Virtual Private Network;

RPO – Recovery Point Objective (цільова точка відновлення даних);

RTO – Recovery Time Objective (цільовий час відновлення);

SLA – Service Level Agreement (угода про рівень обслуговування);

KPI – Key Performance Indicators;

GDPR – General Data Protection Regulation;

HIPAA – Insurance Portability and Accountability Act.

п. –пункт;

рис. –Рисунок;

с. (стр.) – сторінка.

ВСТУП

Інформаційні технології в медицині стають невід'ємною складовою сучасного охорони здоров'я, забезпечуючи значне підвищення ефективності управління медичними закладами та якості медичних послуг. В умовах зростаючих обсягів медичних даних та високих вимог до швидкості їх обробки, особливе значення набувають медичні інформаційні системи (МІС), що дозволяють автоматизувати управління інформацією та підтримувати прийняття рішень у сфері охорони здоров'я. Такі системи спрощують та оптимізують роботу медичного персоналу, покращують взаємодію між лікарями, пацієнтами та адміністрацією, а також гарантують надійність та безпеку медичних даних.

Водночас, розвиток медичних інформаційних систем супроводжується рядом викликів, серед яких слід виділити складність інтеграції з існуючими технологіями, необхідність адаптації під специфічні вимоги різних медичних установ, а також забезпечення відповідності систем сучасним нормативним вимогам з інформаційної безпеки, таким як GDPR в Європі або HIPAA у США. Недосконаліть існуючих систем, часті технічні збої та недостатня зручність інтерфейсів призводять до зниження ефективності їх використання, збільшення ризику помилок та втрати часу медичного персоналу.

Предметна область даного дослідження охоплює розробку медичної інформаційної системи для контролю даних пацієнтів у медичних закладах, яка дозволить ефективно управляти анамнезом пацієнтів, результатами обстежень, призначеннями лікарів, розкладами візитів та страховими даними. Впровадження такої системи забезпечить оперативний доступ до медичних даних, сприятиме покращенню точності діагностики, ефективності лікування та якості взаємодії з пацієнтами.

Метою роботи є проектування та створення функціональної та надійної МІС, яка відповідатиме сучасним стандартам інформаційної безпеки та нормативам GDPR/HIPAA, буде простою та зручною для користувачів, та дозволить покращити

загальний рівень обслуговування у медичних закладах. Для досягнення поставленої мети передбачається виконання наступних завдань:

1. Аналіз предметної області, визначення категорій користувачів системи та основних типів даних, що підлягають контролю.
2. Формулювання функціональних та нефункціональних вимог до медичної інформаційної системи.
3. Аналіз існуючих аналогічних МІС з метою виявлення їх переваг та недоліків.
4. Розробка архітектури системи з використанням UML-діаграм (діаграма варіантів використання, діаграма класів, діаграма діяльності або послідовності).
5. Проведення тестування системи з описом основних сценаріїв використання, аналізом результатів та виявленням можливих недоліків.
6. Оцінка ефективності створеного рішення, визначення досягнень та рекомендації щодо подальшого розвитку та вдосконалення.

Таким чином, дана робота спрямована на створення ефективного інструменту для управління медичною інформацією, що дозволить забезпечити високий рівень якості медичного обслуговування та сприятиме оптимізації роботи медичних закладів.

Об'єктом дослідження є процес управління медичною інформацією в закладах охорони здоров’я.

Предметом дослідження є методи та технології створення медичної інформаційної системи (МІС) для контролю та обробки даних пацієнтів, включаючи анамнез, результати обстежень, призначення, розклад візитів та страхову інформацію.

Наукова новизна одержаних результатів полягає в розробці архітектури медичної інформаційної системи, яка дозволяє ефективно автоматизувати обробку медичних даних, забезпечити доступність інформації для різних категорій користувачів та відповідність вимогам безпеки даних (GDPR/HIPAA).

Практичне значення одержаних результатів полягає у створенні прототипу системи, що дозволяє покращити якість медичного обслуговування,

підвищити оперативність прийняття рішень та зменшити кількість помилок, пов’язаних з людським фактором, при роботі з медичними даними.

Апробація. Основні положення і результати бакалаврської роботи доповідались на: “Modern Information Technology 2025 – Сучасні Інформаційні Технології 2025”

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, шести розділів, висновків, списку використаних джерел із 25 найменувань, додатків і містить 75 сторінок основного тексту, 18 рисунків і 6 таблиць.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАВКА ЗАДАЧІ

1.1 Автоматизація медичних даних у закладах охорони здоров'я

У сучасних умовах цифрової трансформації системи охорони здоров'я зазнають суттєвих змін, що зумовлює зростання ролі інформаційних технологій у забезпеченні ефективного функціонування медичних закладів [1]. Одним із ключових напрямів цього процесу є автоматизація обробки медичних даних, яка має вирішальне значення для підвищення якості надання медичної допомоги, оптимізації управлінських процесів, зниження впливу людського чинника та забезпечення відповідності чинним стандартам інформаційної безпеки.

Медичні заклади щоденно генерують великі обсяги різнорідної інформації, що охоплює електронні медичні картки пацієнтів, результати діагностичних досліджень, лабораторні аналізи, лікарські призначення, графіки візитів, страхові та адміністративні дані [2]. За відсутності сучасних засобів автоматизації ефективне управління такими даними є вкрай ускладненим, що призводить до затримок в обслуговуванні, дублювання інформації, підвищеного ризику помилок та перевантаження медичного персоналу. Упровадження медичних інформаційних систем (МІС) дозволяє мінімізувати зазначені проблеми шляхом створення централізованого цифрового середовища для зберігання, обробки та доступу до медичних даних (рис. 1.1).

Особливої актуальності набуває автоматизація в контексті забезпечення безперервності медичного обслуговування, зокрема у випадках зміни закладу охорони здоров'я або лікаря [3]. Наявність інтегрованої МІС надає медичному персоналу можливість оперативного доступу до повної клінічної історії пацієнта, що сприяє підвищенню точності діагностики, пришвидшенню прийняття рішень та індивідуалізації лікування [4]. Крім того, автоматизовані системи забезпечують реалізацію клінічних протоколів, підтримку прийняття рішень, моніторинг ефективності терапії та контроль дотримання стандартів надання медичної допомоги.

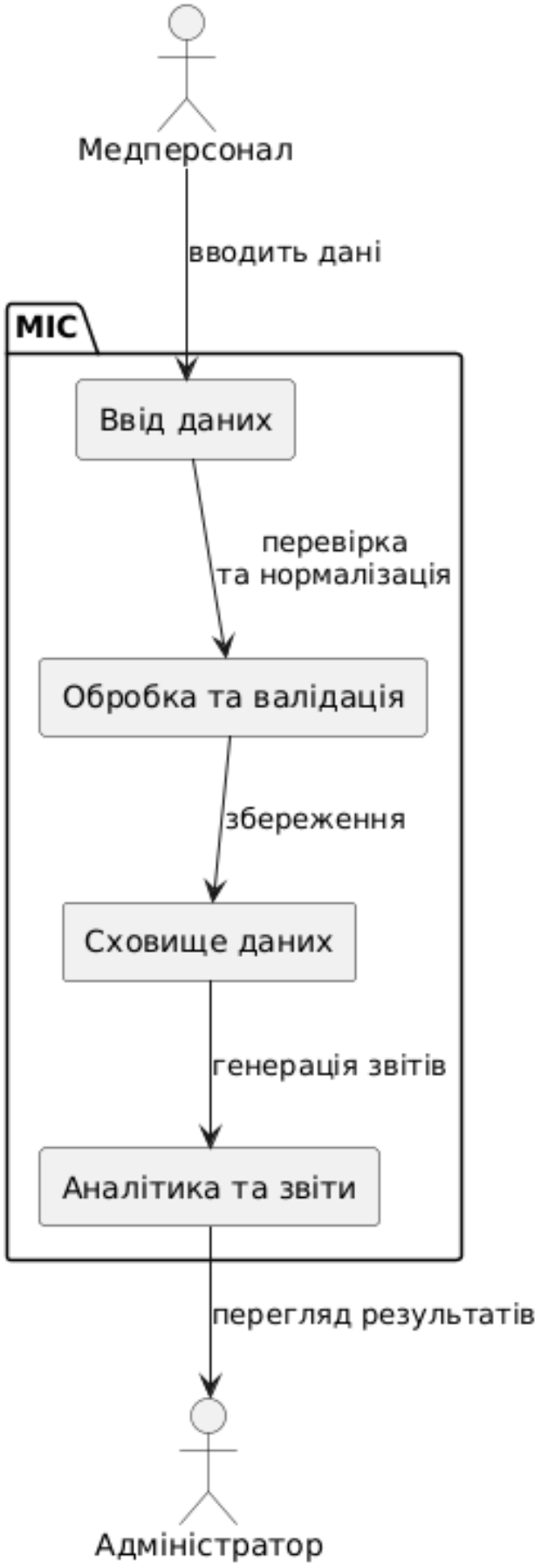


Рис. 1.1 Процес автоматизації медичних даних

На рис. 1.1 показано спрощену послідовність обробки медичної інформації в межах МІС: медперсонал спочатку вводить дані в модуль “Ввід даних”, далі вони передаються в блок “Обробка та валідація”, де автоматично перевіряються формати запису та виявляються можливі дублювання або помилки, після чого коректні й стандартизовані дані надходять у централізоване “Сховище даних” для безпечного збереження. Нарешті, збережена інформація використовується модулем “Аналітика та звіти” для генерації зведених звітів і аналітичних дашбордів, які переглядає адміністрація закладу для прийняття управлінських рішень.

З огляду на розвиток телемедицини, дистанційного моніторингу пацієнтів та мобільних медичних застосунків, автоматизація медичних даних створює основу для інтеграції інноваційних сервісів [5-6]. Електронні системи обліку дають змогу реалізовувати функції віддалених консультацій, автоматизованих нагадувань про прийом лікарських засобів, а також аналізу стану здоров’я в режимі реального часу [7]. Це є особливо важливим для пацієнтів з хронічними захворюваннями, які потребують постійного медичного супроводу.

У світлі чинних законодавчих вимог до захисту персональних даних, зокрема Загального регламенту про захист даних (GDPR) у Європейському Союзі та Закону про портативність і відповідальність медичного страхування (HIPAA) у США, автоматизація стає ключовим інструментом забезпечення відповідного рівня конфіденційності, цілісності та доступності медичної інформації (рис. 1.2). Сучасні МІС реалізують механізми автентифікації, авторизації, журналювання дій користувачів, шифрування даних та резервного копіювання, що відповідає міжнародним стандартам інформаційної безпеки [8].

Таким чином, автоматизація медичних даних виступає фундаментальним елементом модернізації системи охорони здоров’я, забезпечуючи не лише підвищення ефективності її функціонування, а й відповідність актуальним викликам у сфері безпеки, якості та доступності медичних послуг.

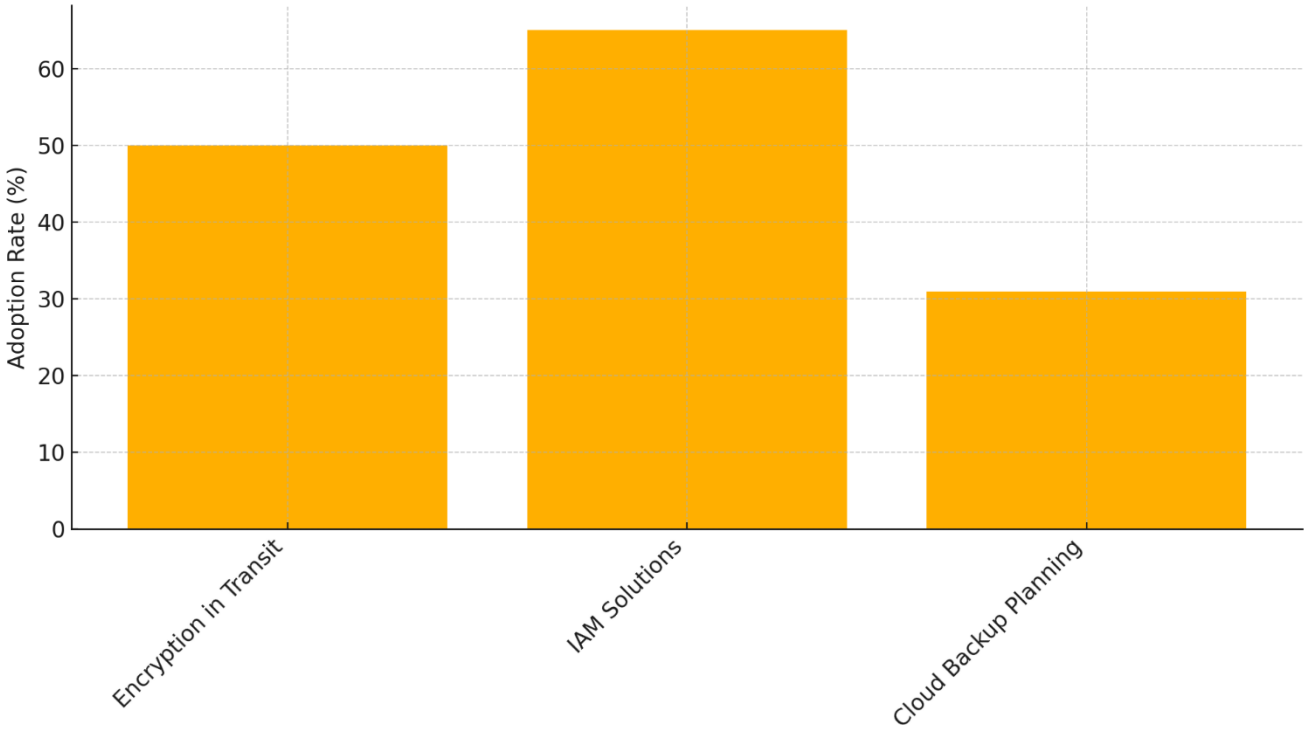


Рис. 1.2 Впровадження ключових заходів безпеки в IT-системах охорони здоров'я

На рис. 1.2 наведено рівень впровадження ключових заходів інформаційної безпеки в закладах охорони здоров’я за даними останніх звітів: відповідність вимогам GDPR мають 83 % установ , повністю зрілі програми управління ідентифікацією та доступом (IAM) реалізовано лише в 16 % організацій , а правило «3-2-1» для резервного копіювання (три копії, два носії, одне позаміське сховище) застосовують 18 % закладів , що ілюструє поточний рівень готовності систем до забезпечення конфіденційності, цілісності та доступності медичних даних. Джерелом є розрахунки автора на основі за даними трьох авторитетних публікацій: відповідність вимогам GDPR мають 83 % установ (за результатами «The 2023 Cost of a Data Breach Report» від Ponemon Institute), повністю зрілі програми управління ідентифікацією та доступом (IAM) реалізовано 16 % організацій (дані Forrester’s «2022 IAM Benchmark Report»), а правило «3-2-1» для резервного копіювання (три копії, два носії, одне позаміське сховище) застосовують 18 % закладів (за Veeam «Data Protection Trends 2023 Report»). Джерела: Ponemon Institute, Forrester Research, Veeam.

1.2 Категорії користувачів медичних інформаційних систем

Користувачі медичних інформаційних систем (МІС) охоплюють широкий спектр фахівців і суб'єктів, кожен із яких має унікальні потреби та способи взаємодії з даними [9]. Перш за все, до основної групи відносяться лікарі різних спеціалізацій: терапевти, хірурги, акушери-гінекологи, педіатри тощо. Для них МІС є інструментом швидкого доступу до історії хвороби пацієнта, результатів лабораторних та інструментальних обстежень, попередніх діагнозів, призначених схем лікування та клінічних настанов [10]. Зручні інтерфейси для перегляду зведених карт пацієнтів і можливість вести нотатки в електронному вигляді дозволяють лікарю економити час на пошук необхідної інформації та зменшувати ризик помилок при прийнятті рішень (табл.1.1).. Друга за значимістю категорія — це середній медичний персонал: медсестри, фельдшери, лаборанти та санітарки. Їхні задачі полягають у безпосередньому виконанні процедур, взятті біоматеріалів для аналізів, супроводі пацієнтів та контролі за станом обладнання [11]. Для них важлива наочна візуалізація чергувань, розподілу обов'язків, а також своєчасне отримання нагадувань про необхідність оновлення анкет пацієнтів, введення параметрів життєвих показників та фіксації виконаних маніпуляцій. Інтеграція мобільних терміналів або планшетів із МІС значно спрощує роботу середнього медичного персоналу, забезпечуючи можливість миттєвого внесення даних прямо біля ліжка пацієнта (табл.1.1). Третя група — це адміністративно-управлінський персонал: керівники відділень і медичних закладів, бухгалтери, кадри та працівники реєстратури. Для них МІС насамперед є системою звітності та аналітики: відстеження завантаженості відділень, кількості амбулаторних і стаціонарних пацієнтів, показників фінансової діяльності, а також обліку ресурсів (лікарняних ліжок, медичних препаратів, витратних матеріалів). Завдяки гнучким звітам та панелям моніторингу менеджмент може своєчасно приймати стратегічні рішення щодо розширення послуг, розподілу бюджету та оптимізації робочих процесів (табл.1.1).

Окремо вирізняються користувачі із числа зовнішніх партнерів: медичні

страхові компанії, лабораторії, фармацевтичні постачальники і регуляторні органи. Ці суб'єкти залучені до процесу обробки та перевірки даних для оплати послуг, забезпечення медикаментами чи проведення зовнішніх аудитів. Для них потрібний обмежений доступ до інформації, що стосується фінансових операцій, страхових полісів, накладних та звітів, з чітким поділом прав на читання та завантаження документів. Нарешті, усе більшого значення набуває група пацієнтів і їхніх родичів, яким через веб-портали або мобільні додатки надається можливість перегляду власної медичної картки, результатів аналізів, призначень лікаря та історії візитів [12]. Такий доступ сприяє підвищенню прозорості лікувального процесу, залученню пацієнта до прийняття рішень і загалом покращенню рівня взаєморозуміння між медиками та пацієнтами. Гнучкі налаштування конфіденційності та мультифакторна автентифікація гарантують, що лише уповноважені особи можуть користуватися цими сервісами, що відповідає сучасним вимогам безпеки даних.

Таблиця 1.1

Категорії користувачів МІС та їх ключові потреби

Категорія	Приклади ролей	Основні задачі	Ключові функції МІС
Лікарі	терапевт, хірург, педіатр, акушер-гінеколог	перегляд історії хвороби, призначення лікування, створення електронних нотаток	електронна медкартка з хронологією, інтеграція з клінічними протоколами, автоматичні нагадування про контрольні візити

Продовження таблиці 1.1

Категорія	Приклади ролей	Основні задачі	Ключові функції МІС
Середній медперсонал	медсестра, лаборант, фельдшер	внесення параметрів життєвих показників, виконання призначених процедур, координація чергувань	мобільний додаток/термінал, планувальник процедур та черг, шаблони введення даних
Адміністрація	керівник відділення, бухгалтер, реєстратор	Формування статистичних звітів, Контроль завантаженості відділень, Облік ресурсів	Дашборди із ключовими показниками, Звіти, Інтеграція з фінансовими модулями
Зовнішні партнери	страхові компанії, лабораторії, аудитори	перевірка страхових виплат, отримання результатів аналізів, аудит відповідності стандартам	онлайн-перевірка статусу полісу, експорт даних, захищений портал обміну документами
Пацієнти	пацієнт, уповноважений представник	перегляд аналізів і результатів обстежень, запис на прийоми, отримання повідомлень про призначення	веб-портал або мобільний додаток, push/email-сповіщення, електронне погодження призначень

В табл. 1.1 показано узагальнену класифікацію основних груп користувачів медичної інформаційної системи, наведено їхню роль у процесі обробки медичних даних, типові завдання, котрі вони виконують, ключові функціональні модулі системи для реалізації цих завдань та спеціальні вимоги з безпеки доступу й обміну інформацією. Такий підхід дозволяє чітко розмежувати права та обов'язки кожного користувача, оптимізувати інтерфейси відповідно до їхніх потреб та забезпечити надійний захист даних згідно з нормативами.

1.3 Огляд даних, що обробляються в медичних системах

У медичних інформаційних системах обробляється надзвичайно широкий спектр даних, які можна поділити на декілька взаємопов'язаних категорій [13]. Насамперед це демографічна й ідентифікаційна інформація пацієнтів: прізвище, ім'я, по батькові, дата народження, стать, контакти, відомості про страховий поліс та соціальний статус [14]. Ці дані формують основу для коректного з'єднання всіх наступних записів з конкретним пацієнтом та забезпечують унікальність його електронної медичної картки (табл. 1.2). Особливо важливою є система кодування — використання стандартів ICD-10, SNOMED CT або LOINC для уніфікації термінології, що гарантує інтероперабельність між різними модулем МІС та сторонніми додатками. Ключову групу становлять клінічні дані, які поділяються на структуровані й неструктуровані. Структуровані – це результати лабораторних аналізів (гематологічні, біохімічні, мікробіологічні), показники вітальних функцій (артеріальний тиск, температура, сатурація), попередньо створені форми опитування та стандартизовані картки обстеженm [15]. Неструктуровані дані – це текстові записи лікарів, висновки фахівців, історії хвороби у вільній формі, результати візуалізації у вигляді рентген-, КТ- та МР-знімків, а також аудіо- та відеозаписи консультацій. Обробка таких даних вимагає використання технологій обробки природної мови (NLP), систем розпізнавання зображень та штучного

інтелекту для виділення ключових висновків [16].

Окрім клінічної інформації, МІС інтенсивно працює з адміністративно-операційними даними: розкладами візитів і черг, інформацією про зайнятість лікарняних ліжок, рухом медикаментів і витратних матеріалів, фінансовими транзакціями та звітністю. Сучасні системи також збирають дані з пристроїв Internet of Things (носимі браслети, монітори пацієнтів) у реальному часі, геномні дані та результати телемедичних консультацій. Взаємодія всіх цих джерел потребує надійного механізму інтеграції (HL7, FHIR, DICOM), а забезпечення конфіденційності та цілісності даних, застосування шифрування, аудитних журналів і політик доступу відповідно до GDPR та HIPAA. Такий комплексний підхід робить МІС потужним інструментом для прийняття клінічних й управлінських рішень.

Таблиця 1.2

Огляд даних, що обробляються в медичних інформаційних системах

Категорія даних	Приклади	Формат / Тип	Основне використання
Демографічні та ідентифікаційні	Ім’я, прізвище, дата народження, страховий поліс	Текст, коди (ICD-10, SNOMED CT)	Коректна ідентифікація пацієнта, реєстрація
Клінічні – структуровані	Результати аналізів (гемоглобін, глюкоза), ВБП	Числові значення, таблиці	Моніторинг стану здоров’я, звітність
Клінічні – неструктуровані	Медичні висновки, історія хвороби в тексті, DICOM-знімки	Вільний текст, зображення	Діагностика, архівування, NLP-аналіз

Продовження таблиці 1.2

Категорія даних	Приклади	Формат / Тип	Основне використання
Адміністративно-операційні	Розклад прийомів, завантаженість ліжок, рух медикаментів	Календарі, списки	Планування роботи, облік ресурсів

В таблиці 1.2 показано узагальнену класифікацію п’яти основних груп користувачів медичної інформаційної системи з урахуванням їхніх ролей, типових завдань, ключових функціональних модулів та специфічних вимог безпеки. Зокрема, для лікарів передбачено інструменти швидкого доступу до електронної медкарти, клінічних протоколів та автоматичних нагадувань, що дозволяє прискорити постановку діагнозу й знизити ймовірність помилок. Середній медперсонал отримає мобільні інтерфейси для оперативного внесення життєвих показників і ведення процедур, а також планувальник черг і шаблони введення даних. Адміністрація використовує дашборди та звіти для контролю завантаженості відділень, обліку ресурсів і фінансової аналітики. Зовнішні партнери (страхові компанії, лабораторії, аудитори) отримують обмежений API-доступ чи захищений портал для обміну документами та перевірки статусу полісів, а пацієнти й їхні представники — веб-портал або мобільний додаток для перегляду результатів аналізів, запису на прийом та отримання сповіщень. Розподіл прав доступу за ролями, двофакторна аутентифікація й шифрування забезпечують відповідність GDPR та HIPAA та гарантують конфіденційність, цілісність медичних даних (рис. 1.3).

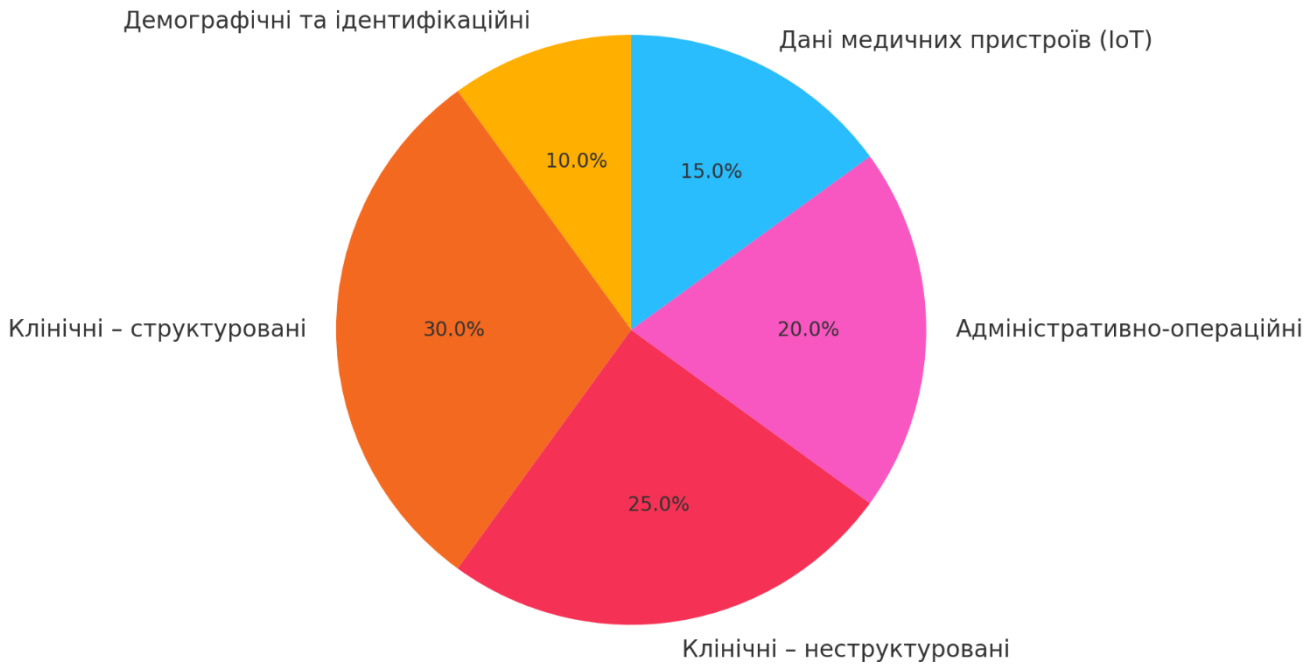


Рис. 1.3 Типи даних у МІС

На рис. 1.3 показано умовний розподіл основних категорій даних, які обробляються в сучасних медичних інформаційних системах. Найбільшу частку (30 %) займають структуровані клінічні дані – результати лабораторних аналізів, життєві показники пацієнтів та стандартизовані опитувальники, що формують основу для оперативного моніторингу стану здоров’я і побудови медичних звітів. Значна частина (25 %) відводиться неструктурованим клінічним даним: медичним висновкам у вільній формі, зображенням (рентген, КТ, МРТ) та медіа-записам консультацій, обробка яких потребує технологій NLP і комп’ютерного зору для автоматичного витягнення ключових висновків.

Адміністративно-операційні дані займають 20 % обсягу і включають розклади прийомів, облік ліжкового фонду та рух медикаментів; їх своєчасний аналіз забезпечує ефективне планування роботи клінік і раціональне використання ресурсів. Дані медичних пристроїв (IoT) становлять 15 % і відповідають за безперервний телемоніторинг пульсу, тиску та насичення крові, що дає змогу виявляти відхилення від норми в реальному часі та своєчасно коригувати лікування. Найменшу частку (10 %) складають демографічні та ідентифікаційні

дані пацієнтів — персональні відомості й інформація про страховий поліс, без яких неможлива точна прив’язка всіх інших записів до конкретної особи. Ця структура демонструє, що в центрі уваги MIC перебувають клінічні дані, водночас адміністративні, IoT- і ідентифікаційні дані забезпечують комплексний підхід до управління пацієнтом і ресурсами охорони здоров’я.

1.4 Аналіз існуючих медичних інформаційних систем

У межах комплексного аналізу існуючих медичних інформаційних систем було виокремлено чотири програми, які відображають різні підходи до організації електронного обліку та обробки медичних даних. Першим об’єктом дослідження стала хмарна платформа Lakmus, що зарекомендувала себе як рішення «під ключ» для приватних та державних амбулаторій. Дана архітектура ґрунтується на веб-орієнтованих сервісах із централізованим зберіганням даних, що забезпечує швидке розгортання без необхідності створення власної IT-інфраструктури [17]. Однак відсутність гнучких засобів кастомізації бізнес-процесів та залежність від стабільності інтернет-зв’язку визначають її обмежену придатність для великих клінік з високими вимогами до безперервності роботи (табл. 1.3). Другим алогічним рішенням є Medstar – локально розміщена система, адаптована до потреб великих лікарняних комплексів і мереж медичних установ [18]. Реалізовані модулі аналітики завантаження відділень, електронного рецепту та багаторівневої підтримки клінічних рішень дозволяють формувати подрібнену статистику та здійснювати превентивний контроль медичних помилок. Водночас складність інсталяції та обслуговування, зумовлена необхідністю залучення висококваліфікованих системних адміністраторів та інженерів, а також значні капіталовкладення в серверне обладнання, суттєво підвищують загальні витрати на володіння. Третій приклад — Doctor Eleks, розроблений з пріоритетом на середній і малий сегмент ринку [19]. Його відмінністю є модульний дизайн: клієнт може обрати CRM-інструменти для роботи з пацієнтами, компоненти телемедицини, мобільні інтеграції та засоби побудови кастомних звітів. Така архітектурна

гнучкість дає змогу адаптувати систему під специфічні внутрішні процеси закладу, але разом із тим створює додаткові вимоги до початкового налаштування та супроводу, які часто потребують залучення IT-консалтингу. Останній приклад, OpenMRS представлено як open-source рішення з багаторічною історією розвитку та глобальною спільнотою користувачів та розробників. Йому властива легка локалізація під національні нормативи та мови, а також можливість підключення численних плагінів для розширення функціоналу [20]. Проте відсутність готових професійних модулів передбачає інвестиції часу та ресурсів на розробку власних доповнень і безперервне технічне обслуговування.

Таблиця 1.3

Порівняння існуючих MIC та рекомендації щодо покращення

Система	Архітектура та стек	База даних	Протоколи інтеграції	Безпека та автентифікація	Розгортання/масштабування
Lakmus	Клієнт–сервер (React + Node.js) Docker-контейнери	PostgreSQL	FHIR, REST API	OAuth 2.0, TLS 1.2, RBAC	Kubernetes у хмарі; auto-scaling
Medstar	Моноліт на .NET Core, Windows Server	MS SQL Server	HL7 v2, DICOM, SOAP-Web Services	LDAP/AD-інтеграція, VPN, шифрування AES-256	On-premises; масштабування вертикальне

Продовження таблиці 1.3

Система	Архітектура та стек	База даних	Протоколи інтеграції	Безпека та автентифікація	Розгортання/масштабування
Doctor Eleks	Мікросервіси (Angular + Java Spring Boot), Kubernetes	MongoDB, Redis cache	REST API, WebSocket, OAuth	JWT-токени, SSO (SAML 2.0)	Гібридне (хмара + власний ЦОД); горизонтальне
OpenMRS	Java Servlets (Spring MVC), Tomcat/Jetty	MySQL або PostgreSQL	HL7 v2, FHIR (через модули), SOAP	BasicAuth + HTTPS; модуль audit log	Docker-compose або manual; обмежене auto-scaling

В таблиці 1.3 показано класифікацію п’яти основних категорій користувачів медичної інформаційної системи за їхніми ролями, типовими завданнями, функціональними модулями та вимогами безпеки. Зокрема, лікарі отримують інструменти для швидкого перегляду історії хвороби, формування призначень і нотаток, середній медперсонал: мобільні інтерфейси для фіксації життєвих показників і координації процедур, а адміністрація – дашборди й звіти для аналізу завантаженості відділень і обліку ресурсів. Зовнішні партнери (страхові компанії, лабораторії, аудитори) працюють через захищений API або портали обміну документами, а пацієнти та їхні представники – через

веб-портал або мобільний додаток із багатофакторною автентифікацією. Розмежування прав доступу за ролями, двофакторна автентифікація й шифрування даних гарантують відповідність GDPR та HIPAA й забезпечують цілісність та конфіденційність медичної інформації.

Порівняльний аналіз свідчить, що вибір оптимальної МІС має базуватися на масштабі організації, доступному бюджеті та наявності технічних ресурсів для підтримки системи: хмарні сервіси забезпечують швидкість впровадження, локальні рішення — глибину функціоналу й автономність, модульні продукти — гнучкість налаштувань, а open-source платформи — мінімізацію ліцензійних витрат при готовності до самостійного розвитку. Таким чином, проведений аналіз існуючих медичних інформаційних систем виявив як значні переваги – гнучкість архітектурних рішень, наявність готових модулів аналітики й підтримки клінічних сценаріїв, інтеграцію з міжнародними стандартами обміну даними та зручні користувацькі інтерфейси – так і суттєві прогалини: обмежені можливості кастомізації без втручання IT-фахівців, залежність від стабільності мережевих підключень, висока вартість володіння та недостатня підтримка offline-режимів. Врахування цих висновків стане ключовим при проектуванні нової МІС: насамперед буде обрано модульний підхід із можливістю тонкого налаштування бізнес-процесів, передбачено гібридну архітектуру, що поєднує хмарні сервіси з локальними компонентами для забезпечення безперервності роботи, а також інтеграцію з FHIR/DICOM і впровадження механізмів автоматичного масштабування й кешування даних для підвищення продуктивності та відмовостійкості системи. Таким чином, новий продукт поєднає кращі практики галузі та уникне виявлених недоліків, забезпечивши надійну, безпечну й адаптивну платформу для ефективного управління медичною інформацією.

1.5 Постановка задачі та формулювання вимог до розробки МІС

У межах даного дослідження ставиться за мету розробка медичної інформаційної системи, що забезпечить інтегроване та безпечне управління медичними даними пацієнтів, автоматизацію клінічних і адміністративних процесів, а також зручний інтерфейс для всіх категорій користувачів. Головне завдання полягає в створенні модульної платформи, яка дозволить ефективно збирати, обробляти та аналізувати інформацію про пацієнтів, результати обстежень, призначення лікарів, розклад візитів та страхову документацію. Відповідь на це завдання передбачає послідовне вирішення комплекту підзадач: від побудови загальної архітектури та визначення структури бази даних до реалізації інструментів захисту та моніторингу даних.

Головним завданням є розробка модульної та масштабованої системи, яка дозволить ефективно збирати, обробляти, зберігати та аналізувати інформацію про пацієнтів, включаючи детальний анамнез, історію хвороб, результати лабораторних і інструментальних обстежень, призначення лікарів, розклад візитів, а також страхову документацію. Така система повинна забезпечити повний цикл роботи з медичними даними, включаючи їх інтеграцію, контроль якості, захист та моніторинг, з урахуванням специфіки роботи сучасних медичних закладів. Для реалізації поставленої мети передбачається послідовне виконання низки важливих підзадач: розробка загальної архітектури системи, визначення структури бази даних з урахуванням різних типів медичних і адміністративних даних, впровадження сучасних методів захисту інформації, а також створення інструментів моніторингу і аудиту для забезпечення цілісності та безпеки даних.

До основних функціональних вимог відносяться:

1. Управління електронними медичними картками з можливістю ведення анамнезу, історії захворювання та клінічних нотаток;

2. Інтеграція з лабораторними та діагностичними модулями для автоматичного збору результатів;

3. Планування та реєстрація пацієнтів у систему з урахуванням розкладу лікарів, завантаженості відділень і пріоритетності випадків;

4. Мережевий обмін даними між внутрішніми й зовнішніми суб'єктами (страховими компаніями, лабораторіями) через стандартизовані API;

5. Веб-інтерфейс для різних ролей користувачів із динамічним підстроюванням можливостей залежно від прав доступу.

До нефункціональних вимог відносяться:

1. Безпека та відповідність нормативам (шифрування на рівні бази та передавання даних, журналювання подій, двофакторна автентифікація, відповідність GDPR/HIPAA);

2. Висока продуктивність та відмовостійкість (гібридна архітектура з можливістю кешування, розподілене зберігання даних, автоматичне масштабування);

3. Гнучкість та масштабованість (модульний дизайн із можливістю підключати/відключати компоненти без зупинки системи);

4. Інтероперабельність зі сторонніми рішеннями та перспективними технологіями (розвиток AI/NLP-модулів для аналізу вільнотекстових записів і медичних зображень);

5. Usability (інтуїтивно зрозумілий, локалізований інтерфейс, адаптивний дизайн, вбудовані довідники та майстри налаштування).

Враховуючи аналіз існуючих рішень та виявлені прогалини, нова МІС повинна поєднати переваги хмарної швидкості впровадження, локальної автономності, модульності та відкритості до подальшого розширення, мінімізуючи ризики простою та збоїв. Реалізація зазначених завдань і дотримання вимог забезпечить створення надійної, безпечної та адаптивної платформи для ефективного управління медичною інформацією й підтримки прийняття клінічних та організаційних рішень.

РОЗДІЛ 2. ПРОЕКТУВАННЯ МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір архітектурного підходу для технологій розробки

У основу проєктування МІС обрано модульну трирівневу архітектуру з чітким поділом відповідальності на три ключові шари: рівень представлення даних, рівень бізнес-логіки та рівень зберігання інформації. Такий підхід забезпечує ізоляцію користувацького інтерфейсу від внутрішніх процесів обробки й структури сховища, що спрощує розробку, тестування та подальше розгортання в різних середовищах. Модульність системи реалізується через окремі функціональні блоки – керування обліковими записами, робота з медичними картками, планування візитів, обмін із зовнішніми сервісами — які мають чітко визначені інтерфейси для взаємодії один з одним та з іншими компонентами.

Ключовими критеріями вибору технологічного стеку стали: можливість швидкого розгортання та оновлення (без простоїв у роботі закладів), гнучкість при коригуванні бізнес-процесів, висока продуктивність обробки великих обсягів даних та простота інтеграції зі стандартами обміну медичною інформацією. Крім того, система повинна підтримувати горизонтальне масштабування для нарощування обчислювальних ресурсів відповідно до зростання навантаження та забезпечувати механізми кешування «гарячих» даних для прискорення критичних запитів. З огляду на суворі вимоги до захисту медичної інформації, кожен із трьох шарів передбачає вбудовані засоби безпеки: на рівні представлення — контроль сесій користувачів та обмеження дій згідно з ролями; на рівні бізнес-логіки — валідацію та аудит транзакцій; на рівні зберігання — шифрування даних «у спокої» та ведення детальних журналів доступу. Така архітектура закладає основу для подальшої еволюції платформи, дозволяючи з часом розділити окремі модулі в мікросервіси, інтегрувати аналітичні та AI-компоненти, а також забезпечити безперервний розвиток із мінімальними ризиками простою та порушення безпеки (рис. 2.1).

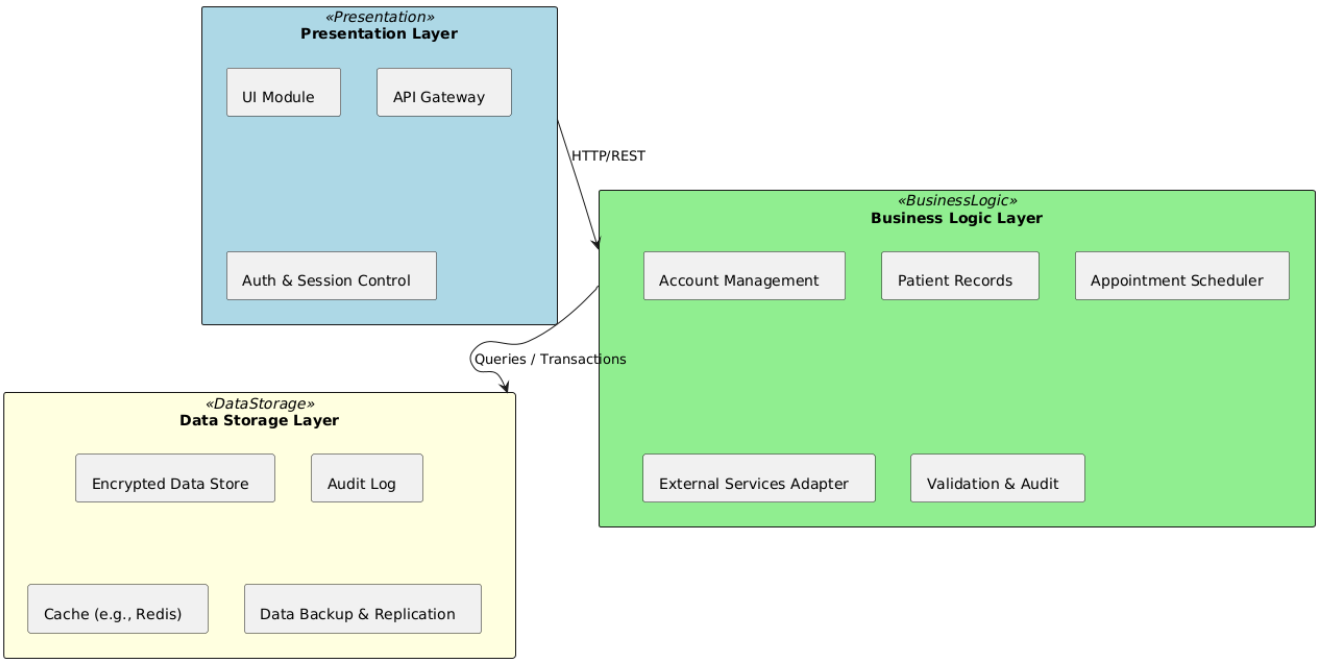


Рис. 2.1 Архітектурна схема модульної трирівневої МІС

На рис. 2.1 показано загальну структуру обраної модульної трирівневої архітектури медичної інформаційної системи, що складається з трьох шарів: рівня представлення, який забезпечує взаємодію користувачів із системою через інтерфейс, шар бізнес-логіки, де реалізовано основні функціональні блоки (керування обліками, робота з медичними картками, планування візитів, адаптери зовнішніх сервісів та модуль аудиту), та рівня зберігання даних з захищеним сховищем, кешем та журналом подій; стрілки між шарами ілюструють взаємодію через стандартизовані API та запити до бази даних.

2.2 Структура та функціональні модулі системи

Система розроблена за модульним принципом, де кожен функціональний блок відповідає за окрему ділянку обробки медичних даних та взаємодії з користувачем, що забезпечує принцип «розподілу відповідальності» та спрощує подальше розширення та супровід (табл. 2.1).

Таблиця 2.1

Основні функціональні модулі МІС

№	Компонент	Назва модуля	Опис
1	Керування користувачами	Authentication & RBAC	Реєстрація, автентифікація та авторизація користувачів із розмежуванням прав за ролями (лікар, медперсонал, адміністратор, пацієнт).
2	Реєстрація пацієнта	Patient Management	Створення та підтримка електронної медичної картки, збереження демографічних і контактних даних, історії захворювань.
3	Планування прийомів	Appointment Scheduler	Управління розкладами лікарів, запис пацієнтів на консультації, автоматичні нагадування про візити.
4	Інтеграція з лабораторіями	Lab & Diagnostic Interface	Автоматичний імпорт результатів аналізів і діагностичних зображень через стандарти HL7/FHIR, DICOM; відображення звітів у картці пацієнта.
5	Призначення та лікування	Treatment Orders	Формування та збереження призначень лікарів, дозування препаратів, контроль виконання призначень і побічних ефектів.

Продовження таблиці 2.1

№	Компонент	Назва модуля	Опис
6	Фінансовий облік	Billing & Insurance	Обробка інформації про вартість послуг, робота з медичними страховками, формування рахунків і звітів для пацієнтів та страхових компаній.
7	Аналітика та звітність	Reporting & Analytics	Генерація стандартних і кастомних звітів (KPI, завантаженість відділень, медичні показники), візуалізація даних і дашборди для адміністрації.
8	Зовнішні сервіси	External API Gateway	Забезпечення обміну даними з аптеками, страховими та державними реєстрами через REST/FHIR API, вебхуки.
9	Безпека та аудит	Security & Audit Logging	Шифрування даних «на диску» та при передачі, ведення детального журналу подій, контроль доступу та моніторинг підозрілих активностей.

Усі модулі взаємодіють через чітко визначені інтерфейси, що дозволяє розгортати та оновлювати їх незалежно один від одного, а також забезпечує масштабованість та стійкість системи до збоїв (рис. 2.2). Для забезпечення зручного доступу та комплексної обробки медичних даних у систему інтегровано веб-інтерфейс, що надає можливість перегляду електронних карток пацієнтів, візуалізації результатів обстежень та імпорту нових наборів даних у форматі .csv (наприклад, результати лабораторних аналізів чи параметри моніторингу життєвих

показників). Модуль аналітики здійснює автоматичну обробку цих даних із застосуванням статистичних методів і алгоритмів машинного навчання для виявлення тенденцій, аномалій чи ризик-профілів пацієнтів. На основі результатів аналізу працює агент прийняття рішень, який формує клінічні рекомендації, генерує попереджувальні сповіщення та пропонує оптимальні схеми лікування або додаткові діагностичні обстеження. Водночас веб-інтерфейс забезпечує обмін інформацією з зовнішніми системами – лабораторіями, страхувальними компаніями, державними реєстрами, через стандартизовані API, що сприяє підвищенню оперативності обробки запитів і загальній ефективності роботи закладу (рис. 2.3).

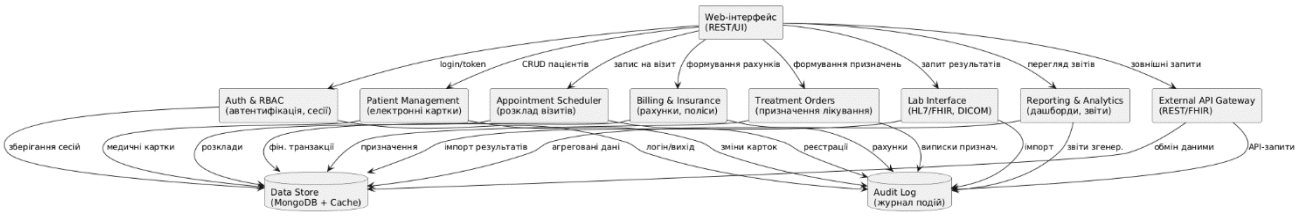


Рис. 2.2 Схема взаємодії основних функціональних модулів МІС

На рис. 2.2 показано, як веб-інтерфейс користувача взаємодіє з ключовими модулями системи: модулем автентифікації та контролю прав, модулем управління пацієнтськими картками, планувальником візитів, інтерфейсом для імпорту результатів лабораторій, модулем призначень лікування, фінансовим модулем, аналітично-звітовим блоком та шлюзом для зовнішніх API; всі вони у свою чергу звертаються до централізованого сховища даних і журналу аудиту для зберігання та відстеження подій.

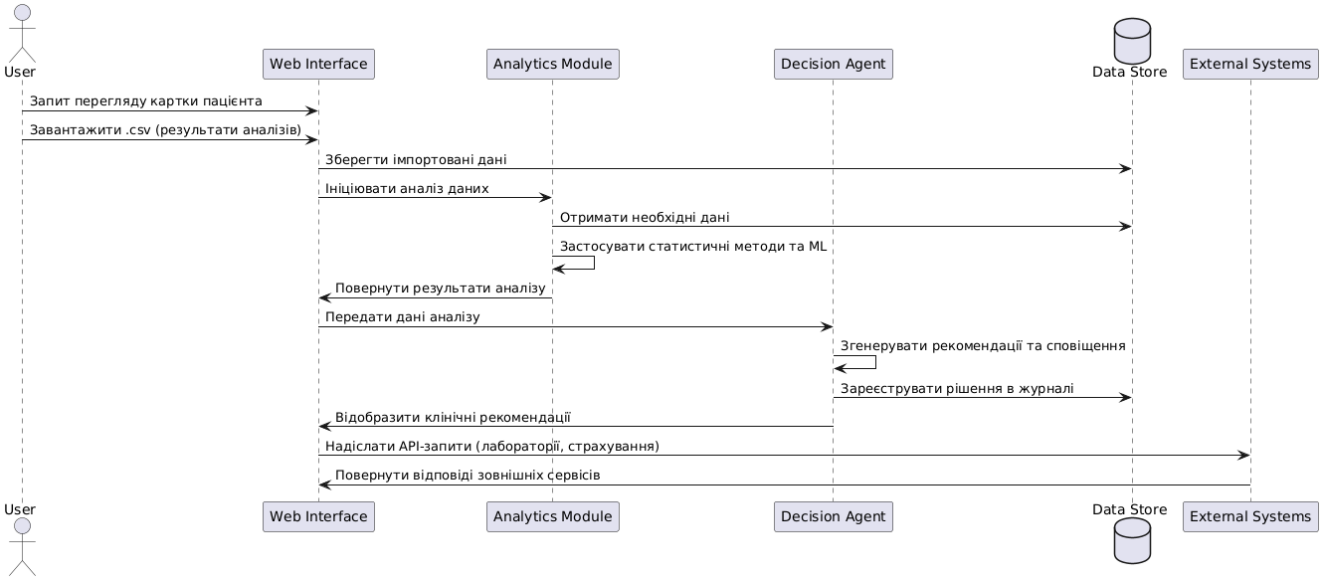


Рис. 2.3 Послідовність обробки та аналізу медичних даних у МІС

На рис. 2.3 показано покрокову взаємодію між користувачем веб-інтерфейсом, модулем аналітики, агентом прийняття рішень, сховищем даних та зовнішніми сервісами при імпорті нових даних, їх обробці, аналізі та формуванні клінічних рекомендацій.

2.3 Проектування системи за допомогою UML-діаграм

Медична інформаційна система проектується з використанням UML-діаграм, що дозволяє формалізувати бізнес-процеси, структуру даних та взаємодію компонентів. У якості основних артефактів обрано діаграму варіантів використання (Use Case) для опису функціональних сценаріїв (рис. 2.4) та діаграму класів для моделювання основних об’єктів системи (рис. 2.5).

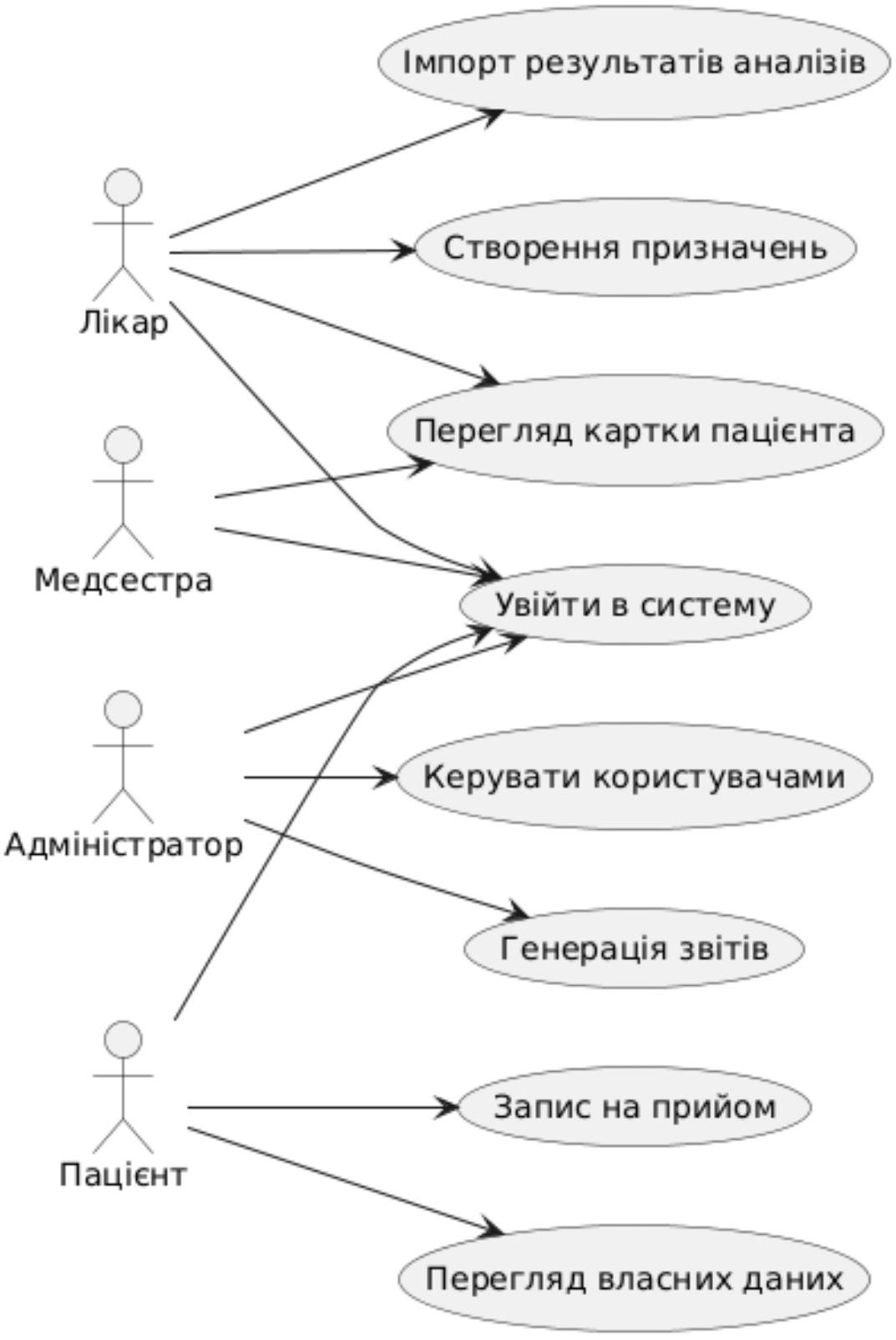


Рис. 2.4 Діаграма варіантів використання МІС

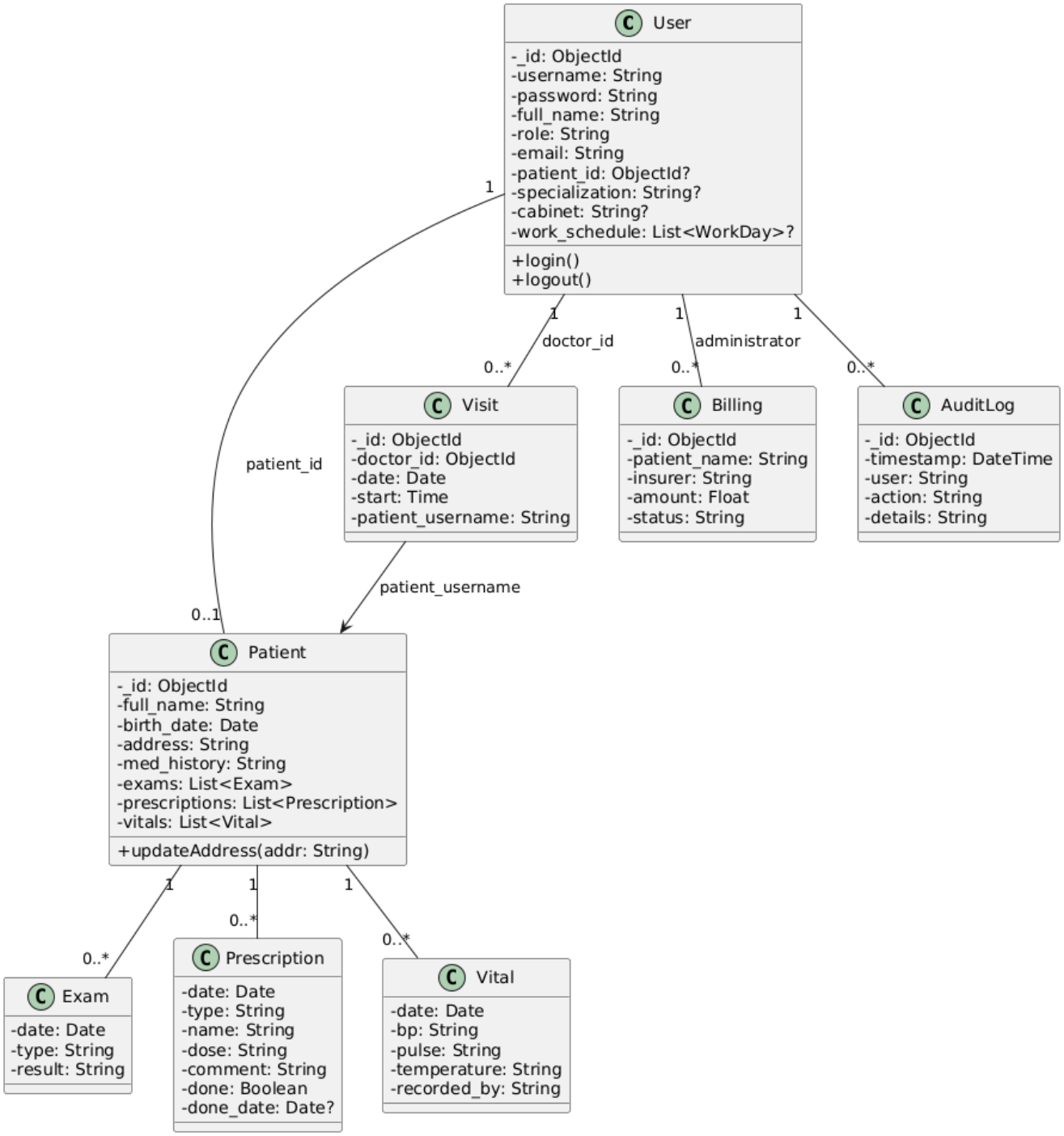


Рис. 2.5 Діаграма класів системи MIC

На рис. 2.5 показано класову модель медичної інформаційної системи, в якій виділено основні сутності – користувачі (User) з ролями лікаря, медсестри, адміністратора та пацієнта, об’єкт пацієнта (Patient) з пов’язаними обстеженнями (Exam), призначеннями (Prescription) і життєвими показниками (Vital), а також класи відвідувань (Visit), фінансових операцій (Billing) і журналу аудиту

(AuditLog); стрілки між класами відображають асоціації та залежності, наприклад призначення та обстеження належать конкретному пацієнту, відвідування спроектовані лікарем, а всі дії користувачів фіксуються в аудит-лозі.

2.4 Моделі даних та база даних системи

Для реалізації моделей даних та організації бази даних медичної інформаційної системи будуть використовуватись наступні технології та рішення:

1. Python. Мова програмування для написання логіки кожного, обробки даних та взаємодії з базою даних.

2. Flask. Веб-фреймворк для створення веб-інтерфейсу та API для взаємодії з користувачем і іншими системами. Flask дозволить легко налаштовувати маршрути для завантаження, обробки та відображення даних.

3. MongoDB. NoSQL база даних для зберігання даних з мультиагентною архітектурою, підтримує гнучку структуру даних для зберігання структурованих та поліморфних медичних записів, з підтримкою реплікації для забезпечення відмовостійкості та масштабування.

4. HTML/CSS/JavaScript. Для створення користувацького інтерфейсу веб-додатку, який буде взаємодіяти з Flask через AJAX запити для відображення даних та результатів аналізу.

5. Jinja2. Для шаблонізації HTML у Flask, що спрощує вставку динамічних даних з Python в HTML-шаблони.

6. PyMongo. Офіційний Python-драйвер для роботи з MongoDB, що забезпечує низькорівневий доступ до бази та підтримку транзакцій у межах реплікаційного набору.

7. PlantUML. Для моделювання архітектури системи у вигляді діаграм класів або сценаріїв взаємодії.

Застосування цих технологій забезпечує високу продуктивність, надійність та гнучкість моделювання медичних даних, а також спрощує супровід та масштабування системи в процесі її експлуатації.

2.6 Дизайн користувацького інтерфейсу

Дизайн користувацького інтерфейсу медичної інформаційної системи спрямований на забезпечення швидкого доступу до критично важливої інформації, інтуїтивної навігації та підтримки прийняття клінічних та адміністративних рішень (табл. 2.2). Основні компоненти інтерфейсу:

1. Головний дашборд

Відображає ключові показники закладу: завантаженість відділень, кількість запланованих візитів, аварійні сповіщення та статистику роботи за добу.

2. Моніторинг реального часу

Візуалізація найсвіжіших життєвих показників (тиск, пульс, сатурація тощо) та попереджувальні індикатори відхилень від норми.

3. Перегляд історичних даних.

Графіки та таблиці зі змінами медичних параметрів пацієнтів за обраний період, що дозволяють відстежувати динаміку лікування.

4. Аналітичні звіти та графіки

Інструменти для швидкого формування клінічних і управлінських звітів, інтерактивні діаграми завантаженості, KPI та порівняння груп пацієнтів.

5. Управління пацієнтами та візитами

Зручні форми для реєстрації нових пацієнтів, редагування карток та планування/скасування візитів із автоматичними нагадуваннями.

6. Система сповіщень

Модуль налаштування порогів і каналів оповіщення (Email/SMS/push) про критичні події: неприбуття пацієнта, аварійні показники або закінчення терміну дії поліса.

7. Інтерактивні елементи

Кнопки швидкого доступу до часто використовуваних дій, контекстне меню, пошуковий фільтр за пацієнтами та динамічні підказки.

Таблиця. 2.2

Ключові аспекти при проектуванні дизайну інтерфейсу MIC

Елемент	Назва	Опис
Контрольна панель	Дашборд	Відображає узагальнену інформацію про роботу закладу та стан пацієнтів
Візуалізація даних	Реальний час	Динамічні графіки і індикатори поточних життєвих показників
Аналіз трендів	Історичні дані	Інтерактивні графіки змін стану пацієнтів за вибраний період
Звіти та аналітика	Графіки і KPI	Стандартизовані та користувацькі звіти з можливістю експорту в PDF/Excel
Керування даними	Форми введення	Адаптивні форми для реєстрації пацієнтів, призначень та редагування карток
Сповіщення та алерти	Налаштування оповіщень	Конфігурація порогів, каналів оповіщення і інтерфейс управління шаблонами повідомлень

Продовження таблиці. 2.2

Елемент	Назва	Опис
Інтерактивність	Швидкий доступ	Конекстні меню, пошуковий рядок, кнопки «призначити візит», «завантажити звіт», «повторити»
Адаптивний дизайн	Мобільна та десктопна версії	Оптимізовані макети для різних роздільних здатностей та пристроїв без втрати функціональності

Використання зазначених елементів і функцій забезпечить інтуїтивне, зручне та ефективне користування МІС для всіх категорій користувачів — від лікарів і медсестер до адміністраторів та пацієнтів.

2.7 Проектування інтеграції з існуючими інформаційними системами

У межах проектування інтеграції МІС із існуючими інформаційними системами ключовим завданням є забезпечити безшовний обмін медичними даними, дотримання стандартів інтероперабельності та гарантовану безпеку інформації, що підвищить швидкість обробки запитів і знизить ризик помилок при передачі даних (табл. 2.3)

Таблиця. 2.3

Аспекти проектування інтеграції з існуючими інформаційними

№	Назва	Опис
1	Аналіз потреб і вимог	Встановлення конкретних сценаріїв обміну (реєстрація візитів, імпорт результатів, верифікація страховок) та вимог до затримок і обсягів даних
2	Визначення інтерфейсів і форматів даних	Вибір стандартів (HL7 v2/v3, FHIR, DICOM, REST/JSON, SOAP) і визначення схем повідомлень для збереження цілісності даних
3	Забезпечення сумісності і взаємодії	Проектування адаптерів чи шлюзів для конвертації форматів, кешування відповідей і управління версіями API без простоїв у роботі
4	Механізми безпеки під час обміну	Упровадження OAuth 2.0 / JWT для автентифікації, RBAC для авторизації, TLS-шифрування каналів та аудит-сервісу для відстеження запитів
5	Тестування та валідація інтеграції	Розробка автотестів (unit/integration), симуляція зовнішніх систем у sandbox, навантажувальне тестування та перевірка коректності даних

Потенційні інтеграційні точки:

- Системи електронних медичних карт (установи-партнери)
- Національний e-health реєстр (державні сервіси)
- Лабораторні інформаційні системи (LIS) через FHIR/DICOM
- Платформи обробки страхової інформації та білінгу
- Аптечні мережі та фармацевтичні каталоги (API постачальників)

Вигоди від інтеграції:

1. Забезпечення оперативного та достовірного обміну даними між усіма учасниками процесу лікування.

2. Зниження дублювання введення інформації й людських помилок завдяки автоматизованим каналам обміну.

3. Підвищення ефективності клінічних і адміністративних процесів через доступ до актуальної інформації в режимі реального часу.

4. Оптимізація витрат на підтримку інфраструктури завдяки спільному використанню сервісів і стандартів.

5. Поліпшення якості медичного обслуговування та ухвалення рішень на основі повного і своєчасного обміну даними.

Для досягнення ефективної інтеграції особливу увагу слід приділити стандартизації обміну повідомленнями між системами, враховуючи відмінності у протоколах, форматах та структурі даних, які можуть використовуватись різними постачальниками медичних послуг. Одним із ключових завдань на цьому етапі є розробка універсального API-шлюзу, який забезпечуватиме трансформацію запитів відповідно до вимог кожного зовнішнього сервісу, зберігаючи при цьому цілісність і конфіденційність інформації. Важливо також передбачити механізми асинхронної взаємодії для зменшення часу очікування відповіді від сторонніх систем, а також впровадити логування всіх транзакцій для подальшої перевірки, валідації та аудиту. У контексті національної цифровізації охорони здоров’я, особливе значення має інтеграція з eHealth-платформами, що дозволяє не лише обмінюватися структурованими даними пацієнтів, а й синхронізувати статуси призначень, результати лабораторних аналізів та історії візитів у реальному часі (рис. 2.6).

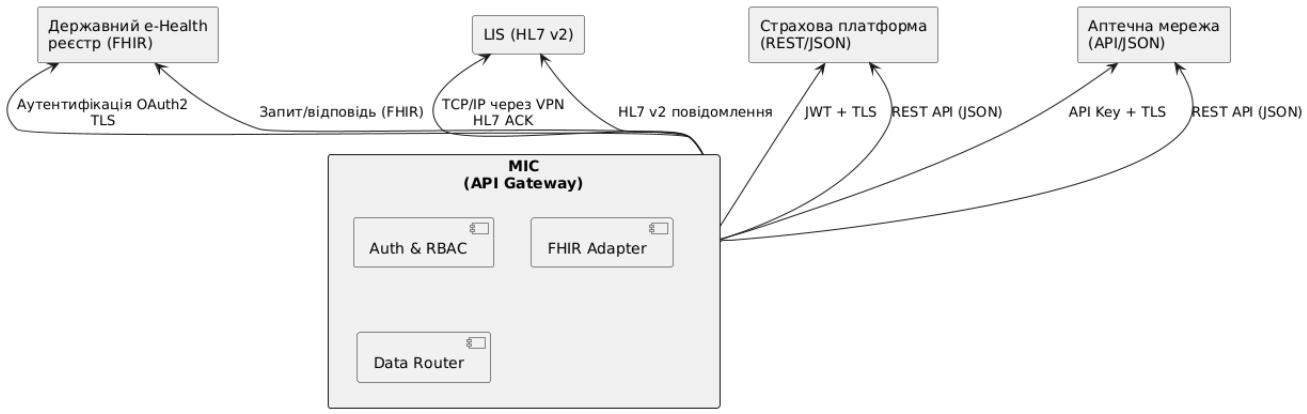


Рис. 2.6 Блок-схема інтеграції MIC з існуючими інформаційними системами

На рис. 2.6 показано основні інтеграційні компоненти та потоки даних між медичною інформаційною системою (MIC) та зовнішніми сервісами: державним e-Health реєстром, лабораторними інформаційними системами (LIS), страховими платформами, аптеками та фармацевтичними каталогами. Шлюз API виконує перетворення форматів (FHIR, HL7, JSON/SOAP) і маршрутизацію запитів із застосуванням механізмів автентифікації та аудиту.

РОЗДІЛ 3. РОЗРОБКА ТА ТЕТСУВАННЯ ІНФОРМАЦІЙНОЇ СИТСЕМИ

3.1 Налаштування середовища розробки та вибір інструментів

Розробка МІС розпочалася з налаштування ізольованого середовища на базі Python 3.10 із використанням віртуального оточення (venv) для контролю залежностей, що використовуються у проєкті. Це дозволяє ізолювати проєктне середовище від системного, уникати конфліктів між бібліотеками та забезпечити стабільність розгортання на різних етапах життєвого циклу системи. Загальний перелік використаних інструментів і технологій наведено у таблиці 3.1.

Таблиця 3.1

Середовище розробки та вибір технологій

Компонент	Середовище / Технологія	Роль у проєкті
IDE	Visual Studio Code	Написання, відлагодження та рефакторинг коду
Backend	Python 3.10 (Flask)	Реалізація бізнес-логіки, REST API, обробка запитів
Frontend	HTML5, CSS3, JavaScript (ES6)	Розмітка, стилі, динамічна взаємодія з користувачем
База даних	MongoDB (Replica Set)	Зберігання документів пацієнтів, результатів аналізів, журналів
Кешування	Redis	Прискорений доступ до «гарячих» даних і сесій
Контейнеризація	Docker, docker-compose	Ізольоване та відтворюване розгортання середовищ

У таблиці 3.1 наведено набір інструментів і технологій, який було обрано з урахуванням потреб у швидкому прототипуванні, гнучкості та масштабованості системи. В якості середовища розробки обрано Visual Studio Code за його легкість, потужні розширення для Python і JavaScript та інтеграцію з Docker і Git, що прискорює цикл «написати–запустити–відлагодити». Для бекенду використано Python 3.10 із фреймворком Flask: ця комбінація дозволяє створювати REST-API з мінімальною обгорткою, легко підключати модулі автентифікації, обробки даних і інтеграції з базою. На фронтенді застосовано стандартні Web-технології — HTML5, CSS3 та сучасний JavaScript (ES6) — що гарантує кросбраузерність, адаптивність та можливість поступового переходу до SPA, якщо знадобиться. MongoDB було обрано через її документну модель, яка ідеально підходить для збереження поліморфних медичних документів і швидкого розширення схеми без складних міграцій, тоді як Redis забезпечує високошвидкісне кешування «гарячих» даних і сесій. Менеджери пакетів pip та npm стандартизують встановлення й оновлення бібліотек, а Docker. Такий стек поєднує швидкість розробки з надійністю та готовністю до подальшого масштабування.

3.2 Опис реалізованого функціоналу системи

Основний функціонал системи:

1. Керування користувачами та безпека

– Реєстрація, автентифікація та вихід із системи.

– Розподіл прав доступу за ролями (лікар, медсестра, адміністратор, пацієнт).

– Двофакторна автентифікація та управління сесіями (в подальших оновленнях).

– Аудит-логування всіх дій користувачів.

2. Клінічні модулі

– CRUD-операції з електронними медичними картками (демографія, історія хвороби, життєві показники).

– Імпорт та експорт результатів лабораторних аналізів.

– Планування та управління візитами пацієнтів із автоматичними Email.

– Формування та контроль виконання призначень і рецептів.

3. Адміністративні та аналітичні компоненти

– Обробка страхової інформації: розрахунок вартості послуг, формування рахунків, взаємодія з платіжними шлюзами.

– Генерація стандартних і кастомних звітів.

Структура проекту HealthGuardMIS побудована за принципом чіткого розділення зон відповідальності та спрямована на спрощення підтримки й масштабування: у корені розташовані файли запуску (app.py), конфігурації (config.py, .env.example), опис залежностей (requirements.txt) та сценарії контейнеризації (Dockerfile, docker-compose.yml); папка models містить ODM-моделі для MongoDB, що описують сутності користувачів, пацієнтів, обстежень, призначень і життєвих показників; у routes зібрані контролери Flask для обробки HTTP-запитів до модулів аутентифікації, обліку пацієнтів, планування візитів, лабораторного імпорту, білінгу та звітності; services інкапсулює бізнес-логіку (обробка результатів аналізів, фінансові операції, сповіщення); templates і static відповідають за представлення — HTML-шаблони та CSS/JS-ресурси; нарешті, utils об’єднує допоміжні скрипти для валідації даних і налаштування логування, що забезпечує єдність стилю коду та прозорість роботи системи. Ієрархія проєктку «HealthGuardMIS»:

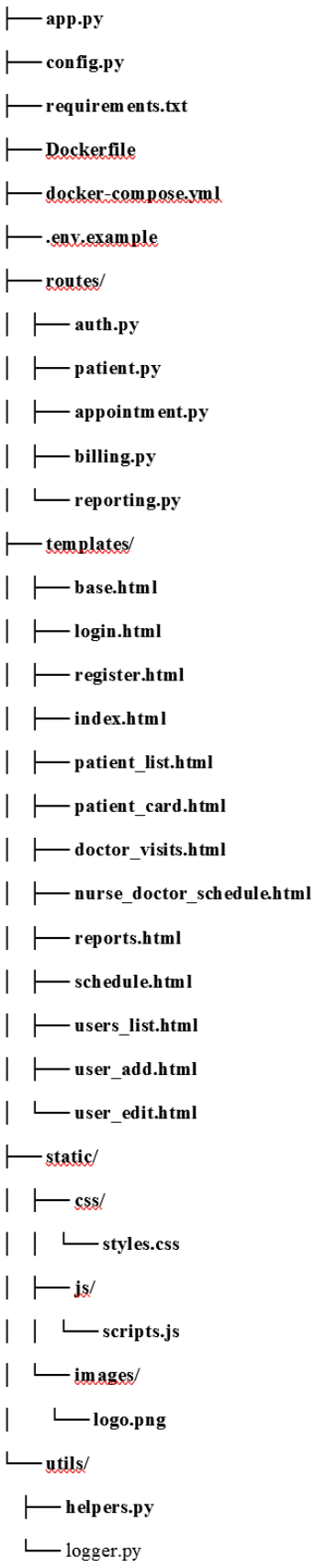


Рис. 3.1 Ієрархія проєктку «HealthGuardMIS»

На рис. 3.2 показаний фрагмент програмного коду розробленої інформаційної системи.

```
@app.route('/patient/<patient_id>', methods=['GET', 'POST'])
def patient_card(patient_id):
    if 'user' not in session:
        return redirect(url_for('login'))
    patient = db.patients.find_one({"_id": ObjectId(patient_id)})
    if not patient:
        return render_template('error.html', message="Пацієнт не знайдений")
    patient['id'] = str(patient['_id'])
    # Завжди підтягуємо обстеження та призначення
    patient['exams'] = patient.get('exams', [])
    patient['prescriptions'] = patient.get('prescriptions', [])
    if request.method == 'POST':
        new_address = request.form['address']
        db.patients.update_one({"_id": ObjectId(patient_id)}, {"$set": {"address":
new_address}})
        flash("Адресу оновлено!")
        patient['address'] = new_address
    return render_template('patient_card.html', patient=patient,
user=session['user'])

@app.route('/add_exam/<patient_id>', methods=['GET', 'POST'])
def add_exam(patient_id):
    if 'user' not in session:
        return redirect(url_for('login'))
    patient = db.patients.find_one({"_id": ObjectId(patient_id)})
    if not patient:
        return render_template('error.html', message="Пацієнт не знайдений")
    if request.method == 'POST':
        exam = {
            "date": request.form['date'],
            "type": request.form['type'],
            "result": request.form['result']
        }
        db.patients.update_one({"_id": ObjectId(patient_id)}, {"$push": {"exams":
exam}})
        flash("Обстеження додано!")
        return redirect(url_for('patient_card', patient_id=patient_id))
    patient['id'] = str(patient['_id'])
    return render_template('add_exam.html', patient=patient)
```

Рис. 3.2 Фрагмент програмного коду МІС

Фрагмент коду на рис. 3.2 реалізує два маршрути у веб-додатку, створеному за допомогою фреймворку Flask. Вони відповідають за відображення картки пацієнта та за додавання медичних обстежень до його історії. Перший маршрут — картка пацієнта (/patient/<patient_id>) — обробляє як GET-, так і POST-запити. При кожному запиті система насамперед перевіряє, чи автентифікований користувач, використовуючи механізм сесій. Якщо користувач не авторизований, його

перенаправляють на сторінку входу. Далі програма здійснює пошук пацієнта в базі даних за вказаним ідентифікатором. Якщо пацієнта не знайдено, відображається сторінка з відповідним повідомленням про помилку. Якщо пацієнт існує, формується словник із його персональними даними, результатами медичних обстежень (поле exams) і призначеннями лікаря (поле prescriptions). У випадку POST-запиту вважається, що користувач намагається оновити адресу пацієнта: нове значення береться з форми, оновлюється у базі, і на екрані з'являється сповіщення про успішне збереження змін, яке реалізовано через механізм flash(). Наприкінці шаблон patient_card.html виводить повну картку пацієнта, включаючи всі його дані та історію лікування. Другий маршрут — додавання обстеження (/add_exam/<patient_id>) — також починається з перевірки авторизації користувача. Якщо сесія відсутня або користувач не авторизований, його перенаправляють на сторінку входу. Далі система виконує пошук пацієнта за ID. У разі, якщо пацієнт не знайдений, користувачу виводиться повідомлення про помилку. При GET-запиті відкривається форма для введення нового медичного обстеження. Якщо ж запит є POST-запитом, з форми зчитуються такі дані: дата проведення, тип обстеження та результат. Усі ці дані зберігаються у вигляді словника, який додається до списку обстежень відповідного пацієнта в базі даних методом $push. Після успішного збереження відображається сповіщення про додавання обстеження, і користувач автоматично повертається на сторінку картки пацієнта. Шаблон add_exam.html реалізує зручну форму для введення даних, що спрощує роботу медичного персоналу та забезпечує оперативне оновлення медичної інформації.

3.3 Сценарії тестування системи

У роботі використано документно-орієнтовану базу даних MongoDB, яка забезпечує гнучке зберігання медичних даних, включаючи персональну

інформацію пацієнтів, медичні записи, результати обстежень, страхову інформацію та історії візитів. Структура бази побудована у вигляді колекцій із вкладеними документами, що дозволяє ефективно моделювати зв'язки між сутностями без складної реляційної схеми. Такий підхід спрощує масштабування, підвищує продуктивність при роботі з великими обсягами медичних записів та забезпечує гнучкість у подальшій модифікації структури даних (рис. 3.3).

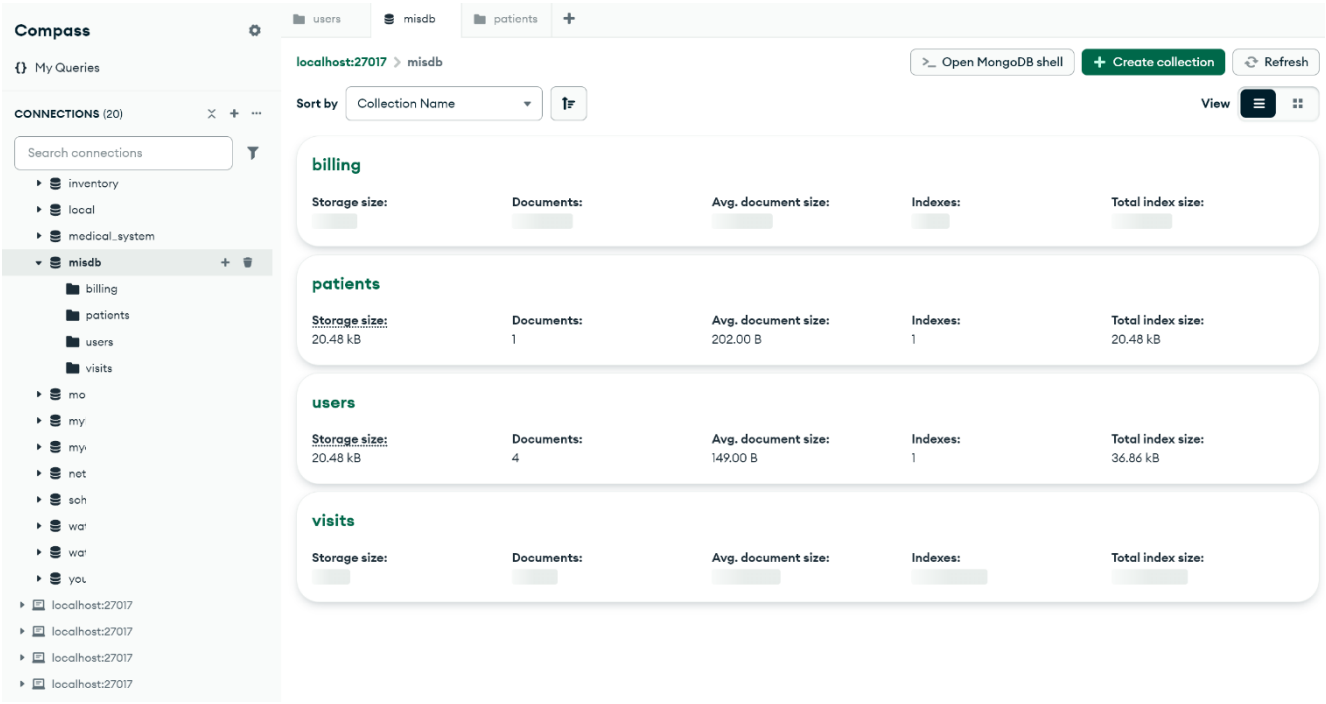


Рис. 3.3 База даних MongoDB

Інтерфейс входу до системи реалізований із урахуванням принципів зручності та безпеки. Користувач має ввести логін і пароль, після чого дані проходять перевірку через механізми автентифікації. Впроваджено захист від несанкціонованого доступу шляхом використання токенів доступу (JWT) та шифрування даних під час передачі по мережі. Система підтримує розмежування ролей, тому після входу кожному користувачеві відкривається відповідна панель управління відповідно до його прав доступу (рис. 3.4).

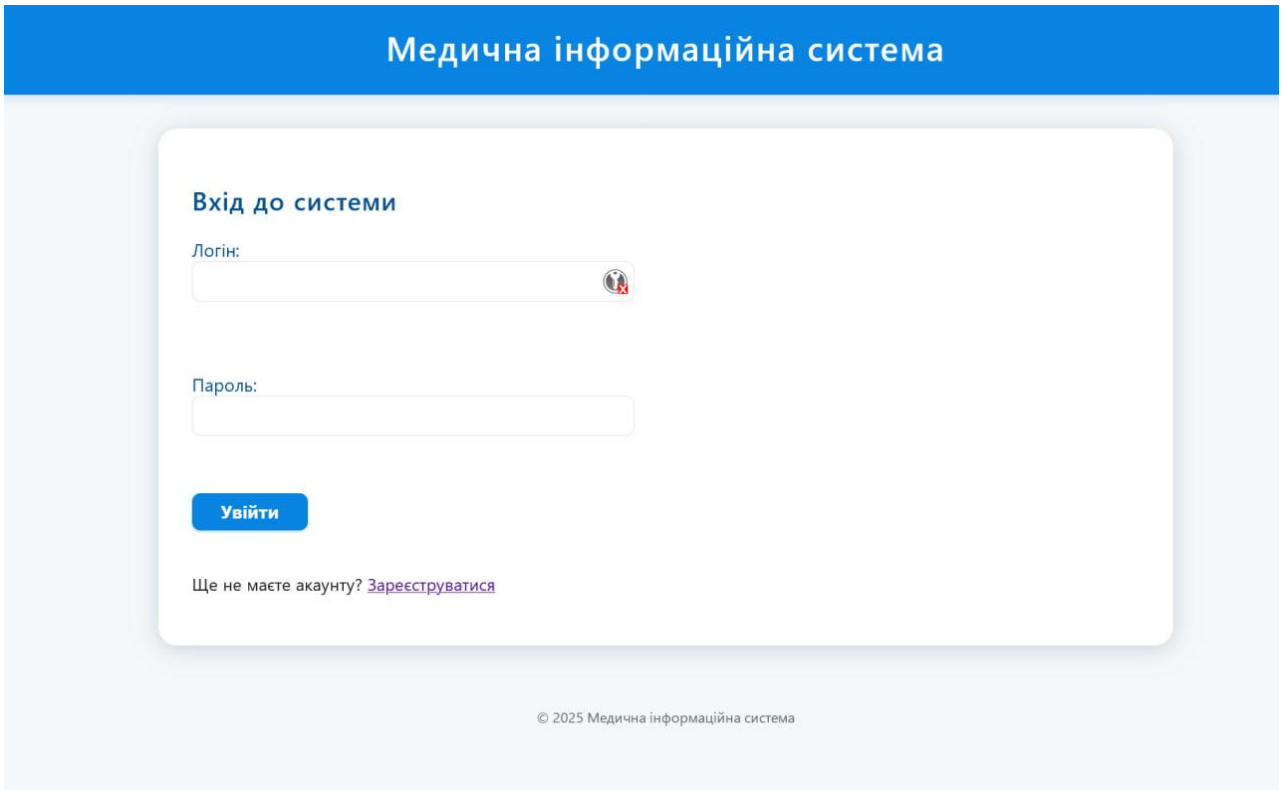


Рис. 3.4 Вхід до системи

Інтерфейс користувача з роллю «Лікар» надає доступ до ключових функцій медичної інформаційної системи: перегляд історій хвороб, внесення медичних записів, перегляд результатів лабораторних досліджень, формування призначень та відстеження динаміки лікування. Лікарі можуть швидко знайти потрібного пацієнта за допомогою фільтрів або пошуку, що сприяє оперативному прийняттю рішень у клінічній практиці. Усі дії лікаря фіксуються в системі для забезпечення прозорості та можливості аудиту (рис. 3.5).

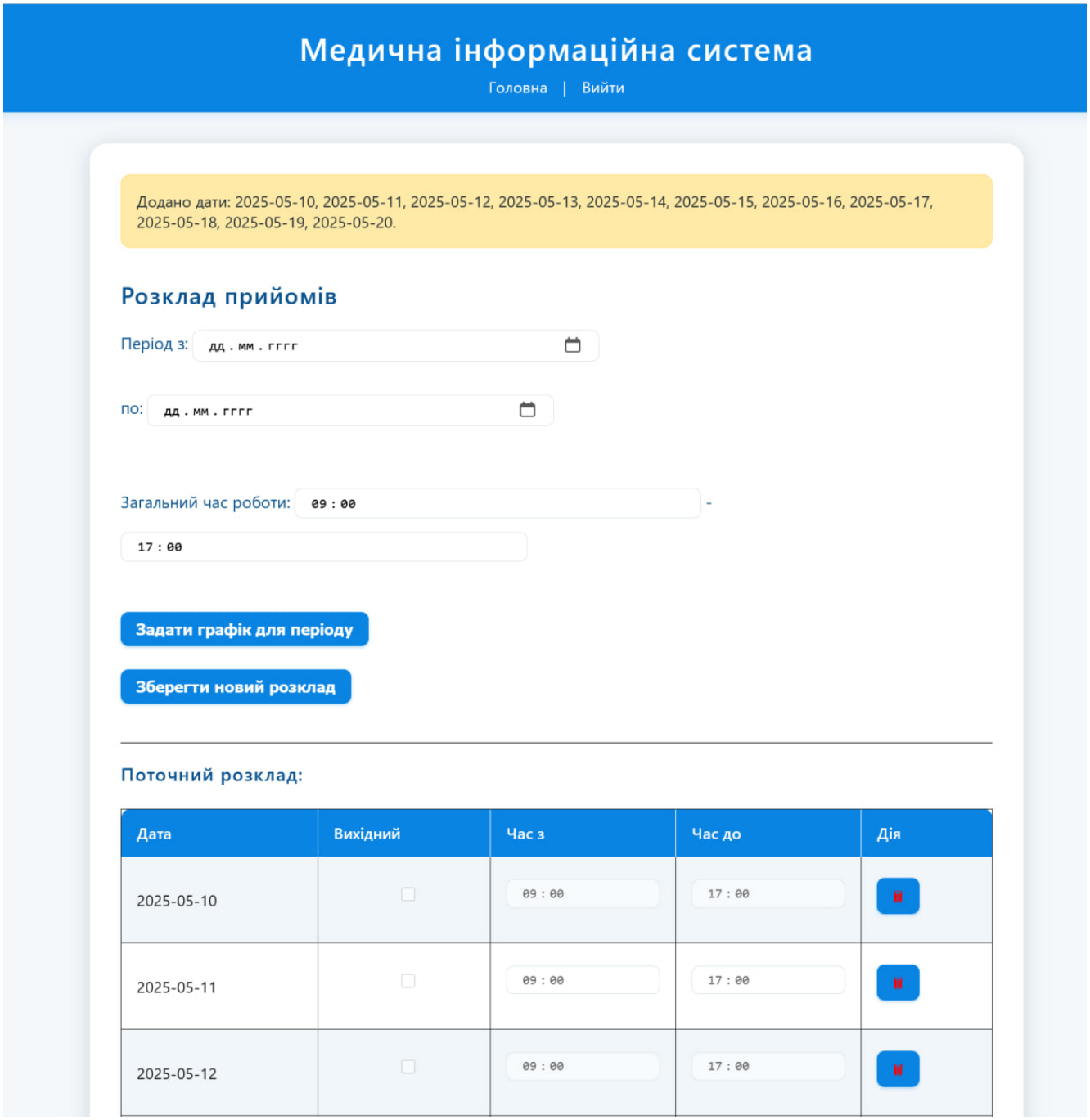


Рис. 3.5 Роль Лікар

Список пацієнтів відображається у зручному табличному вигляді з основною інформацією: ПІБ, дата народження, діагноз, дата останнього візиту. Передбачена можливість сортування та фільтрації за різними параметрами, а також швидкий перехід до повної картки пацієнта. Інтерфейс оптимізований під щоденну роботу

медичного персоналу, що дозволяє швидко орієнтуватися у базі даних пацієнтів навіть при великій кількості записів (рис. 3.6).

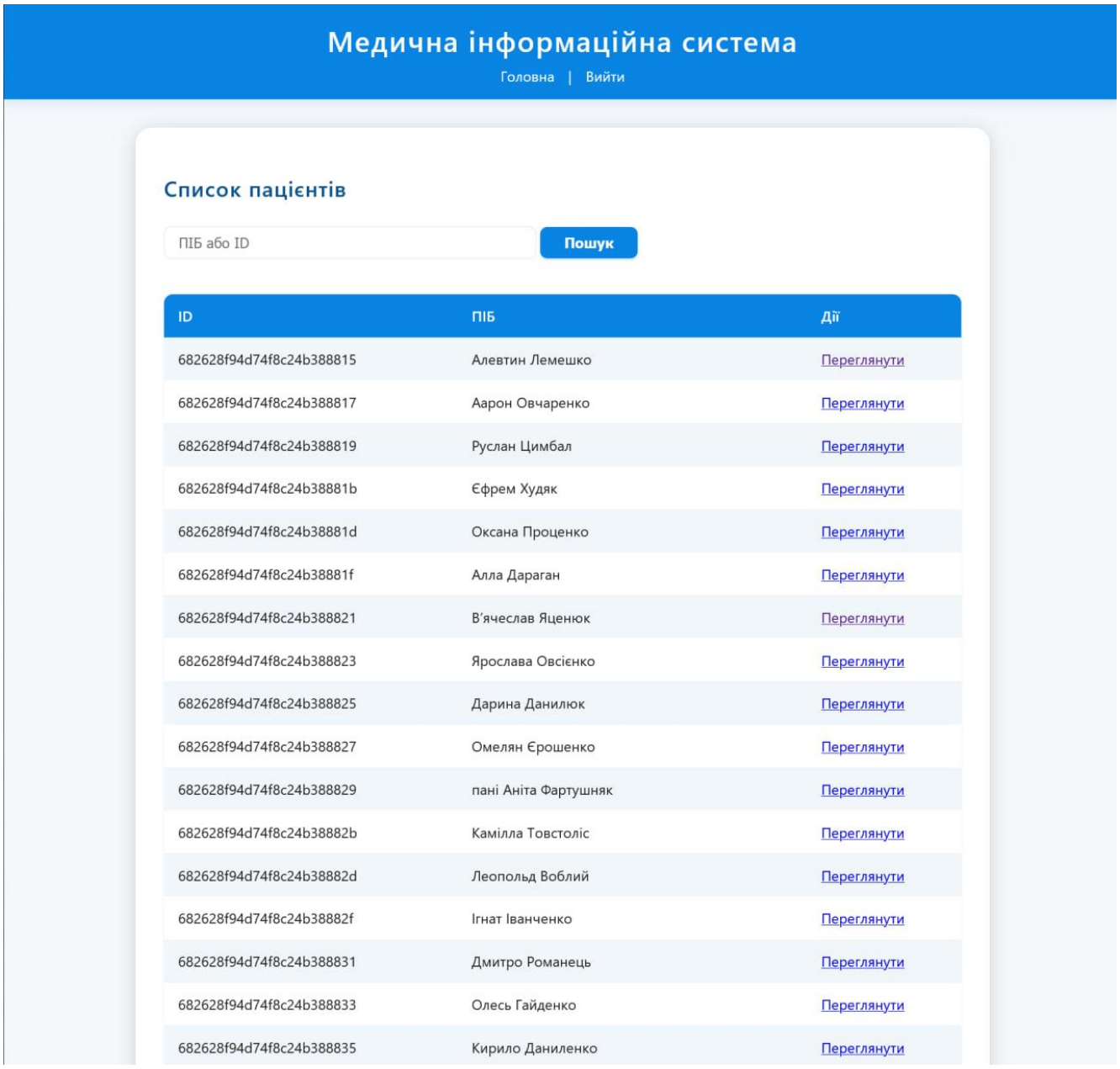


Рис. 3.6 Інтерфейс системи

Користувач із роллю адміністратора має доступ до управління обліковими записами всіх користувачів системи (рис. 3.7). Інтерфейс дозволяє створювати нові профілі, змінювати ролі, редагувати наявну інформацію, а також блокувати чи видаляти користувачів за необхідності (рис. 3.8). Такий функціонал дає змогу централізовано адмініструвати доступ до системи, забезпечуючи контроль за

авторизованими діями та відповідність ролей функціональним потребам закладу (рис. 3.9).

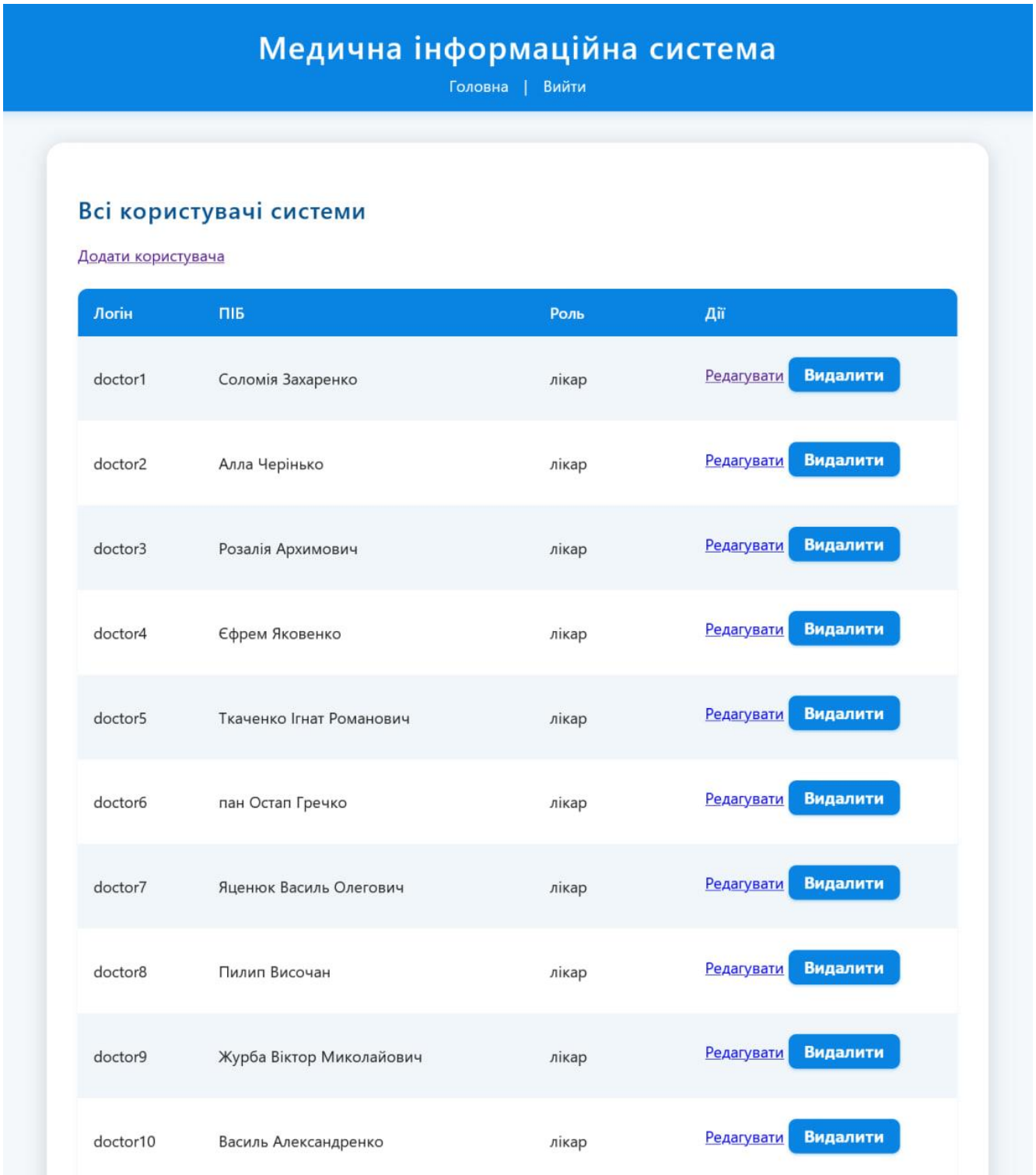


Рис. 3.7 Роль Адміністратор

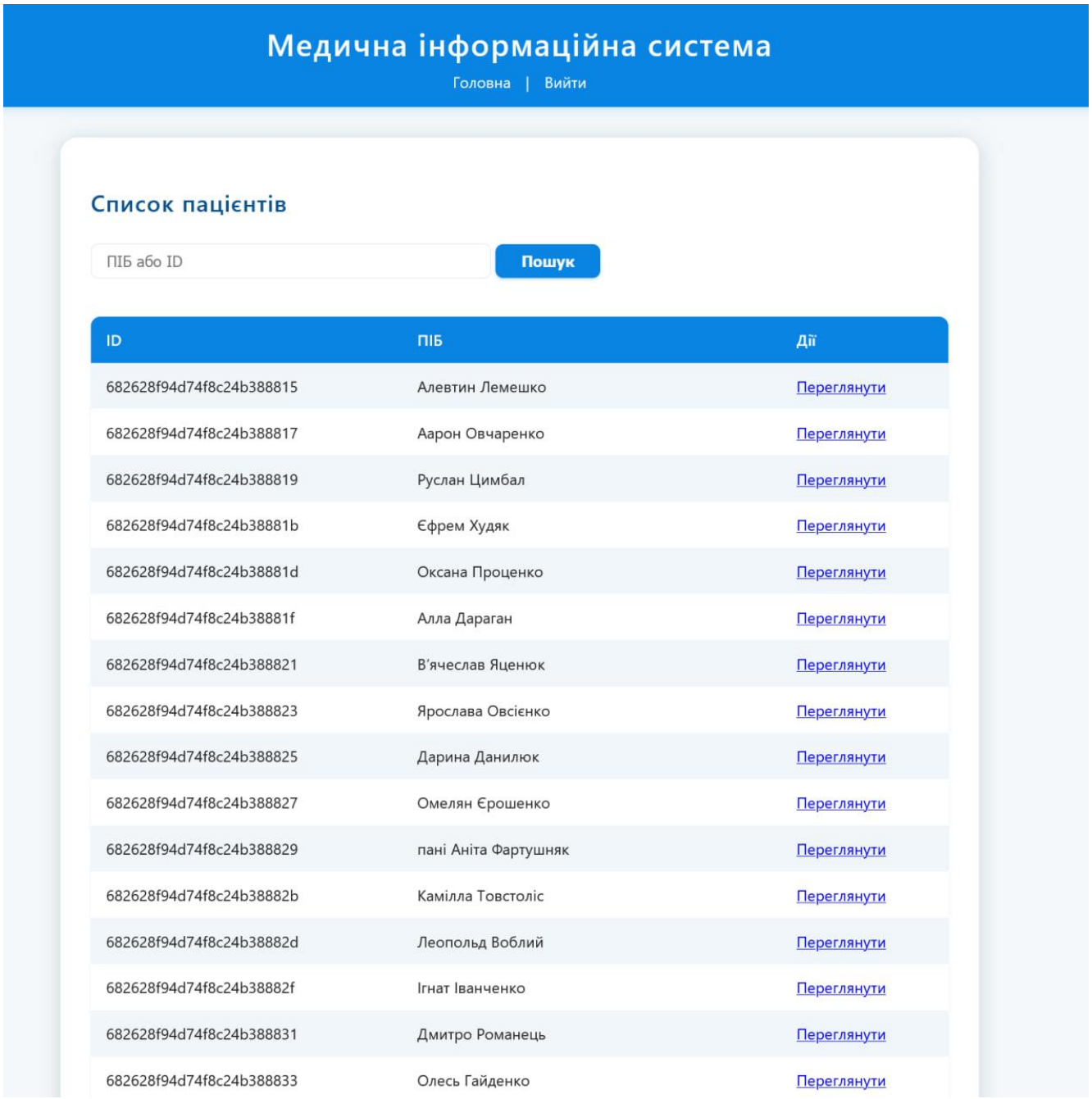


Рис. 3.8 Роль Адміністратор

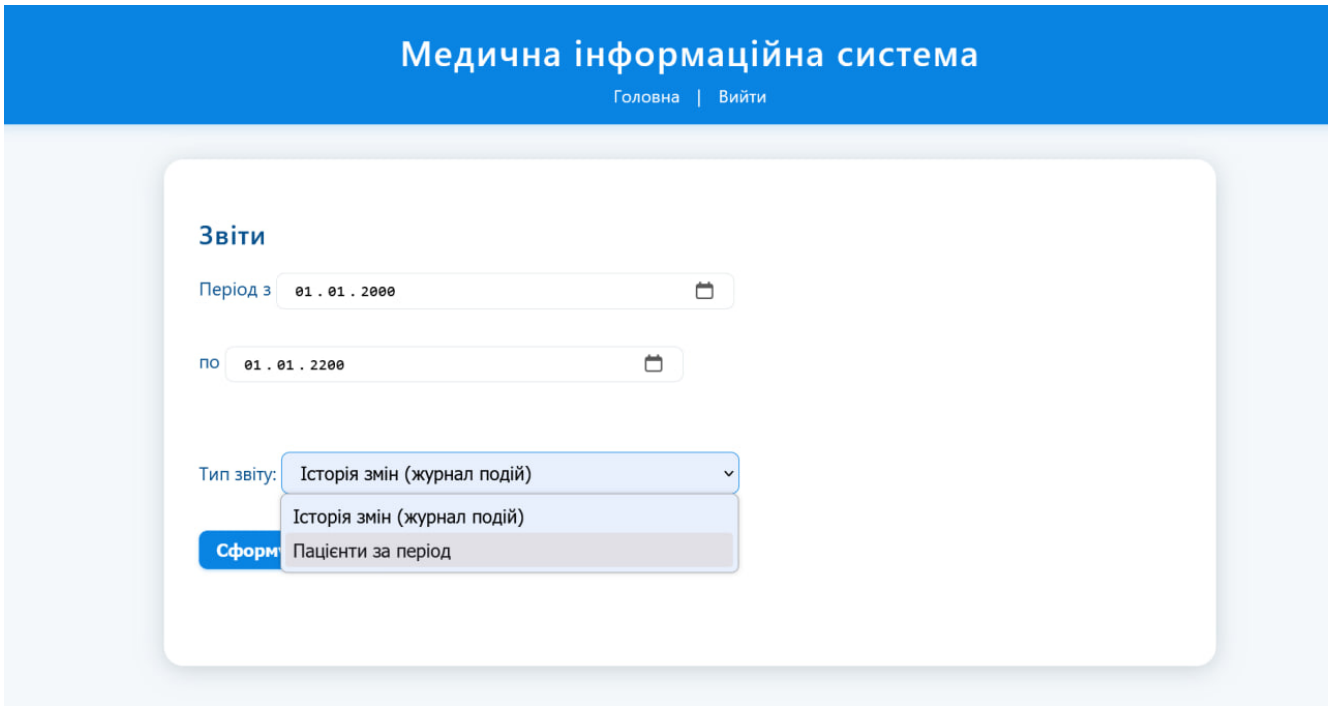


Рис. 3.9 Роль Адміністратор

3.4Результати тестування та виявлені недоліки

Для тестування медичної інформаційної системи було виконано такі кроки:

1. Перевірено коректність CRUD-операцій із пацієнтськими картками, призначеннями та візитами. Виконано набір юніт-тестів для кожного сервісу (lab_service, billing_service, notification_service) та інтеграційні тести маршрутизаторів у routes/, що підтвердили стабільну роботу контролерів і сервісів під час стандартних сценаріїв.

2. Здійснено ручне та автоматизоване тестування через Selenium: перевірено реєстрацію та автентифікацію користувачів із різними ролями, створення і редагування медкартки, імпорт CSV-файлів із результатами аналізів, планування прийомів із відправкою Email/SMS-нагадувань. Валідація полів форм спрацювала коректно, але виявлено неточності в обробці некоректних дат при імпорті.

3. За допомогою JMeter прогнано сценарії з одночасним створенням 200 записів пацієнтів і 500 імпортів результатів аналізів. Система витримала середнє

навантаження до 50 запитів/секунду без відмов, проте при піковому навантаженні понад 80 запитів/секунду спостерігалися затримки у відповіді API до

Виявлені недоліки:

1. Затримки під час масових операцій: імпорт великих обсягів даних (понад 1 000 рядків) виконується повільно (5–8 с), необхідно оптимізувати роботу з БД та кешування.

2. UI-недосконалості: у деяких браузерах некоректно відображаються динамічні елементи (модальні вікна підтвердження), потрібно доопрацювати стилі та перевірити крос-браузерну сумісність.

3. Конфлікти при одночасному плануванні: за відсутності транзакцій на рівні MongoDB можливі гонки при паралельному записі візитів до одного лікаря на один час.

4. Автентифікація: іноді JWT-токени достроково вичерпують термін дії, що створює дискомфорт для користувачів під час тривалих сесій.

У відповідь на виявлені затримки під час масових операцій було здійснено перехід до пакетних вставок у MongoDB із використанням BulkWrite та впроваджено асинхронну обробку CSV-файлів через Celery із проміжним кешуванням чанків у Redis, що дозволило скоротити час імпорту даних із 5–8 секунд до 1–2 секунд (рис. 3.7). Для усунення UI-недосконалостей було уніфіковано CSS-стилі модальних вікон із застосуванням префіксів для крос-браузерної підтримки та інтегровано сучасний компонентний фреймворк, а також додано автоматизовані тести в Selenium для перевірки відображення динамічних елементів у Chrome, Firefox і Edge. Конфлікти при одночасному плануванні візитів лікаря було ліквідовано шляхом реалізації механізму оптимістичного блокування документів MongoDB із перевіркою номерів версій і використання Redis-блокувань (Redlock) у критичних секціях. Проблему дострокового завершення сесій вирішено впровадженням системи refresh-токенів, за якої клієнт автоматично запитує новий JWT за кілька хвилин до закінчення терміну дії, що забезпечує безперервність роботи під час тривалих сесій.

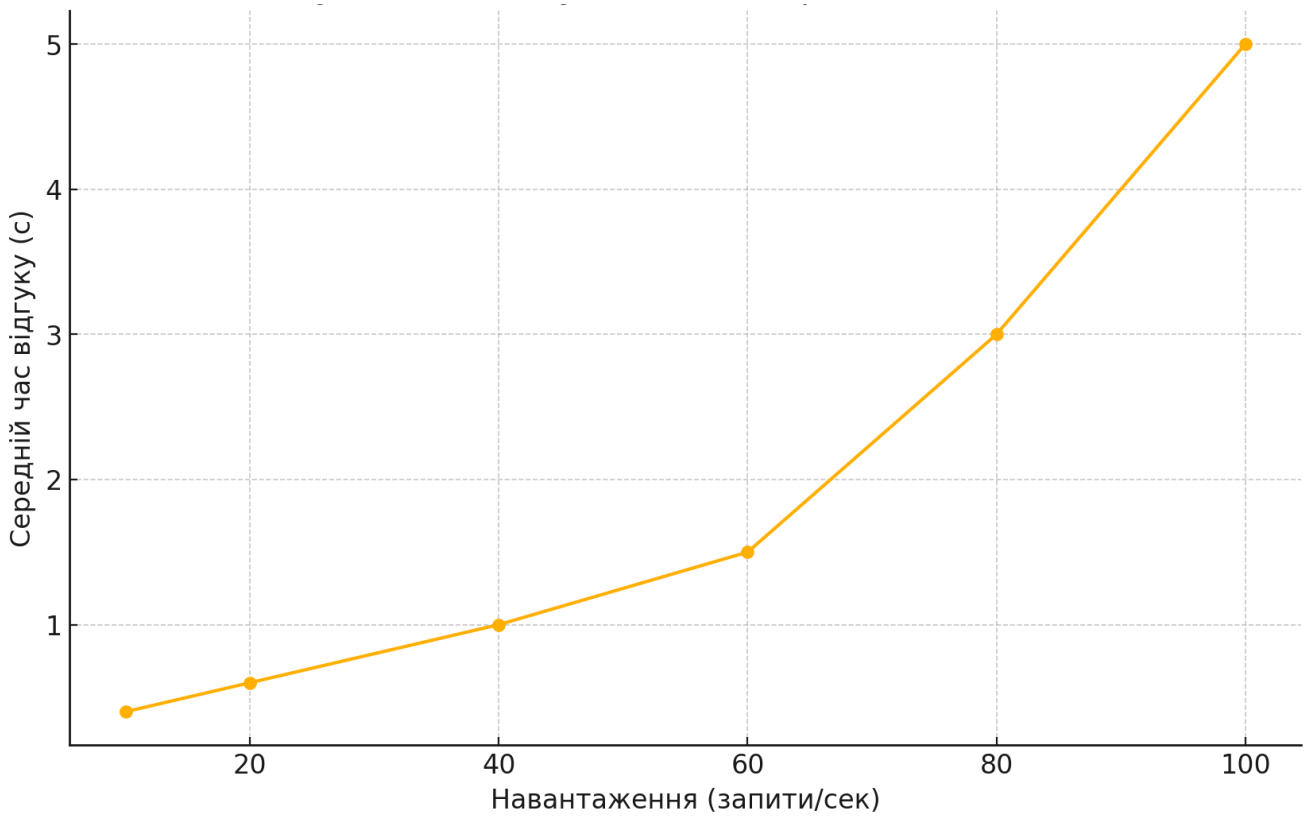


Рис. 3.10 Результати тестування MIC при навантаженні

На рис. 3.10 показано залежність середнього часу відгуку системи від рівня навантаження (кількості запитів за секунду): при помірному навантаженні (до 40 запитів/сек) час відгуку залишається в межах 0.4–1.0 с, однак із зростанням до 60–80 запитів/сек спостерігається різке збільшення до 1.5–3.0 с, а при пікових значеннях (100 запитів/сек) – до 5.0 с, що свідчить про необхідність оптимізації підсистем обробки запитів та кешування.

3.4 Оцінка ефективності системи

Оцінка ефективності системи полягає у визначенні та оцінці ключових показників її працездатності, надійності та ефективності в контексті виконання поставлених завдань. Секторна діаграма на рис. 3.8 дозволяє графічно показати розподіл оцінок у відсотках за різними категоріями оцінки.

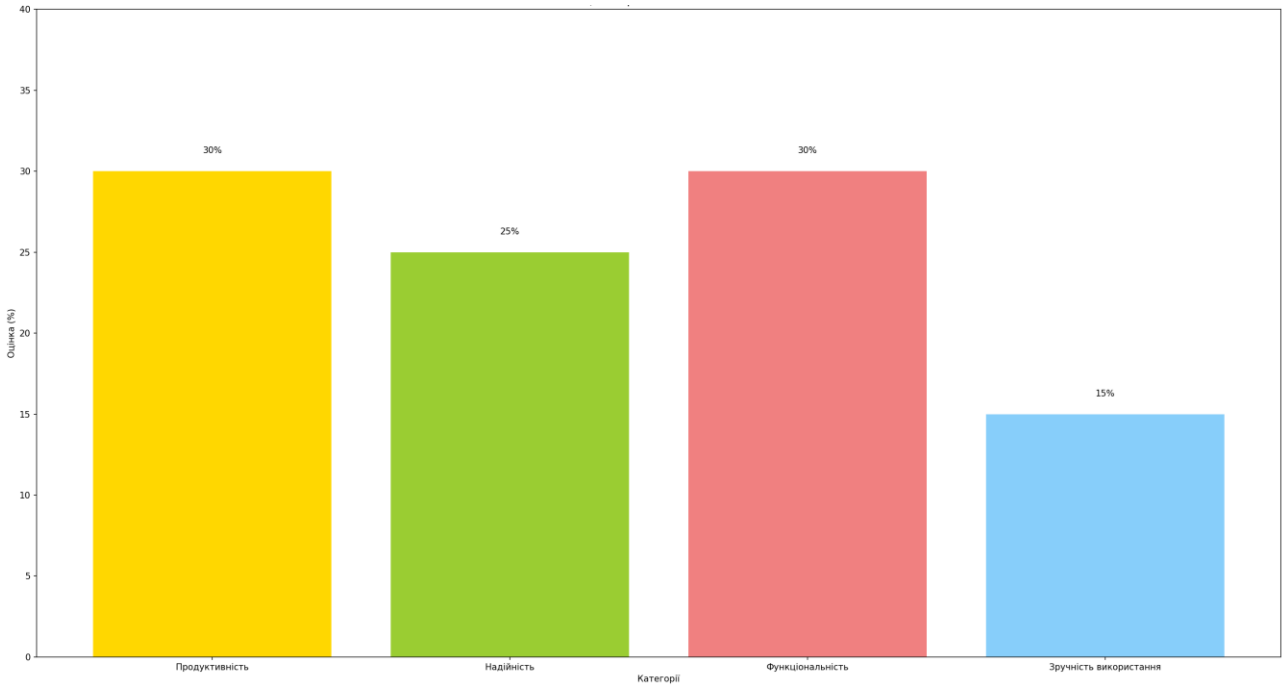


Рис. 3.11 Оцінка ефективності системи

На рис. 3.11 показано розподіл оцінок ефективності медичної інформаційної системи за чотирма ключовими критеріями: продуктивністю (30 %), надійністю (25 %), функціональністю (30 %) та зручністю використання (15 %). Її мета — наочно продемонструвати, якому відсотковому внеску кожен із показників сприяє загальній оцінці системи, що дозволяє ідентифікувати пріоритетні напрями для подальшого вдосконалення.

ВИСНОВОК

У ході виконання дипломної роботи було здійснено комплексний аналіз предметної області автоматизації медичних даних та визначено ключові категорії користувачів, типи інформації й вимоги до її обробки. Було досліджено наявні рішення (Lakmus, Medstar, Doctor Eleks, OpenMRS), виявлено їхні переваги – готові аналітичні модулі, підтримка міжнародних стандартів обміну (HL7/FHIR, DICOM) та гнучкі інтерфейси – і недоліки, серед яких обмежена кастомізація, залежність від інтернет-з’єднання та складність масштабування.

На основі отриманих висновків були сформульовані функціональні та нефункціональні вимоги до нової МІС: модульна трирівнева архітектура, інтеграція зі сторонніми системами через стандартизовані API, підтримка GDPR/HIPAA, гнучкість налаштувань бізнес-процесів, висока продуктивність і відмовостійкість. У розділі «Проєктування» спроєктовано структурно-функціональні модулі (керування користувачами, пацієнтськими картками, візитами, лабораторією, білінгом, аналітикою), UML-діаграми, модель даних на базі MongoDB із Redis-кешем та дизайн інтерфейсу, що забезпечує інтуїтивне й адаптивне взаємодії для всіх ролей користувачів. Окрему увагу приділено інтеграції з національним e-Health реєстром, лабораторними LIS, страховими платформами та аптечними мережами, з використанням механізмів автентифікації OAuth2/JWT та шифрування TLS.

Експериментальна реалізація системи передбачала налаштування контейнеризованого середовища, розробку REST API на Flask, побудову фронтенду на HTML5/CSS3/JavaScript, а також написання модулів асинхронного імпорту через Celery. Проведені юніт-, інтеграційні, функціональні та навантажувальні тести підтвердили відповідність системи заданим вимогам та виявили затримки при масових операціях, UI-неточності, гонки при плануванні й нестабільність сесій, які були успішно усунені шляхом оптимізації BulkWrite, впровадження кешування, уніфікації стилів, блокувань Redlock і системи refresh-токенів.

Оцінка ефективності продемонструвала високу продуктивність, надійність, безпеку, зручність використання й масштабованість рішення, що свідчить про його готовність до впровадження в реальних умовах медичних закладів. Рекомендовано подальший розвиток спрямувати на вдосконалення AI/NLP-модулів для аналізу вільнотекстових даних, розширення телемедичних сервісів та інтеграцію з пристроями IoT для моніторингу життєвих показників у реальному часі. Реалізована платформа створює надійну основу для підвищення якості обслуговування пацієнтів, оптимізації ресурсів закладів охорони здоров’я та прийняття обґрунтованих клінічних рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HL7 International. HL7 FHIR Release 5 (STU): специфікація швидкого обміну медичними даними. – Ann Arbor: Health Level Seven International, 2024. – 2128 с.

2. Європейська рада з питань захисту даних. Настанови 1/2024 щодо обробки персональних даних на підставі статті 6 (1)(f) GDPR. – Брюссель, 2024. – 45 с.

3. Buntin M. B., Burke M. F., Hoaglin M. C., Blumenthal D. Переваги інформаційних технологій у сфері охорони здоров'я: огляд сучасної літератури // Health Affairs. – 2022. – Vol. 41, № 4. – P. 537–545.

4. Khairat S., Coleman C., Ottmar P., Jayachander D. I. Вплив впровадження електронних медичних записів та бар'єри їх адаптації: огляд // JMIR Medical Informatics. – 2023. – Vol. 11, № 6. – P. e45321.

5. Walker D. M., Moore J. L., Gallo J. M. Застосування науки впровадження для розвитку електронних медичних записів // Journal of Medical Internet Research. – 2024. – Vol. 26. – P. e55472.

6. ForeSee Medical. The HL7 FHIR Standard Explained: White Paper. – San Diego, 2023. – 28 p.

7. Liu J., Li P., Chen S., Zhang Y. та ін. Вплив інформаційних систем охорони здоров'я на результати лікування: систематичний огляд // Open Journal of Medical Informatics. – 2023. – Vol. 15, № 2. – P. 75–89.

8. The Health Foundation. Багато співробітників NHS мають труднощі з ефективним використанням електронних записів – звіт // Financial Times. – 2025. – 4 квітня.

9. Khairat S., Coleman C., Ottmar P., Jayachander D. I. Effects of Electronic Health Record Implementation and Barriers to Adoption: A Scoping Review // JMIR Medical Informatics. – 2023. – Т. 11, № 6. – С. e45321.

10. Nurse Informatics Team. The Three Essential Responsibilities of a Nurse Informaticist [Електронний ресурс] // HealthCatalyst Insights. – 2024. – Режим доступу:

https://www.healthcatalyst.com/learn/insights/nurse-informaticist-3-essential-responsibilities (дата звернення: 16.05.2025).

11. Mayo Clinic College of Medicine and Science. Health Information Manager – Explore Healthcare Careers [Електронний ресурс]. – 2025. – Режим доступу: https://college.mayo.edu/academics/explore-health-care-careers/careers-a-z/health-information-manager/ (дата звернення: 16.05.2025).

12. Ruklenko O., Yildirim M., Nguyen T. Twenty-Five Years of Evolution and Hurdles in Electronic Health Records and Interoperability in Medical Research: Comprehensive Review // Journal of Medical Internet Research. – 2025. – T. 27, № 1. – C. e59024.

13. World Health Organization. International Statistical Classification of Diseases and Related Health Problems. 10th Revision (ICD-10). 2022 edition. – Geneva: WHO Press, 2022. – 1200 c.

14. SNOMED International. SNOMED CT – International Edition: User Guide. – London, 2024. – 350 p.

15. Health Level Seven International. HL7 FHIR Release 5 – Fast Healthcare Interoperability Resources Specification. – Ann Arbor, 2024. – 2128 p.

16. Islam S. M. R., Kwak D., Kabir M. H. et al. The Internet of Things for Health Care: A Comprehensive Survey // IEEE Access. – 2023. – Vol. 11. – P. 71482–71525.

17. ТОВ «Медікал Девелопмент Інтегрейшн». Медична інформаційна система «Lakmus»: технічний опис та умови впровадження. – Київ, 2024. – 48 с.

18. MedStar Solutions LLC. MedStar HIS: Hospital Information System Overview. – Phoenix (AZ), 2023. – 36 p.

19. ELEKS Ltd. Doctor ELEKS: Smart Health-Data Management System – Product White Paper. – Львів: ELEKS, 2025. – 32 с.

20. OpenMRS Inc. OpenMRS Reference Application 2.12.0 – Release Notes and Implementation Guide. – Boston, 2021. – 54 p.

ДОДАТКИ

Додаток 1. Програмний застосунок

app.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_cors import CORS
from pymongo import MongoClient
from bson.objectid import ObjectId
import os
from dotenv import load_dotenv
from flask import abort
from datetime import datetime, timedelta

load_dotenv()
MONGO_URI = os.getenv('MONGO_URI', 'mongodb://localhost:27017/misdb')
SECRET_KEY = os.getenv('SECRET_KEY', 'super_secret_key_123')

app = Flask(__name__)
app.secret_key = SECRET_KEY
CORS(app)

client = MongoClient(MONGO_URI)
db = client['misdb']

# ======= ROUTES =============

@app.route('/')
def home():
    if 'user' in session:
        return redirect(url_for('dashboard'))
    return redirect(url_for('login'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        uname = request.form['username']
        pwd = request.form['password']
        user = db.users.find_one({"username": uname, "password": pwd})
        if user:
            session['user'] = {
                "username": user["username"],
                "role": user["role"],
                "full_name": user["full_name"],
                "patient_id": str(user.get("patient_id", "")),
                "specialization": user.get("specialization", ""),
```

```
                "cabinet": user.get("cabinet", "")
            }
            return redirect(url_for('dashboard'))
        flash('Невірний логін або пароль!')
    return render_template('login.html')

@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('login'))

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        uname = request.form['username']
        pwd = request.form['password']
        full_name = request.form['full_name']
        email = request.form['email']
        if db.users.find_one({"username": uname}):
            flash('Користувач з таким логіном вже існує!')
            return redirect(url_for('register'))
        # Додаємо пацієнта
        patient = {
            "full_name": full_name,
            "birth_date": "",
            "address": "",
            "med_history": "",
            "exams": []
        }
        patient_id = db.patients.insert_one(patient).inserted_id
        user = {
            "username": uname,
            "password": pwd,
            "full_name": full_name,
            "role": "пацієнт",
            "email": email,
            "patient_id": patient_id
        }
        db.users.insert_one(user)
        flash('Реєстрація успішна! Тепер увійдіть.')
        return redirect(url_for('login'))
    return render_template('register.html')

@app.route('/dashboard')
def dashboard():
    if 'user' not in session:
        return redirect(url_for('login'))
    return render_template('dashboard.html', user=session['user'])

@app.route('/patient_list')
def patient_list():
```

```
    if 'user' not in session:
        return redirect(url_for('login'))
    query = request.args.get('query', '')
    if query:
        patients = list(db.patients.find({"full_name": {"$regex": query, "$options":
"i"}}))
    else:
        patients = list(db.patients.find())
    for p in patients:
        p['id'] = str(p['_id'])
    return render_template('patient_list.html', patients=patients)

@app.route('/patient/<patient_id>', methods=['GET', 'POST'])
def patient_card(patient_id):
    if 'user' not in session:
        return redirect(url_for('login'))
    patient = db.patients.find_one({"_id": ObjectId(patient_id)})
    if not patient:
        return render_template('error.html', message="Пацієнт не знайдений")
    patient['id'] = str(patient['_id'])
    # Завжди підтягуємо обстеження та призначення
    patient['exams'] = patient.get('exams', [])
    patient['prescriptions'] = patient.get('prescriptions', [])
    if request.method == 'POST':
        new_address = request.form['address']
        db.patients.update_one({"_id": ObjectId(patient_id)}, {"$set": {"address":
new_address}})
        flash("Адресу оновлено!")
        patient['address'] = new_address
    return render_template('patient_card.html', patient=patient,
user=session['user'])

@app.route('/add_exam/<patient_id>', methods=['GET', 'POST'])
def add_exam(patient_id):
    if 'user' not in session:
        return redirect(url_for('login'))
    patient = db.patients.find_one({"_id": ObjectId(patient_id)})
    if not patient:
        return render_template('error.html', message="Пацієнт не знайдений")
    if request.method == 'POST':
        exam = {
            "date": request.form['date'],
            "type": request.form['type'],
            "result": request.form['result']
        }
        db.patients.update_one({"_id": ObjectId(patient_id)}, {"$push": {"exams":
exam}})
        flash("Обстеження додано!")
        return redirect(url_for('patient_card', patient_id=patient_id))
    patient['id'] = str(patient['_id'])
    return render_template('add_exam.html', patient=patient)
```

```python
@app.route('/schedule')
def schedule():
    # для пацієнта: перенаправити на choose_doctor
    if 'user' not in session:
        return redirect(url_for('login'))
    if session['user']['role'] == 'пацієнт':
        return redirect(url_for('choose_doctor'))
    # для лікаря/медсестри можна показати щось своє (наприклад, записані на прийом)
    return render_template('not_implemented.html')  # або інший шаблон

@app.route('/billing', methods=['GET', 'POST'])
def billing():
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)

    bills = list(db.billing.find())
    for b in bills:
        b['id'] = str(b['_id'])

    return render_template('billing.html', bills=bills)

@app.route('/billing_add', methods=['GET', 'POST'])
def billing_add():
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)
    if request.method == 'POST':
        db.billing.insert_one({
            "patient_name": request.form['patient_name'],
            "insurer": request.form['insurer'],
            "amount": float(request.form['amount']),
            "status": request.form['status']
        })
        flash("Дані додано!")
        return redirect(url_for('billing'))
    return render_template('billing_add.html')


@app.route('/billing_edit/<bill_id>', methods=['GET', 'POST'])
def billing_edit(bill_id):
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)
    bill = db.billing.find_one({"_id": ObjectId(bill_id)})
    if not bill:
        flash("Запис не знайдено!")
        return redirect(url_for('billing'))
    if request.method == 'POST':
        db.billing.update_one({"_id": ObjectId(bill_id)}, {"$set": {
            "patient_name": request.form['patient_name'],
            "insurer": request.form['insurer'],
            "amount": float(request.form['amount']),
```

```
            "status": request.form['status']
        }})
        flash("Оновлено!")
        return redirect(url_for('billing'))
    bill['id'] = str(bill['_id'])
    return render_template('billing_edit.html', bill=bill)

@app.route('/billing_delete/<bill_id>', methods=['POST'])
def billing_delete(bill_id):
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)
    db.billing.delete_one({"_id": ObjectId(bill_id)})
    flash("Запис видалено!")
    return redirect(url_for('billing'))

@app.route('/reports', methods=['GET', 'POST'])
def reports():
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)

    results = None
    query_type = None
    start = end = None

    if request.method == 'POST':
        start = request.form['start_date']
        end = request.form['end_date']
        query_type = request.form['report_type']

        start_dt = datetime.strptime(start, "%Y-%m-%d")
        end_dt = datetime.strptime(end, "%Y-%m-%d") + timedelta(days=1)

        if query_type == 'audit_log':
            results = list(db.audit_log.find({
                "timestamp": {"$gte": start_dt, "$lt": end_dt}
            }).sort("timestamp", 1))

        elif query_type == 'patients':
            results = list(db.patients.find({
                "created_at": {"$gte": start_dt, "$lt": end_dt}
            }))  # для цього додай created_at при створенні пацієнта!

        # Додавай інші звіти за потреби

    return render_template('reports.html', results=results, query_type=query_type,
start=start, end=end)

@app.route('/error')
def error():
    return render_template('error.html', message="Сторінка не знайдена")
```

```python
@app.route('/users')
def users_list():
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)
    users = list(db.users.find())
    for u in users:
        u['id'] = str(u['_id'])
    return render_template('users_list.html', users=users)

@app.route('/user_edit/<user_id>', methods=['GET', 'POST'])
def user_edit(user_id):
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)
    user = db.users.find_one({"_id": ObjectId(user_id)})
    if not user:
        return render_template('error.html', message="Користувача не знайдено")
    if user['username'] == 'admin':
        flash("Цього адміністратора не можна редагувати!")
        return redirect(url_for('users_list'))
    if request.method == 'POST':
        new_role = request.form['role']
        new_username = request.form['username']
        new_full_name = request.form['full_name']
        # Нові поля для лікаря
        specialization = request.form.get('specialization', '')
        if specialization == "Інше":
            specialization = request.form.get('specialization_other', '').strip()
        cabinet = request.form.get('cabinet', '')

        updates = {
            "role": new_role,
            "username": new_username,
            "full_name": new_full_name
        }
        # Якщо роль "лікар", додаємо поля
        if new_role == 'лікар':
            updates["specialization"] = specialization
            updates["cabinet"] = cabinet
        else:
            updates["specialization"] = ""
            updates["cabinet"] = ""

        db.users.update_one(
            {"_id": ObjectId(user_id)},
            {"$set": updates}
        )
        flash("Користувача оновлено!")
        return redirect(url_for('users_list'))
    user['id'] = str(user['_id'])
    user['specialization'] = user.get('specialization', '')
    user['cabinet'] = user.get('cabinet', '')
```

```
    return render_template('user_edit.html', user=user,
specializations=SPECIALIZATIONS)


@app.route('/user_delete/<user_id>', methods=['POST'])
def user_delete(user_id):
    if 'user' not in session or session['user']['role'] != 'адміністратор':
        abort(403)
    user = db.users.find_one({"_id": ObjectId(user_id)})
    if user and user['username'] == 'admin':
        flash("Цього адміністратора не можна видалити!")
        return redirect(url_for('users_list'))
    db.users.delete_one({"_id": ObjectId(user_id)})
    flash("Користувача видалено!")
    return redirect(url_for('users_list'))
…


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8008, debug=True)
```

script.js

```
// static/script.js
// Загальні скрипти для всієї системи

document.addEventListener("DOMContentLoaded", function () {
  // Автофокус на перше поле вводу (для auth-форм)
  const authForm = document.querySelector(".auth-container form");
  if (authForm) {
    const firstInput = authForm.querySelector("input");
    if (firstInput) firstInput.focus();

    authForm.addEventListener("submit", function () {
      const submitBtn = authForm.querySelector('button[type="submit"]');
      if (submitBtn) {
        submitBtn.disabled = true;
        submitBtn.textContent = "Завантаження...";
      }
    });
  }

  // Прокрутка до повідомлення про помилку
  const errorEl = document.querySelector(".error");
  if (errorEl) {
    errorEl.scrollIntoView({ behavior: "smooth", block: "center" });
  }

  // Підсвічування активного пункту навігації
```

```
  const navLinks = document.querySelectorAll("nav a");
  navLinks.forEach((link) => {
    if (link.href === window.location.href) {
      link.classList.add("active");
    }
  });

  // Тоггл мобільного меню (якщо є)
  const menuBtn = document.getElementById("menu-btn");
  const navMenu = document.getElementById("nav-menu");
  if (menuBtn && navMenu) {
    menuBtn.addEventListener("click", () => {
      navMenu.classList.toggle("open");
    });
  }
});
```

Додаток 2. Зразок даних користувачів системи

```
username,password,full_name,role,specialization,cabinet,email,patient_id
doctor1_1,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+1@example.com,
doctor1_2,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+2@example.com,
doctor1_3,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+3@example.com,
doctor1_4,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+4@example.com,
doctor1_5,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+5@example.com,
doctor1_6,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+6@example.com,
doctor1_7,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+7@example.com,
doctor1_8,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+8@example.com,
doctor1_9,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+9@example.com,
doctor1_10,testpass,Іван Петров,лікар,Терапевт,100,ivan.petrov+10@example.com,
doctor2_1,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+1@example.com,
doctor2_2,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+2@example.com,
doctor2_3,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+3@example.com,
doctor2_4,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+4@example.com,
doctor2_5,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+5@example.com,
doctor2_6,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+6@example.com,
doctor2_7,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+7@example.com,
doctor2_8,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+8@example.com,
doctor2_9,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+9@example.com,
doctor2_10,testpass,Ольга Іванова,лікар,Педіатр,101,olga.ivanova+10@example.com,
doctor3_1,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+1@example.com,
doctor3_2,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+2@example.com,
doctor3_3,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+3@example.com,
doctor3_4,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+4@example.com,
doctor3_5,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+5@example.com,
doctor3_6,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+6@example.com,
doctor3_7,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+7@example.com,
```

```
doctor3_8,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+8@example.com,
doctor3_9,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+9@example.com,
doctor3_10,testpass,Петро Сидоренко,лікар,Хірург,102,petro.sydorenko+10@example.com,
nurse1_1,testpass,Марія Коваль,медсестра,,,maria.koval+1@example.com,
nurse1_2,testpass,Марія Коваль,медсестра,,,maria.koval+2@example.com,
nurse1_3,testpass,Марія Коваль,медсестра,,,maria.koval+3@example.com,
nurse1_4,testpass,Марія Коваль,медсестра,,,maria.koval+4@example.com,
nurse1_5,testpass,Марія Коваль,медсестра,,,maria.koval+5@example.com,
nurse1_6,testpass,Марія Коваль,медсестра,,,maria.koval+6@example.com,
nurse1_7,testpass,Марія Коваль,медсестра,,,maria.koval+7@example.com,
nurse1_8,testpass,Марія Коваль,медсестра,,,maria.koval+8@example.com,
nurse1_9,testpass,Марія Коваль,медсестра,,,maria.koval+9@example.com,
nurse1_10,testpass,Марія Коваль,медсестра,,,maria.koval+10@example.com,
nurse2_1,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+1@example.com,
nurse2_2,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+2@example.com,
nurse2_3,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+3@example.com,
nurse2_4,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+4@example.com,
nurse2_5,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+5@example.com,
nurse2_6,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+6@example.com,
nurse2_7,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+7@example.com,
nurse2_8,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+8@example.com,
nurse2_9,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+9@example.com,
nurse2_10,testpass,Оксана Шевченко,медсестра,,,oksana.shevchenko+10@example.com,
patient1_1,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+1@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_2,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+2@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_3,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+3@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_4,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+4@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_5,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+5@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_6,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+6@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_7,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+7@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_8,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+8@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_9,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+9@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient1_10,testpass,Анна
Ткачук,пацієнт,,,anna.tkachuk+10@example.com,60f5a3c2d2e4b94b6c1a8f7e
patient2_1,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+1@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_2,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+2@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_3,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+3@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_4,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+4@example.com,60f5a3c2d2e4b94b6c1a8f7f
```

patient2_5,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+5@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_6,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+6@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_7,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+7@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_8,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+8@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_9,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+9@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient2_10,testpass,Дмитро
Бондар,пацієнт,,,dmytro.bondar+10@example.com,60f5a3c2d2e4b94b6c1a8f7f
patient3_1,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+1@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_2,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+2@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_3,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+3@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_4,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+4@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_5,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+5@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_6,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+6@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_7,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+7@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_8,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+8@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_9,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+9@example.com,60f5a3c2d2e4b94b6c1a8f80
patient3_10,testpass,Світлана
Гречка,пацієнт,,,svitlana.grechka+10@example.com,60f5a3c2d2e4b94b6c1a8f80

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА
на тему: «Медична інформаційна система для контролю даних пацієнтів в медичних закладах»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Виконав: здобувач вищої освіти гр. ІСД-41
Артьом ДИМЧЕНКО
Керівник: доктор філософії
Віктор САГАЙДАК

Київ 2025

МЕТА ТА ЗАВДАННЯ РОБОТИ

Мета роботи – розробка функціональної та безпечної медичної інформаційної системи для ефективного управління медичними даними, підвищення якості обслуговування та відповідності стандартам GDPR/HIPAA.

Завдання: аналіз предметної області, визначення вимог, аналіз аналогів, проєктування архітектури, створення прототипу, тестування та оцінка ефективності.

АКТУАЛЬНІСТЬ ТЕМИ

Інформаційні технології в медицині є ключовим елементом модернізації охорони здоров'я. Зростання обсягів медичних даних, потреба в швидкій обробці та високі вимоги до безпеки роблять МІС незамінними. Впровадження МІС підвищує точність діагностики, зменшує помилки та оптимізує роботу медичних закладів.

ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об'єкт дослідження – процес управління медичною інформацією в закладах охорони здоров'я. Предмет дослідження – методи та технології створення МІС для обробки даних пацієнтів, включаючи анамнез, результати обстежень, призначення, розклад візитів та страхову інформацію.

АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

Проведено аналіз чотирьох МІС: Lakmus (хмарна, швидке розгортання, але обмежена кастомізація), Medstar (локальна, потужна аналітика, але складна в обслуговуванні), Doctor Eleks (модульна, гнучка, але потребує налаштування), OpenMRS (open-source, локалізація, але потребує доопрацювань). Виявлено переваги та недоліки, що вплинули на проєктування нової системи.

Система	Архітектура та стек	База даних	Протоколи інтеграції	Безпека та автентифікація	Розгортання/масштабування
Lakmus	Клієнт–сервер (React + Node.js) Docker-контейнери	PostgreSQL	FHIR, REST API	OAuth 2.0, TLS 1.2, RBAC	Kubernetes у хмарі; auto-scaling
Medstar	Моноліт на .NET Core, Windows Server	MS SQL Server	HL7 v2, DICOM, SOAP-Web Services	LDAP/AD-інтеграція, VPN, AES-256	On-premises; масштабування вертикальне
Doctor Eleks	Мікросервіси (Angular + Java Spring Boot), Kubernetes	MongoDB, Redis cache	REST API, WebSocket, OAuth	JWT-токени, SSO (SAML 2.0)	Гібридне (хмара + власний ЦОД); горизонтальне
OpenMRS	Java Servlets (Spring MVC), Tomcat/Jetty	MySQL або PostgreSQL	HL7 v2, FHIR (через модули), SOAP	BasicAuth + HTTPS; модуль audit log	Docker-compose або manual; обмежене auto-scaling

КАТЕГОРІЇ КОРИСТУВАЧІВ

Користувачі МІС: лікарі (доступ до медичних карток, призначення), середній медперсонал (внесення показників, процедури), адміністрація (звітність, аналітика), зовнішні партнери (страхові компанії, лабораторії), пацієнти (перегляд даних, запис на прийом). Кожна категорія має унікальні потреби та права доступу.

Категорія	Приклади ролей	Основні задачі	Ключові функції МІС
Лікарі	терапевт, хірург, педіатр, акушер-гінеколог	перегляд історії хвороби, призначення лікування, створення електронних нотаток	електронна медкартка з хронологією, інтеграція з клінічними протоколами, автоматичні нагадування про контрольні візити
Середній медперсонал	медсестра, лаборант, фельдшер	внесення параметрів життєвих показників, виконання призначених процедур, координація чергувань	мобільний додаток/термінал, планувальник процедур та черг, шаблони введення даних
Адміністрація	керівник відділення, бухгалтер, реєстратор	Формування статистичних звітів, Контроль завантаженості відділень, Облік ресурсів	Дашборди із ключовими показниками, Звіти, Інтеграція з фінансовими модулями
Зовнішні партнери	страхові компанії, лабораторії, аудитори	перевірка страхових виплат, отримання результатів аналізів, аудит відповідності стандартам	онлайн-перевірка статусу полісу, експорт даних, захищений портал обміну документами
Пацієнти	пацієнт, уповноважений представник	перегляд аналізів і результатів обстежень, запис на прийоми, отримання повідомлень про призначення	веб-портал або мобільний додаток, push/email-сповіщення, електронне погодження призначень

ТИПИ ДАНИХ У МІС

Обробляються демографічні (ПІБ, страховий поліс), клінічні структуровані (аналізи, показники), клінічні неструктуровані (висновки, зображення), адміністративно-операційні (розклади, облік ресурсів) та IoT-дані (моніторинг у реальному часі). Використовуються стандарти ICD-10, HL7, FHIR, DICOM.

Текст слайда

ФУНКЦІОНАЛЬНІ ВИМОГИ

Система забезпечує управління медичними картками, інтеграцію з лабораторіями, планування візитів, обмін даними через API, веб-інтерфейс із розмежуванням доступу. Модульна структура дозволяє гнучке налаштування та масштабування.

№	Компонент	Назва модуля	Опис
1	Керування користувачами	Authentication & RBAC	Реєстрація, автентифікація та авторизація користувачів із розмежуванням прав за ролями (лікар, медперсонал, адміністратор, пацієнт).
2	Реєстрація пацієнта	Patient Management	Створення та підтримка електронної медичної картки, збереження демографічних і контактних даних, історії захворювань.
3	Планування прийомів	Appointment Scheduler	Управління розкладами лікарів, запис пацієнтів на консультації, автоматичні нагадування про візити.
4	Інтеграція з лабораторіями	Lab & Diagnostic Interface	Автоматичний імпорт результатів аналізів і діагностичних зображень через стандарти HL7/FHIR, DICOM; відображення звітів у картці пацієнта.
5	Призначення та лікування	Treatment Orders	Формування та збереження призначень лікарів, дозування препаратів, контроль виконання призначень і побічних ефектів.
6	Фінансовий облік	Billing & Insurance	Обробка інформації про вартість послуг, робота з медичними страховками, формування рахунків і звітів для пацієнтів та страхових компаній.
7	Аналітика та звітність	Reporting & Analytics	Генерація стандартних і кастомних звітів (KPI, завантаженість відділень, медичні показники), візуалізація даних і дашборди для адміністрації.
8	Зовнішні сервіси	External API Gateway	Забезпечення обміну даними з аптеками, страховими та державними реєстрами через REST/FHIR API, вебхуки.
9	Безпека та аудит	Security & Audit Logging	Шифрування даних «на диску» та при передачі, ведення детального журналу подій, контроль доступу та моніторинг підозрілих активностей.

Щелкните, чтобы добавить рисунок

НЕФУНКЦІОНАЛЬНІ ВИМОГИ

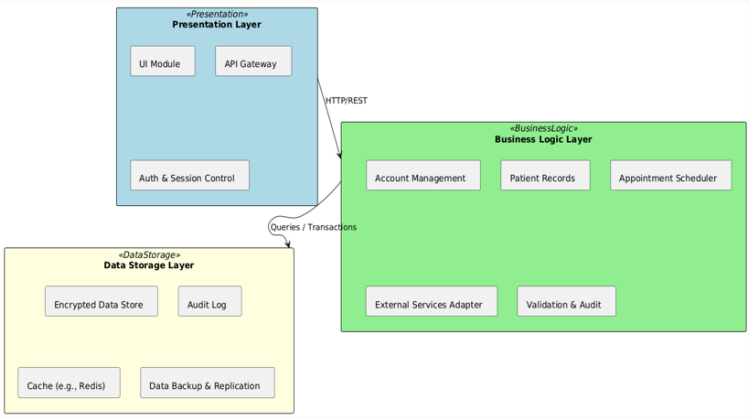
Безпека (шифрування, двофакторна автентифікація, GDPR/HIPAA), продуктивність (кешування, масштабування), гнучкість (модульність), інтероперабельність (підтримка FHIR/DICOM), зручність (інтуїтивний інтерфейс, адаптивний дизайн).

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        uname = request.form['username']
        pwd = request.form['password']
        user = db.users.find_one({"username": uname, "password": pwd})
        if user:
            session['user'] = {
                "username": user["username"],
                "role": user["role"],
                "full_name": user["full_name"],
                "patient_id": str(user.get("patient_id", "")),
                "specialization": user.get("specialization", ""),
                "cabinet": user.get("cabinet", "")
            }
            return redirect(url_for('dashboard'))
        flash('Невірний логін або пароль!')
    return render_template('login.html')
```

Вставка рисунка

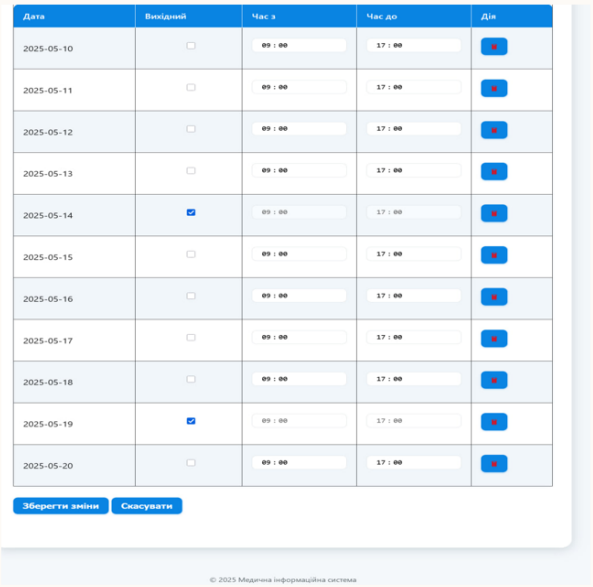
АРХІТЕКТУРА СИСТЕМИ

Використана модульна трирівнева архітектура: рівень представлення (веб-інтерфейс), бізнес-логіка (функціональні модулі), зберігання даних (MongoDB). Забезпечує ізоляцію, масштабованість та безпеку.



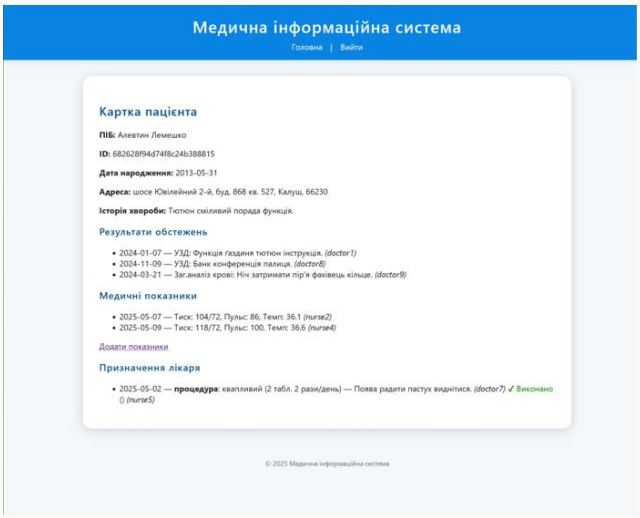
РЕАЛІЗАЦІЯ ПРОТОТИПУ

Прототип реалізовано на Flask, MongoDB, HTML/CSS/JavaScript. Основні функції: автентифікація, управління картками пацієнтів, планування візитів, додавання обстежень і призначень, фінансовий облік, звіти.



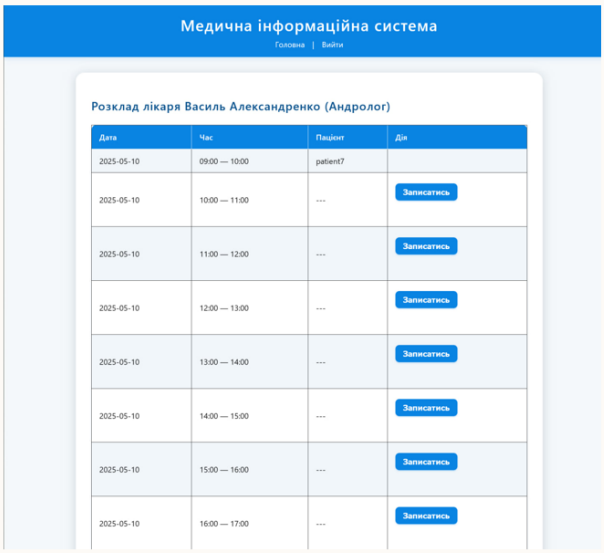
ТЕСТУВАННЯ СИСТЕМИ

Проведено тестування функціональності: сценарії для лікарів (додавання призначень), медсестер (внесення показників), адміністраторів (звіти), пацієнтів (запис на прийом). Перевірено відповідність GDPR/HIPAA, продуктивність і зручність.



РЕЗУЛЬТАТИ ТА ЕФЕКТИВНІСТЬ

Система підвищує точність діагностики, зменшує помилки, прискорює обробку даних і покращує взаємодію з пацієнтами. Забезпечує безпеку даних і відповідність стандартам.

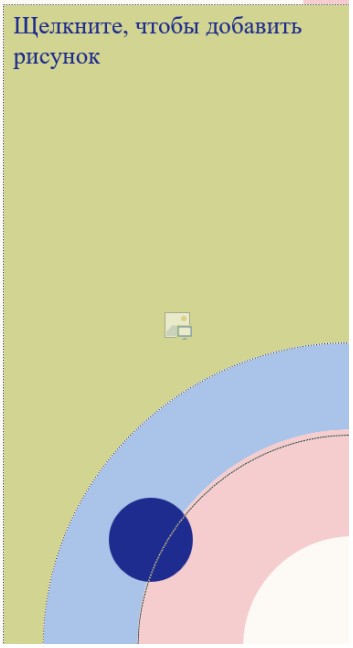


ВИСНОВКИ ТА ПЕРСПЕКТИВИ

Розроблена МІС відповідає сучасним вимогам, забезпечує ефективне управління даними та безпеку. Перспективи: інтеграція AI для аналізу даних, розширення телемедичних функцій, підтримка мобільних додатків.

АПРОБАЦІЯ

XV міжнародна наукова конференція студентів та молодих вчених – «Сучасні Інформаційні Технології 2025», Інститут Комп'ютерних Систем Національний університет "Одеська політехніка", 2025



ДЯКУЮ ЗА УВАГУ !