НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва кафедри)

«До захисту допущено»
В.о.завідувачки кафедри БМК

_____ Світлана АЛХІМОВА
(підпис) (ініціали, ПРІЗВИЩЕ)

"___" __*червня*__ 2025р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп'ютерні технології в біології та медицині»
зі спеціальності *122 «Комп'ютерні науки»*

на тему***:*** ***Веб-застосунок для керування медичною документацією та зображеннями в DICOM-стандарті***

Виконав: студент **IV** курсу, групи БС-15

ВІЧЕПОЛЬСЬКИЙ ЮРІЙ ЮРІЙОВИЧ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник: *ст. вик. каф. біомедичної кібернетики (БМК) Корнієнко Галина Альбертівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по ініціали) (підпис)

Консультант з розділів дипломної роботи:
проф. каф. БМК, проф., д.м.н., Коваленко Олександр Сергійович _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент: *асистент каф. біомедичної інженерії, Попов Станіслав Володимирович* _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»
(код і назва)

ЗАТВЕРДЖУЮ
В.о.завідувачки кафедри БМК
_____ Світлана АЛХІМОВА
«____» *квітень* 2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

ВІЧЕПОЛЬСЬКОМУ ЮРІЮ ЮРІЙОВИЧУ

(прізвище, ім'я, по батькові)

1. Тема роботи **Веб-застосунок для керування медичною документацією та зображеннями в DICOM-стандарті**

керівник роботи
Корнієнко Галина Альбертівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26» *травня* 2025 р. №***1758-вс***

2. Термін подання студентом роботи ***22-23 травня 2025 року***
3. Вихідні дані: *медичні DICOM зображення (джерело MAGNETOM World SIEMENS Healthineers), база даних MongoDB.*
4. Зміст роботи: *Проаналізувати джерела з існуючих підходів обробки медичними зображеннями у форматі DICOM. Проаналізувати аналоги існуючих аналогів та методів керування медичною документацією та зображеннями в DICOM стандарті. Проаналізувати та обрати засоби реалізації веб-застосунку та засоби захисту інформації. Зробити загальні висновки та пропозиції з подальшого розвитку та вдосконалення.*
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) *Презентація з захисту дипломної роботи на 15-25 слайдах*
6. Консультанти розділів роботи*

* Якщо визначені консультанти. Консультантом не може бути зазначено наукового керівника магістерської дисертації.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Дипломної роботи	Коваленко Олександр Сергійович, проф., д.м.н., проф. каф. БМК	07.04.25	17.05.25

7. Дата видачі завдання ***06 квітня 2025р.***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 24.12.2024р.	***виконано***
2	Переддипломна практика. Виконання практичної частини ДР та додаткових розділів	1-4 тиждень за графіком практики	***виконано***
3	Виконання основних розділів ДР (Вступ, Розділи ДР: «Огляд теоретичних основ обробки медичних зображень у форматі DICOM», «Аналіз сучасних програмних середовищ для керування медичною документацією та зображеннями в DICOM стандарті», «Засоби реалізації програмного забезпечення веб-застосунку для керування медичною документацією та зображеннями в DICOM стандарті», «Інформаційне забезпечення. Програмна реалізація та методика роботи веб-застосунку», «Захист інформації»)	Протягом усієї практики	***виконано***
4	Перевірка ДР науковим керівником	до 20.05.2025	***виконано***
5	Подання в електронному вигляді ДР на перевірку нормоконтролера та плагіат (StrikePlagiarism.com). Доопрацювання ДР	Не пізніше 23.05.2025	***виконано***
6	Отримати відгук від керівника ДР	Не пізніше 05.06.2025	***виконано***
7	Надати пакету документів по ДР та супровідних до неї документів на засідання кафедри		***виконано***
8	Подання ДР рецензенту. Отримання рецензії.	09.06 – 13.06.2025	
9	Захист ДР в ЕК	*16.06 – 21.06.2025*	

Студент _____ **Юрій ВІЧЕПОЛЬСЬКИЙ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота за темою «Веб-застосунок для керування медичною документацією та зображеннями в DICOM-стандарті» виконана студентом кафедри біомедичної кібернетики ФБМІ *Вічепольським Юрієм Юрійовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; п'яти розділів («Огляд теоретичних основ обробки медичних зображень у форматі DICOM», «Аналіз сучасних програмних середовищ для керування медичною документацією та зображеннями в DICOM стандарті», «Засоби реалізації програмного забезпечення веб-застосунку для керування медичною документацією та зображеннями в DICOM стандарті», «Інформаційне забезпечення. Програмна реалізація та методика роботи веб-застосунку», «Захист інформації»), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 55 джерела, та додатків. Загальний обсяг роботи 88 сторінок.

Актуальність теми. Необхідність підвищення ефективності роботи медичних працівників шляхом оптимізації процесів керування великими обсягами медичної документації та діагностичних зображень є надзвичайно важливою. Сучасні медичні заклади активно впроваджують цифрові технології, і стандарт DICOM є ключовим для обміну та зберігання медичних зображень. Розробка спеціалізованого веб-застосунку дозволить централізувати доступ до даних, спростити їх обробку та аналіз, а також покращити взаємодію між лікарями, консультантами та пацієнтами, що безпосередньо впливає на якість та швидкість надання медичної допомоги.

Мета і завдання роботи.

Метою роботи є підвищення ефективності роботи медичних працівників з оптимізації процесів керування медичною документацією та медичними зображеннями у стандарті DICOM за допомогою веб-застосунку.

Її досягнення передбачає вирішення наступних *завдань*:

1. Проаналізувати теоретичні основи представлення та обробки медичних зображень у стандарті DICOM.
2. Провести аналіз сучасних веб-технологій та інструментів для обґрунтування вибору стеку для реалізації веб-застосунку.
3. Спроектувати архітектуру веб-застосунку, включаючи структуру бази даних.
4. Реалізувати серверну частину для обробки бізнес-логіки та взаємодії з базою даних і PACS-сервером.
5. Створити клієнтську частину з адаптивними інтерфейсами для різних ролей користувачів.
6. Інтегрувати функціонал перегляду DICOM-зображень у веб-браузері.
7. Забезпечити взаємодію веб-застосунку з PACS-сервером.
8. Реалізувати систему автентифікації та розмежування прав доступу.
9. Розглянути аспекти захисту інформації в системі.
10. Провести тестування основних модулів та продемонструвати сценарії роботи.

Використані методи. У роботі застосовано методи аналізу науково-технічної літератури та існуючих рішень, системний аналіз для визначення вимог до програмного продукту, сучасні методи веб-розробки для реалізації клієнтської та серверної частин застосунку, методи тестування програмного забезпечення для перевірки функціональності та надійності, а також метод функціонально-вартісного аналізу для оцінки економічної ефективності проєкту.

Отримані результати. Розроблено веб-застосунок, який надає можливості для ефективного керування медичною документацією та DICOM-зображеннями. Реалізовано розмежування доступу для ролей адміністратора, лікаря, консультанта та пацієнта. Забезпечено інтеграцію з

PACS-сервером та вбудовано DICOM-переглядач.

Апробація результатів роботи. *Не заплановано.*

Публікації. *Не заплановано.*

Ключові слова. DICOM, PACS, медичні зображення, комп’ютерна томографія, веб-застосунок.

Бібліографічний опис ДР

Вічепольський Ю. Ю. Веб-застосунок для керування медичною документацією та зображеннями в DICOM-стандарті : дипломна роб. бакалавра : 122 Комп’ютері науки / Вічепольський Юрій Юрійович. – Київ, 2025. – 107 с.

ANNOTATION

The bachelor's thesis titled "Web Application for Managing Medical Documentation and DICOM Images" was completed by Yurii Yuriiovych Vichepolskyi, a student of the Department of Biomedical Cybernetics at the Faculty of Biomedical Engineering, specializing in 122 "Computer Science" under the educational and professional program "Computer Technologies in Biology and Medicine." The work consists of: an introduction; 5 chapters ("Overview of Theoretical Foundations for Processing Medical Images in the DICOM Format," "Analysis of Modern Software Environments for Managing Medical Documentation and Images in the DICOM Standard," "Tools for Implementing the Web Application for Managing Medical Documentation and Images in the DICOM Standard," "Information Support. Software Implementation and Operating Methodology of the Web Application," "Information Security"). conclusions to each of these chapters, general conclusions, a list of references (55 sources), and appendices. The total volume of the work is 88 pages.

Relevance of the topic. The need to improve the efficiency of healthcare professionals by optimizing the management of large volumes of medical documentation and diagnostic images is extremely important. Modern medical institutions are actively implementing digital technologies, and the DICOM standard is key for the exchange and storage of medical images. The development of a specialized web application will allow centralized access to data, simplify their processing and analysis, and enhance interaction between doctors, consultants, and patients, which directly affects the quality and speed of medical care.

Purpose and objectives of the thesis.

The aim of the thesis is to enhance the efficiency of healthcare professionals by optimizing the processes of managing medical documentation and medical images in the DICOM standard through the use of a web application.

To achieve this goal, the following objectives were set:

1. Analyze the theoretical foundations of representation and processing of

medical images in the DICOM standard.

2. Conduct an analysis of modern web technologies and tools to justify the choice of a technology stack for the implementation of the web application.
3. Design the architecture of the web application, including the database structure.
4. Implement the server side to handle business logic and interact with the database and PACS server.
5. Create a client side with adaptive interfaces for different user roles.
6. Integrate functionality for viewing DICOM images in a web browser.
7. Ensure interaction between the web application and the PACS server.
8. Implement an authentication system and access control.
9. Address information security aspects in the system.
10. Test the main modules and demonstrate usage scenarios.

Methods used. The thesis applies methods of analysis of scientific and technical literature and existing solutions, systems analysis to determine software requirements, modern web development methods for implementing the client and server parts of the application, software testing methods to verify functionality and reliability, and functional-cost analysis to evaluate the economic efficiency of the project.

Results obtained. A web application has been developed that enables effective management of medical documentation and DICOM images. Role-based access control has been implemented for administrators, doctors, consultants, and patients. Integration with a PACS server has been ensured, and a DICOM viewer has been embedded.

Approbation of the results. *Not planned.*

Publications. *Not planned.*

Keywords. DICOM, PACS, medical images, computed tomography, web application.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 12

ВСТУП 13

РОЗДІЛ 1 ОГЛЯД ТЕОРЕТИЧНИХ ОСНОВ ОБРОБКИ МЕДИЧНИХ ЗОБРАЖЕНЬ У ФОРМАТІ DICOM 17

1.1 Загальна характеристика стандарту DICOM 17

1.2 Структура та представлення даних у DICOM 18

1.2.1 Інформаційна модель: від пацієнта до зображення 18

1.2.2 Елементи даних DICOM – атрибути 19

1.2.3 Стиснення даних зображень у DICOM 19

1.3 Структура DICOM-файлу 20

1.4 Візуалізація та інтеграція DICOM-даних 20

1.4.1 Стандартне відображення відтінків сірого 20

1.4.2 Засоби перегляду DICOM 20

1.5 Системи архівації та передачі зображень 21

1.6 Забезпечення інтероперабельності 22

Висновки до розділу 1 22

РОЗДІЛ 2 АНАЛІЗ СУЧАСНИХ ПРОГРАМНИХ СЕРЕДОВИЩЬ ДЛЯ КЕРУВАННЯ МЕДИЧНОЮ ДОКУМЕНТАЦІЄЮ ТА ЗОБРАЖЕННЯМИ В DICOM-СТАНДАРТІ 23

2.1 Аналіз сучасних веб-застосунків у медичній галузі 23

2.1.1 Тенденції розвитку 23

2.1.2 Проблематика та виклики 25

2.2 Аналіз існуючих програмних рішень 27

2.2.1 Helsi 27

2.2.2 Ambra Health 29

2.2.3 OHIF Viewer *32*

Висновки до розділу 2 34

РОЗДІЛ 3 ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ МЕДИЧНОЮ ДОКУМЕНТАЦІЄЮ ТА ЗОБРАЖЕННЯМИ В DICOM-СТАНДАРТІ 36

3.1 Аналіз та обґрунтування вибору програмних технологій та інструментів 36

3.1.1 Мова програмування TypeScript *36*

3.1.2 Вибір технологій для розробки інтерфейсу користувача. React та Next.js *39*

3.1.3 CSS-фреймворк Tailwind CSS *44*

3.1.4 Система керування базами даних MongoDB *48*

3.1.5 Рішення для автентифікації NextAuth *51*

3.2 Технології для реалізації DICOM-переглядача Cornerstone.js та аналіз альтернатив 54

3.2.1 Обґрунтування вибору бібліотеки Cornerstone.js *55*

3.2.2 Порівняльний аналіз DICOM-переглядачів *56*

3.3 Аналіз та обґрунтування вибору PACS-серверу 58

3.3.1 Обґрунтування вибору Orthanc *59*

3.3.2 Порівняльний аналіз PACS-серверів *61*

Висновки до розділу 3 62

РОЗДІЛ 4 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ВЕБ-ЗАСТОСУНКУ 64

4.1 Діаграма бази даних веб-застосунку 64

4.2 Реалізація ключових функціональних модулів 66

4.2.1 Система автентифікації та розмежування доступу *66*

4.2.2 Модуль управління пацієнтами *72*

4.2.3 Модуль управління візитами *75*

4.2.4 Модуль завантаження DICOM-досліджень *78*
4.2.5 Інтегрований DICOM-переглядач *78*
4.2.6 Модуль консультацій *81*
4.2.7 Кабінет пацієнта *85*
4.3 Демонстрація сценаріїв роботи користувачів 87
4.3.1 Сценарій 1: робота лікаря *87*
4.3.2 Сценарій 2: робота консультанта *88*
4.3.3 Сценарій 3: доступ пацієнта *89*
4.4 Розрахунок економічного ефекту 89
Висновки до розділу 4 91
РОЗДІЛ 5 ЗАХИСТ ІНФОРМАЦІЇ 93
5.1 Технічні засоби захисту інформації в системі 93
5.2 Організаційно-правові аспекти та політика безпеки для користувачів 95
Висновки до розділу 5 96
ЗАГАЛЬНІ ВИСНОВКИ 97
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 100

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DICOM – Digital Imaging and Communications in Medicine

PACS – Picture Archiving and Communication System

HL7 – Health Level 7

FHIR – Fast Healthcare Interoperability Resources

CDA – Clinical Document Architecture

MDCT – Multidetector Computed Tomography

HU – Hounsfield Units

UID – Unique Identifiers

SCU – Service Class User

SCP – Service Class Provider

DIMSE – DICOM Message Service Element

GSDF – Grayscale Standard Display Function

КТ – Комп’ютерна томографія

МРТ – Магнітно-резонансна томографія

УЗД – Ультразвукове дослідження

ВСТУП

Сучасна медицина немислима без використання цифрових технологій, які кардинально змінили підходи до діагностики, лікування та управління медичною інформацією. Особливе місце серед них займають системи роботи з медичними зображеннями, що генерують величезні обсяги даних, ефективне управління якими є критично важливим. Стандарт DICOM став ключовим для обробки таких зображень, забезпечуючи інтероперабельність.

Актуальність поставленої задачі в роботі полягає у тому, що зі зростанням обсягів медичних даних та поширенням цифрових діагностичних методів виникає нагальна потреба у створенні зручних та ефективних інструментів для роботи з цією інформацією. Веб-застосунки надають гнучкі можливості для доступу до медичної документації та зображень, що сприяє покращенню взаємодії між медичними працівниками, прискоренню діагностичного процесу та залученню пацієнтів. Розробка спеціалізованого веб-застосунку, орієнтованого на роботу з медичною документацією та DICOM-зображеннями, дозволить оптимізувати робочі процеси, підвищити якість діагностики та забезпечити надійне зберігання даних. Методи досягнення поставленої мети включають аналіз предметної області, проєктування архітектури системи, розробку програмних модулів з використанням сучасних веб-технологій, інтеграцію з PACS-системами та DICOM-бібліотеками, тестування та економічне обґрунтування.

Мета і завдання роботи

Метою роботи є підвищення ефективності роботи медичних працівників з оптимізації процесів керування медичною документацією та медичними зображеннями у стандарті DICOM за допомогою веб-застосунку.

Її досягнення передбачає вирішення наступних *завдань*:

1. Проаналізувати теоретичні основи представлення та обробки медичних зображень у стандарті DICOM, включаючи його структуру, інформаційну модель, мережеві протоколи та взаємодію з системами архівації та передачі зображень.
2. Провести аналіз сучасних веб-технологій, інструментів для розробки клієнтської та серверної частин, систем управління базами даних, рішень для автентифікації, бібліотек для роботи з DICOM-зображеннями та PACS-серверів для обґрунтування вибору технологічного стеку
3. Спроектувати архітектуру веб-застосунку, що включає структуру бази даних для зберігання медичної інформації та логіку взаємодії між основними компонентами
4. Реалізувати серверну частину веб-застосунку для обробки бізнес-логіки, взаємодії з базою даних та комунікації з PACS-сервером.
5. Розробити клієнтську частину веб-застосунку на базі React та Next.js, зі стилізацією за допомогою Tailwind CSS, що забезпечує адаптивні інтерфейси для різних ролей користувачів.
6. Інтегрувати функціонал перегляду DICOM-зображень у веб-браузері, використовуючи бібліотеку Cornerstone.js, та реалізувати базові інструменти маніпуляції зображеннями.
7. Забезпечити взаємодію веб-застосунку з PACS-сервером Orthanc для управління DICOM-дослідженнями.
8. Реалізувати систему автентифікації користувачів та розмежування прав доступу відповідно до їх ролей.
9. Розглянути ключові аспекти захисту інформації в системі.
10. Провести тестування основних функціональних модулів та продемонструвати сценарії роботи користувачів.

Використані методи. Для вирішення поставлених завдань у роботі були використані: методи аналізу науково-технічної літератури та існуючих

рішень для визначення актуальності проблеми та обґрунтування вибору технологій; методи системного аналізу для розробки архітектури веб-застосунку; сучасні технології веб-розробки (TypeScript, React, Next.js, Node.js, MongoDB, Tailwind CSS, Cornerstone.js, Orthanc) для програмної реалізації; методи тестування програмного забезпечення для перевірки коректності роботи функціональних модулів; метод функціонально-вартісного аналізу для оцінки економічної ефективності проєкту.

Отримані результати. В результаті виконання дипломної роботи було розроблено веб-застосунок, що забезпечує функціонал для керування медичною документацією та зображеннями у стандарті DICOM. Створена система дозволяє реєструвати користувачів з різними ролями, керувати даними пацієнтів та їх візитів, взаємодіяти з PACS-сервером для завантаження та отримання DICOM-досліджень, а також переглядати та виконувати базові маніпуляції з DICOM-зображеннями безпосередньо у веб-браузері.

Структура роботи

Дипломна робота за темою «Веб-застосунок для керування медичною документацією та зображеннями в DICOM-стандарті» виконана студентом *Вічепольським Юрієм Юрійовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині»,* побудована за класичним типом та викладена на 88 сторінки машинописного тексту. Вона складається з: вступу; п'яти розділів («Огляд теоретичних основ обробки медичних зображень у форматі DICOM», «Аналіз сучасних програмних середовищ для керування медичною документацією та зображеннями в DICOM стандарті», «Засоби реалізації програмного забезпечення веб-застосунку для керування медичною документацією та зображеннями в DICOM стандарті», «Інформаційне забезпечення. Програмна реалізація та методика роботи веб-застосунку»,

«Захист інформації»), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 55 джерел та додатків (5 – на кирилиці, 50 – на латиниці). В роботі представлено 43 рисунків і 6 таблиць.

РОЗДІЛ 1
ОГЛЯД ТЕОРЕТИЧНИХ ОСНОВ ОБРОБКИ МЕДИЧНИХ ЗОБРАЖЕНЬ У ФОРМАТІ DICOM

В розділі розглянуто огляд теоретичних основ обробки медичних зображень у форматі DICOM: загальна характеристика та структура представлення даних в стандарті DICOM; файловий формат DICOM; візуалізація та інтеграція DICOM-даних; системи архівації та передачі зображень; забезпечення інтероперабельності.

1.1 Загальна характеристика стандарту DICOM

Зростання обсягів та складності даних, що генеруються цифровими методами візуалізації, такими як КТ, створило нагальну потребу в єдиному підході до їх обміну та управління. Вирішенням цієї проблеми став міжнародний стандарт DICOM [55, 34, 6]. Цей стандарт, розроблений спільними зусиллями американського коледжу радіології та національної асоціації виробників електрики і вперше опублікований у версії 3.0 у 1993 році, встановлює комплексні правила для всіх аспектів роботи з медичними зображеннями та пов’язаною з ними інформацією – від їх створення та форматування до передачі по мережі, зберігання, пошуку, обробки та візуалізації [55, 34, 6, 3]. Стандарт постійно розвивається та підтримується міжнародним комітетом, а також визнаний міжнародною організацією зі стандартизації як ISO 12052 [55, 34].

Ключовою метою DICOM є забезпечення інтероперабельності, тобто здатності медичного обладнання, програмного забезпечення та інформаційних систем від різних виробників безперешкодно взаємодіяти між собою [55, 34, 6, 3]. Надаючи єдину технічну «мову», DICOM дозволяє КТ-сканеру одного виробника надсилати зображення на робочу станцію іншого,

а системі архівації – коректно зберігати та надавати доступ до даних, отриманих з різноманітних джерел. Це стало фундаментом для переходу до цифрових робочих процесів у багатьох галузях медицини [34, 8, 3].

Хоча стандарт виник переважно для потреб радіології, його сфера застосування сьогодні надзвичайно широка і охоплює практично всі методи візуалізації: КТ, МРТ, УЗД, рентгенографію, мамографію, ПЕТ, ангіографію, ядерну медицину, а також кардіологію, променеву терапію, офтальмологію, стоматологію, ендоскопію, мікроскопію, цифрову патологію та інші [55, 34, 6, 8]. Така універсальність зробила DICOM одним із найважливіших стандартів у цифровій охороні здоров’я у світі [34].

Важливою перевагою DICOM є його комплексність. На відміну від інших стандартів, що можуть регулювати лише окремі аспекти обміну даними, DICOM надає повну специфікацію, яка охоплює структуру інформаційних об’єктів, правила кодування даних, детальні протоколи мережевої взаємодії та формат файлів для офлайн-обміну. Ця повнота значно спрощує інтеграцію різнорідних систем і забезпечує вищий рівень автоматичної сумісності між ними [6].

1.2 Структура та представлення даних у DICOM

Для забезпечення порядку та однозначності при роботі з медичною інформацією, DICOM використовує чітко структуровану модель даних та систему їх опису за допомогою атрибутів.

1.2.1 Інформаційна модель: від пацієнта до зображення

Дані в DICOM організовані за ієрархічною моделлю, що відображає природний клінічний контекст [55, 6]. Ця модель зазвичай включає чотири основні рівні: пацієнт, дослідження, серія та екземпляр. Найвищий рівень ідентифікує пацієнта. Наступний рівень, дослідження, об’єднує всі дані, що стосуються однієї медичної процедури. Одне дослідження може містити одну або кілька серій, де кожна серія представляє логічно пов’язану групу даних.

Найнижчий рівень, екземпляр, представляє собою окремий інформаційний об’єкт, найчастіше – одне зображення, але також може бути звітом, кривою тощо [55, 6]. Ключовим аспектом цієї моделі є інкапсуляція: кожен екземпляр містить не лише власні дані, але й інформацію, що ідентифікує відповідні серію, дослідження та пацієнта, забезпечуючи нерозривний зв’язок даних з їхнім контекстом [55, 34, 21].

1.2.2 Елементи даних DICOM – атрибути

Вся інформація всередині DICOM-об’єктів представлена у вигляді стандартизованих елементів даних, які називаються атрибутами або тегами [34, 21]. Кожен атрибут відповідає за певний фрагмент інформації – ім’я пацієнта, дату народження, тип обладнання, параметри зображення тощо. Структура атрибуту чітко визначена стандартом і включає унікальний числовий ідентифікатор – тег, що складається з номера групи та номера елемента [21]. Крім тегу, атрибут має представлення значення – код із двох літер, який визначає тип даних та правила їх інтерпретації [21, 6]. Атрибут також містить інформацію про довжину свого значення та, власне, саме значення [21]. Стандарт DICOM містить великий словник даних, який описує всі стандартизовані теги, їх представлення значення та множинність значення, тобто скільки значень може містити атрибут [21, 6].

1.2.3 Стиснення даних зображень у DICOM

Для зменшення обсягів даних, особливо при передачі по мережі або довготривалому зберіганні, DICOM інтегрує підтримку різних алгоритмів стиснення для піксельних даних. Інформація про використаний метод стиснення включається до синтаксису передачі [34, 21]. Стандарт підтримує як алгоритми стиснення без втрат, наприклад, RLE, різні варіанти JPEG Lossless, JPEG-LS, JPEG 2000 Lossless, які гарантують повне відновлення вихідних даних, так і алгоритми стиснення з втратами, наприклад, JPEG Baseline, JPEG 2000, які дозволяють досягти вищого ступеня стиснення, але призводять до незворотних змін у зображенні [34, 21]. Вибір методу залежить від вимог до якості зображення для конкретної клінічної задачі [34].

1.3 Структура DICOM-файлу

Файл у форматі DICOM містить повний набір даних одного екземпляра SOP [55, 34, 21]. Важливою особливістю цього формату є наявність на початку файлової мета-інформації – спеціального заголовка з атрибутами групи 0002 [21, 6]. Ця мета-інформація містить ключові дані для інтерпретації решти файлу, зокрема, UID SOP класу та екземпляра, що зберігаються, а також UID синтаксису передачі, в якому закодовано основний набір даних [21]. Це робить файл DICOM самодостатнім, дозволяючи програмам правильно його прочитати навіть без попередньої інформації про формат кодування [21]. Для файлів DICOM також визначені стандартні MIME-тип (application/dicom) та UTI (org.nema.dicom) [55, 34].

1.4 Візуалізація та інтеграція DICOM-даних

Стандартизація формату даних є передумовою для їх коректної візуалізації та інтеграції в медичні системи.

1.4.1 Стандартне відображення відтінків сірого

Щоб забезпечити однакове візуальне сприйняття яскравості зображень на різних моніторах, DICOM визначає функцію стандартного відображення відтінків сірого. Це стандартна крива, що пов’язує значення пікселів з рівнем яскравості дисплея, враховуючи особливості людського зору. Використання моніторів, відкаліброваних за GSDF, сприяє послідовності та надійності діагностичної інтерпретації [34, 6].

1.4.2 Засоби перегляду DICOM

Для роботи з DICOM-файлами необхідні спеціальні програми – DICOM переглядачі [55]. Вони вміють читати складну структуру файлу, коректно відображати зображення, надавати доступ до метаданих та пропонувати інструменти для аналізу [55]. Існує велике різноманіття таких програм, від простих переглядачів до повнофункціональних діагностичних станцій [55].

Більш детальний аналіз існуючих програмних рішень для перегляду DICOM, їхніх функціональних можливостей та критеріїв вибору буде проведено у другому розділі цієї роботи, присвяченому аналізу технологій.

1.5 Системи архівації та передачі зображень

Поряд зі стандартом DICOM, який визначає формат даних та протоколи обміну, ключовим елементом сучасної цифрової інфраструктури медичної візуалізації є системи архівації та передачі зображень, відомі за англійською абревіатурою PACS [8, 3]. Якщо DICOM – це «мова» та «правила» спілкування, то PACS – це комплексна система, що використовує цю мову для ефективного управління потоками медичних зображень у межах медичного закладу [8].

Основне призначення PACS – це цифрове отримання, довготривале зберігання, швидкий пошук та передача медичних зображень та пов’язаної з ними інформації [8, 3]. Поява PACS була мотивована необхідністю заміни громіздких та неефективних плівкових архівів, особливо в умовах стрімкого зростання кількості та обсягу цифрових досліджень [3].

Важливо підкреслити, що ефективне функціонування сучасної системи PACS нерозривно пов’язане зі стандартом DICOM [8, 3]. Саме DICOM надає ту універсальну основу, яка дозволяє компонентам PACS від різних виробників взаємодіяти між собою [3]. Модальності надсилають зображення на сервер архіву за допомогою сервісу DICOM C-STORE. Робочі станції використовують сервіси DICOM C-FIND, C-MOVE або C-GET для пошуку та отримання необхідних досліджень з архіву. Навіть передача зображень на медичні принтери відбувається за допомогою DICOM Print Management [55, 34, 21, 6, 8]. Без стандартизації форматів даних та протоколів комунікації, яку забезпечує DICOM, побудова інтегрованої та інтероперабельної системи PACS була б надзвичайно складною, якщо взагалі можливою. Технічно PACS може працювати з нестандартними форматами, але це призводить до

значних обмежень функціональності, проблем із сумісністю, ускладненого обміну даними та залежності від конкретного виробника [8].

1.6 Забезпечення інтероперабельності

Для того, щоб користувачі могли перевірити сумісність обладнання та програмного забезпечення від різних виробників, стандарт DICOM вимагає від виробників надавати заяву про відповідність DICOM, так званий DICOM Conformance Statement [6].

Це детальний технічний документ, який описує, які саме функції та опції стандарту DICOM підтримуються конкретним продуктом. Він містить інформацію про підтримувані SOP класи, ролі для них, синтаксиси передачі, мережеві налаштування та будь-які приватні розширення. Аналізуючи ці заяви для двох систем, можна заздалегідь оцінити їхню потенційну здатність до взаємодії для виконання конкретних завдань. Наявність та вивчення Conformance Statements є важливим кроком при плануванні та впровадженні DICOM-сумісних рішень [6].

Висновки до розділу 1

У даному розділі було розглянуто теоретичні основи, необхідні для розуміння роботи з цифровими медичними зображеннями у стандарті DICOM. Основну увагу було приділено стандарту DICOM як універсальній «мові» сучасної медичної візуалізації. Було проаналізовано його головну мету – забезпечення інтероперабельності, детальну структуру даних на основі атрибутів, правила їх кодування та можливості стиснення. Розглянуто стандартизований файловий формат для зберігання даних. Було представлено концепцію системи архівації та передачі зображень та її нерозривний зв’язок зі стандартом DICOM. Висвітлено роль DICOM у функціонуванні PACS та їх інтеграції з іншими медичними інформаційними системами.

РОЗДІЛ 2
АНАЛІЗ СУЧАСНИХ ПРОГРАМНИХ СЕРЕДОВИЩЬ ДЛЯ КЕРУВАННЯ МЕДИЧНОЮ ДОКУМЕНТАЦІЄЮ ТА ЗОБРАЖЕННЯМИ В DICOM-СТАНДАРТІ

В даному розділі розглянуто аналіз сучасних веб-застосунків у медичній галузі для керування медичною документацією та зображеннями в DICOM стандарті, а саме: аналіз сучасних веб-застосунків у медичній галузі; аналоги існуючих методів та аналогів.

2.1 Аналіз сучасних веб-застосунків у медичній галузі

Сучасна охорона здоров’я зазнає значної цифрової трансформації, яка впливає на всі аспекти галузі, а саме безпеку, прогнозованість, продуктивність та доступність медичної допомоги [4]. Веб-застосунки стали невід’ємною частиною цієї трансформації, слугуючи важливим інструментом для взаємодії між пацієнтами та медичними закладами, надання інформації та доступу до послуг [33].

2.1.1 Тенденції розвитку

Однією з ключових тенденцій є широке розповсюдження Інтернету та веб-платформ як каналів для надання медичних послуг та інформації. Більшість людей сьогодні мають доступ до Інтернету та активно використовують його для пошуку інформації про лікування, а також для отримання підтримки від інших людей зі схожими проблемами [30]. Веб-сайти медичних закладів стають першим кроком для просування послуг, надаючи користувачам зручну навігацію та вичерпну інформацію [33]. Інтернет також став ефективною альтернативою традиційному медичному обслуговуванню, пропонуючи веб-терапії та комп’ютеризовані програми самодопомоги, ефективність яких для певних станів (наприклад, розладів

настрою, хронічних захворювань) підтверджується науковими даними [30, 15].

Поряд із веб-сайтами стрімко розвивається напрямок мобільного здоров'я, що охоплює як веб-застосунки, так і нативні мобільні застосунки [16, 33]. Ці рішення застосовуються для моніторингу здоров'я, промоції здорового способу життя, комунікації, збору даних, телемедицини та підтримки прийняття рішень [16, 1]. Вони спрощують комунікацію між пацієнтом і лікарем, надають зручний доступ до медичних послуг таких як запис на прийом, перегляд медичної інформації, нагадують про візити чи прийом ліків тощо [33]. У деяких країнах, як-от у Німеччині, спостерігається тенденція до інтеграції сертифікованих застосунків мобільного здоров'я у національну систему охорони здоров'я з відшкодуванням їх вартості страховими компаніями [16].

Фундаментом для багатьох сучасних медичних веб-застосунків стають Інтернет медичних речей та хмарні обчислення. IoMT являє собою мережу взаємопов'язаних медичних пристроїв, сенсорів та систем, призначених для збору, обробки та передачі медичних даних [1, 2]. Ці пристрої безперервно генерують дані, які потім передаються на централізовані сервери, часто розташовані у хмарі, для подальшого аналізу [1, 2]. Хмарні обчислення надають необхідну інфраструктуру для зберігання, управління та обробки великих обсягів медичних даних, забезпечуючи гнучкий та масштабований віддалений доступ [4]. Це дозволяє долати географічні обмеження, централізувати електронні медичні записи, підтримувати телемедичні сервіси, сприяти медичним дослідженням та забезпечувати безперервність роботи завдяки рішенням для аварійного відновлення [4]. Спостерігається тенденція до використання різних моделей хмарних сервісів типу IaaS, PaaS, SaaS та спеціалізованих хмарних платформ для охорони здоров'я [4].

Іншою важливою тенденцією є інтеграція штучного інтелекту та машинного навчання у медичні веб-застосунки. Ці інструменти використовуються для аналізу великих обсягів даних з метою прогнозування

захворювань, підтримки клінічних рішень, персоналізації лікування та підвищення точності діагностики [1, 4].

Також спостерігається рух у напрямку персоно-орієнтованого підходу у розробці веб-підтримки. Це передбачає активне залучення пацієнтів та їхніх близьких до процесу розробки для врахування їхнього унікального досвіду, потреб та очікувань [30]. Метою є створення не просто технологічно досконалих, а дійсно корисних та зручних для конкретного користувача інструментів, які б допомагали пацієнтам краще керувати своїм станом та приймати обґрунтовані рішення щодо власного здоров'я [30, 1].

2.1.2 Проблематика та виклики

Попри значний потенціал, широке впровадження веб-застосунків у медичну практику пов'язане з численними проблемами, які потребують ретельного аналізу та вирішення [16, 2].

Безпека та конфіденційність даних

Одним із найгостріших питань є безпека та конфіденційність даних. Медичні веб-застосунки постійно перебувають під загрозою атак, кількість яких значно зросла в останні роки [2]. Вразливості в застосунках призводять до технічних зломів, витоків чутливої інформації пацієнтів, що може мати катастрофічні наслідки як для пацієнтів, так і для медичних закладів [16, 1, 2]. Забезпечення надійного захисту даних на всіх етапах – від збору за допомогою IoMT-пристроїв до передачі та зберігання у хмарі – є першочерговим завданням [1, 2]. Це вимагає впровадження надійних методів шифрування, контролю доступу, безпечних протоколів зв'язку, а також використання новітніх підходів, таких як блокчейн чи гомоморфне шифрування [1, 2]. Існують випадки в яких безпека відходить на другий план порівняно зі швидкістю розробки та зручністю використання, що створює додаткові ризики [2]. Важливою проблемою є також недостатня прозорість політик конфіденційності або їх незрозумілість для користувачів [16].

Зручність використання веб-застосунків

Тісно пов’язаною з безпекою є проблема зручності використання, користувацького досвіду та впровадження. Навіть безпечний та функціональний застосунок не буде ефективним, якщо він незручний або складний для користувачів – як пацієнтів, так і медичних працівників [16, 30, 1, 33]. Проблеми включають поганий дизайн, неінтуїтивну навігацію, відсутність належних інструкцій, неадаптованість до різних пристроїв [16, 33]. Важливим є забезпечення доступності для людей з різними можливостями та рівнем технічної підготовки [1]. Низька зручність використання безпосередньо впливає на прихильність пацієнтів до використання застосунків, що є ще однією значною проблемою. Спостерігаються високі показники відмов від використання через відсутність мотивації, сприйняття застосунку як додаткового тягаря, труднощі з інтеграцією в повсякденне життя або просто нецікавий контент [16, 15]. Недостатній рівень цифрової грамотності також є суттєвим бар’єром [16, 1]. Розв’язання цих проблем вимагає застосування принципів персоно-орієнтованого та користуваче-центрованого дизайну, активного залучення кінцевих користувачів до процесу розробки та тестування [30, 1, 33].

Надійність та точність

Серйозні виклики існують і в технічній сфері та інтероперабельності. Надійність та точність роботи пристроїв IoMT, сенсорів та самих програмних застосунків є критично важливими, оскільки помилки можуть мати летальні наслідки [1, 2]. Проблеми включають апаратні обмеження, програмні збої, нестабільність мережевого з’єднання, а також труднощі з оновленням ПЗ [16, 1]. Ключовою технічною проблемою є забезпечення інтероперабельності – здатності різних систем та пристроїв безперешкодно обмінюватися даними та взаємодіяти між собою [1]. Відсутність уніфікованих стандартів, проблеми сумісності між продуктами різних виробників та складнощі інтеграції даних з різних джерел значно ускладнюють створення єдиного цифрового медичного

простору [1, 2]. Хоча існують стандарти, такі як HL7 та FHIR, їх впровадження та дотримання залишаються викликом [1].

Таким чином, розвиток та впровадження веб-застосунків у медичній галузі відкривають значні перспективи, але водночас стикаються з комплексними викликами, що потребують системного підходу до їх вирішення на технологічному, організаційному, економічному та етичному рівнях.

2.2 Аналіз існуючих програмних рішень

Під час роботи було проаналізовано наступні існуючі методи та аналоги програмних застосунків для керування медичною документацією та зображеннями в DICOM стандарті, такими як: Helsi, Ambra Health, OHIF Viewer. Нижче розглянуто їх основні характеристики, переваги та недоліки в контексті завдань дипломної роботи.

2.2.1 Helsi

Helsi позиціонується як сучасна та надійна електронна медична система, призначена для пацієнтів, лікарів, а також державних і приватних медичних закладів в Україні [26]. Система є найбільшою медичною інформаційною системою в країні, якою користується значна частина населення, велика кількість закладів охорони здоров’я та медичних працівників [54]. За даними, щомісяця через Helsi здійснюється понад 2,5 мільйона записів на прийом до лікаря [54].

Для пацієнтів Helsi пропонує широкий спектр послуг (рис.2.1), включаючи можливість легко знаходити та обирати лікаря, здійснювати швидкий онлайн-запис на прийом для себе та членів родини, а також отримувати доступ до власної електронної медичної картки (ЕМК) [26, 54]. У ЕМК пацієнти можуть переглядати результати аналізів та діагностики, призначення лікаря, план лікування, інформацію про щеплення, направлення та інші медичні документи [26, 54]. Також доступна функція пошуку та

бронювання ліків за рецептом [54].

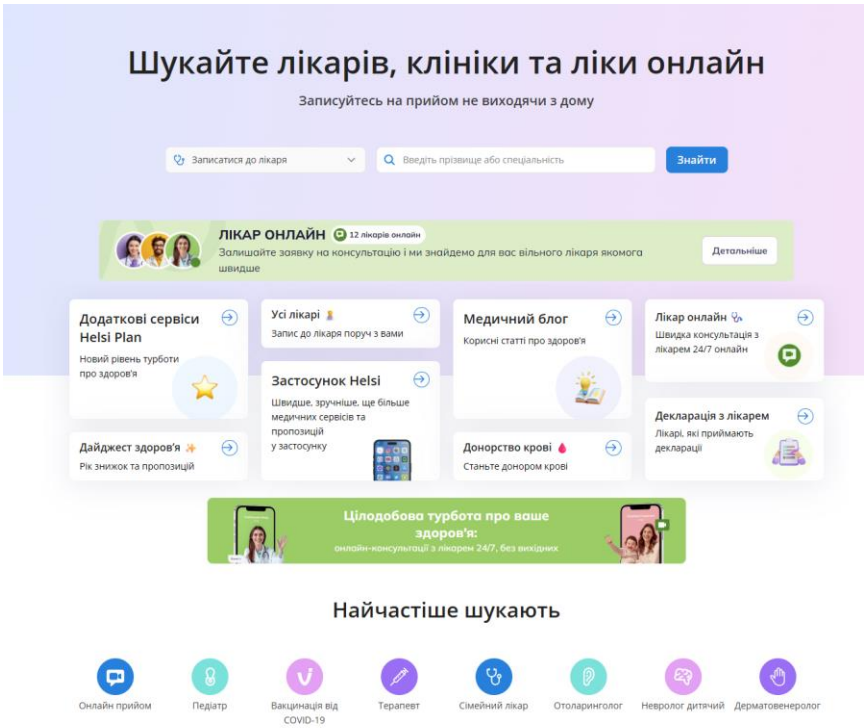

Рисунок 2.1 – Інтерфейс Helsi

Лікарі за допомогою Helsi можуть зручно вести історії хвороб пацієнтів та їх ЕМК, оперативно отримувати результати діагностичних досліджень та аналізів, використовувати клінічні протоколи та керувати прийомом пацієнтів через зрозумілий інтерфейс [26]. Медичні заклади отримують інструмент для комплексної автоматизації роботи, моніторингу, формування звітності та статистики, а також функціонал для участі в державній реформі охорони здоров'я, зокрема, для реєстрації декларацій з пацієнтами [26].

Серед ключових можливостей системи відзначається автоматизація роботи реєстратури та лікаря, управління розкладом, ведення ЕМК, облік медичних препаратів, підтримка стандарту HL7 та формату DICOM [26]. Платформа доступна через веб-інтерфейс та мобільні застосунки для Android та iOS [54]. Важливим аспектом є забезпечення безпеки та конфіденційності даних: стверджується, що всі дані зберігаються у дата-центрі, який отримав

сертифікат комплексної системи захисту інформації від Державної служби спеціального зв'язку та захисту інформації України, а обробка персональної інформації відбувається відповідно до українського законодавства [26, 54]. Інформацію про пацієнта бачить лише сам пацієнт та авторизований медичний персонал [54].

Переваги Helsi полягають у її широкому розповсюдженні в Україні, що забезпечує велику базу користувачів та медичних закладів [54]. Система надає комплексний набір функцій для всіх учасників медичного процесу, сприяючи цифровізації охорони здоров'я [26, 54]. Наявність мобільних застосунків та веб-порталу робить її доступною для широкого кола користувачів [54]. Декларована підтримка стандарту DICOM є важливою для роботи з медичними зображеннями [26]. Система також наголошує на високому рівні захисту даних [26, 54].

Щодо особливостей функціонування, які варто враховувати в контексті розробки спеціалізованого DICOM-рішення: Helsi функціонує переважно як МІС загального призначення та пацієнтський портал [26, 54]. Хоча підтримка DICOM декларується [26], основний акцент системи спрямований на адміністративні, інформаційні та комунікаційні аспекти медичного обслуговування [26, 54]. Надана інформація не деталізує наявність розширених інструментів для поглибленої візуалізації, аналізу та маніпуляції DICOM-зображеннями безпосередньо лікарями в рамках системи, що є ключовим для завдань даної дипломної роботи. Пацієнти, наприклад, можуть завантажувати свої аналізи як документи, але не можуть самостійно вносити медичні дані до ЕМК [54].

2.2.2 Ambra Health

Ambra Health представляє собою компанію, що спеціалізується на управлінні медичними даними та зображеннями у хмарному середовищі, пропонуючи програмне забезпечення як послугу [22, 43]. Платформа Ambra розроблена для забезпечення інновацій та прогресу в обробці зображень для постачальників медичних послуг, включаючи великі системи охорони

здоров’я, радіологічні практики та організації, що займаються клінічними дослідженнями (рис. 2.2) [22, 43]. Компанія звітувала про управління понад 10 мільярдами зображень з понад 100 мільйонів обстежень та наявність понад 600 000 користувачів у своїй мережі в більш ніж 50 країнах [43].

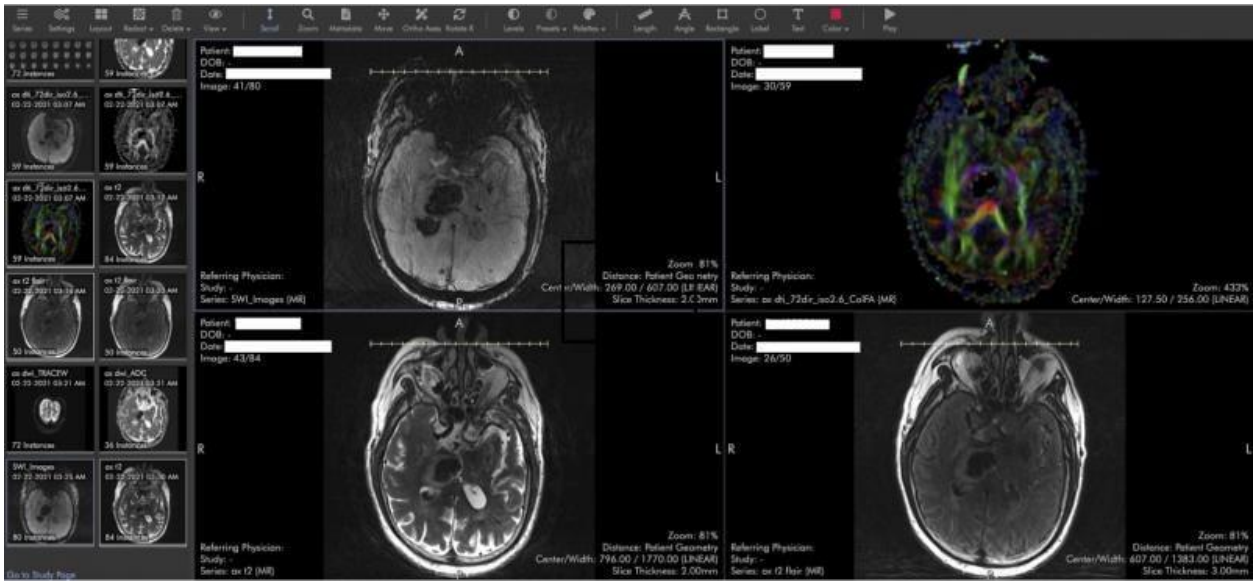

Рисунок 2.2 – Вбудований DICOM-переглядач Ambra Health

Ключовою пропозицією є Ambra Cloud PACS, яка є повністю керованою платформою, що використовує хмарну інфраструктуру, наприклад, Google Cloud [22]. Це рішення дозволяє медичним закладам переміщувати дані досліджень в єдине сховище для легкого доступу та перегляду [22]. Зображення можуть бути доступні в будь-який час та з будь-якого місця через веб-переглядач Ambra, що працює на основі HTML5, має нульовий слід встановлення та отримав дозвіл FDA 510(k) [22].

Платформа Ambra Health підтримує обмін зображеннями, їх архівування, а також надає пацієнтські портали для миттєвого доступу до зображень [43]. Серед можливостей відзначається інтеграція з електронними медичними картами, наприклад, EPIC, що дозволяє пацієнтам, лікарям та адміністраторам отримувати доступ до медичних зображень [43]. Ambra також пропонує інструменти для деідетифікації даних та платформу для

роботи з даними для науковців, що підтримує скрипти на Python, що є важливим для досліджень у сфері штучного інтелекту [22, 43]. Компанія активно розвиває свою платформу, впроваджуючи численні оновлення та функціональні можливості, зокрема, діагностичний переглядач ProViewer з розширеними інструментами візуалізації та аналізу [43].

Ambra Health співпрацює з Google Cloud, використовуючи такі сервіси, як Compute Engine, Cloud Storage та Cloud Data Loss Prevention API для управління та редагування чутливих даних пацієнтів у великих масштабах [22]. Прихильність Google Cloud до стандартів HIPAA та GDPR, а також надійні механізми безпеки та шифрування, спрощують для Ambra Health процес розгортання нових рішень [22].

Переваги Ambra Health включають потужну хмарну архітектуру, що забезпечує масштабованість, надійність та безпеку, підкріплену інфраструктурою Google Cloud [22]. Платформа пропонує комплексне рішення для управління медичними зображеннями, включаючи PACS, VNA, обмін зображеннями та пацієнтські портали [43]. Наявність веб-переглядача DICOM з нульовим слідом, сертифікованого FDA, є суттєвою перевагою для доступності та зручності використання [22]. Можливості для деідетифікації даних та підтримка досліджень у сфері ШІ розширюють сфери застосування платформи [22, 43]. Компанія демонструє значне зростання та має велику клієнтську базу, що свідчить про затребуваність її рішень [43].

Серед особливостей, що варто враховувати, є те, що Ambra Health є комерційним SaaS-рішенням, що передбачає регулярні витрати на використання, на відміну від відкритих програмних продуктів [22, 43]. Специфіка SaaS-моделі може накладати певні обмеження на глибоку кастомізацію або інтеграцію у власну унікальну інфраструктуру [22, 43]. Хоча платформа забезпечує високий рівень безпеки, використання стороннього хмарного провайдера є фактором, який організації враховують при виборі архітектури [22]. У контексті дипломної роботи, спрямованої на розробку власного веб-застосунку, використання готової комерційної

платформи такого масштабу може бути надлишковим та обмежувати гнучкість у реалізації специфічних вимог проєкту.

2.2.3 OHIF Viewer

Open Health Imaging Foundation (OHIF) Viewer – це веб-платформа медичних зображень з відкритим вихідним кодом, яка має на меті надання базової структури для створення складних застосунків для роботи із зображеннями [37, 31]. Вона розроблена для максимально швидкого завантаження великих радіологічних досліджень, випереджувального отримання метаданих та потокової передачі піксельних даних зображень за потребою [37]. OHIF Viewer використовує Cornerstone3D для декодування, рендерингу та анотування медичних зображень (рис. 2.3)[37].

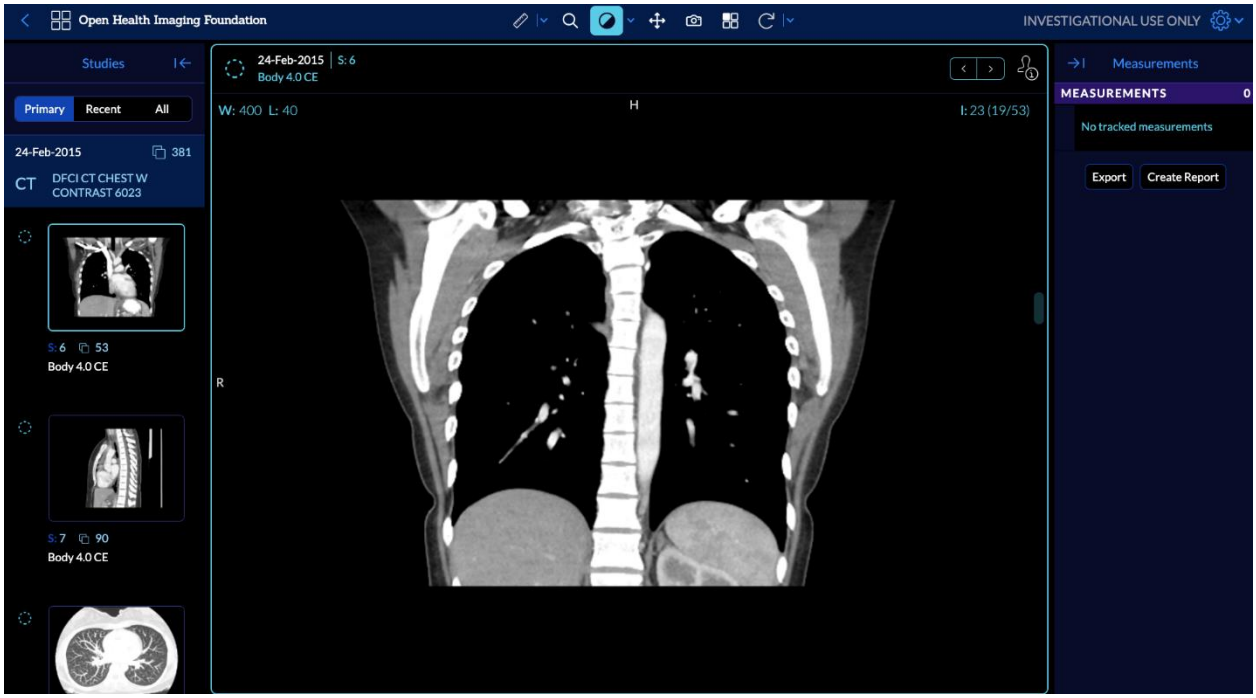

Рисунок 2.3 – DICOM-переглядач OHIF Viewer

Платформа може працювати «з коробки» з архівами зображень, що підтримують стандарт DICOMweb [37, 39], а також пропонує API джерела даних для взаємодії з архівами через пропрієтарні формати API [37]. Важливою особливістю є наявність фреймворку плагінів, що дозволяє створювати режими робочих процесів на основі завдань, які можуть

повторно використовувати основні функціональні можливості [37]. Інтерфейс користувача розроблений з урахуванням розширюваності, а його компоненти доступні у вигляді бібліотеки на React.js та Tailwind CSS [37, 39].

OHIF Viewer підтримує візуалізацію зображень у форматах 2D, 3D, а також реструктуризовані подання [39]. Серед функцій візуалізації відзначаються мультимодальне злиття зображень, проекція максимальної інтенсивності (MIP), мультипланарна реконструкція, відображення в аксіальній, сагітальній та корональній площинах, режим CINE та багато інших [39]. Для анотування доступні різноманітні інструменти вимірювання, включаючи специфічні для сенсорних екранів, як-от масштабування двома пальцями [39]. Архітектура OHIF Viewer є модульною і включає такі компоненти, як ядро з основними функціями, засоби інтернаціоналізації, сам переглядач, бібліотеку UI-компонентів, розширення для реалізації ключових функцій та режими для конфігурування розширень під конкретні завдання [39]. Режими, представлені у версії 3, дозволяють створювати різноманітні міні-застосунки або робочі процеси в межах одного застосунку, наприклад, для 3D-сегментації або відстеження вимірювань [39].

Проєкт активно розвивається, існують приклади його інтеграції з іншими платформами, наприклад, з XNAT – інформаційною платформою для підтримки досліджень в галузі візуалізації [18]. Така інтеграція дозволила додати до XNAT сучасні інструменти візуалізації та анотування, зокрема для створення та зберігання областей інтересу (ROI) у вигляді контурів та масок, інструменти для «розумного» малювання пензлем на КТ-зображеннях, інтеграцію з NVIDIA AIAA, можливість перегляду поверхонь, фракційних карт сегментації та накладання зображень [18].

Переваги OHIF Viewer полягають у його відкритості, нульовому сліді встановлення та веб-орієнтованості, що забезпечує доступність з будь-якого пристрою через браузер [31, 37, 18]. Платформа є розширюваною завдяки системі плагінів та режимів, що дозволяє адаптувати її до конкретних робочих процесів та дослідницьких завдань [37, 39]. Використання сучасних

веб-технологій, таких як React та Tailwind CSS, сприяє створенню якісного та адаптивного інтерфейсу [37]. Підтримка стандарту DICOMweb забезпечує сумісність із сучасними медичними архівами [37, 39]. Активна спільнота та підтримка з боку таких організацій, як Chan Zuckerberg Initiative, сприяють подальшому розвитку проєкту [39]. Важливою є можливість інтеграції з інструментами ШІ-асистованого анотування [18].

До обмежень або аспектів, що вимагають уваги, можна віднести те, що, будучи веб-рішенням, продуктивність при роботі з дуже великими наборами даних може поступатися спеціалізованим десктопним застосункам [18]; також існує залежність від швидкості інтернет-з'єднання та ресурсів браузера [18]. Попри активний розвиток, для використання у складних клінічних сценаріях або як повна заміна PACS, OHIF Viewer може вимагати додаткової адаптації та налаштувань [18]. Інтеграція та кастомізація під специфічні потреби можуть потребувати значних зусиль з розробки, незважаючи на наявність фреймворку плагінів [18, 39]. Хоча платформа активно розвивається, підтримка деяких розширених функцій, таких як повноцінна робота зі специфічними об'єктами DICOM RT чи розширене версіонування ROI, може відрізнятися від можливостей спеціалізованих комерційних систем або потребуватиме додаткової розробки в рамках конкретного впровадження [18].

Висновки до розділу 2

У другому розділі було здійснено аналіз сучасного стану веб-застосунків у медичній сфері, визначено ключові тенденції їх розвитку, такі як поширення мобільного здоров'я, використання хмарних технологій та штучного інтелекту, а також основні виклики, зокрема проблеми безпеки даних, зручності використання та інтероперабельності систем. Окрім цього, було детально розглянуто існуючі програмні рішення для керування медичною документацією та зображеннями у стандарті DICOM, а саме Helsi,

Ambra Health та OHIF Viewer, з метою оцінки їх функціональних можливостей, переваг та недоліків.

Аналіз аналогів показав, що українська система Helsi є комплексною МІС з широким функціоналом для пацієнтів та лікарів, але з обмеженою специфікою саме для поглибленої роботи з DICOM-зображеннями. Ambra Health пропонує потужну хмарну комерційну платформу для управління медичними зображеннями, проте її SaaS-модель та вартість можуть не відповідати задачам розробки власного гнучкого рішення. OHIF Viewer, будучи відкритим веб-фреймворком, надає гарні можливості для створення кастомізованих DICOM-переглядачів, але вимагає значних зусиль на розробку та інтеграцію. Таким чином, проведений аналіз обґрунтовує доцільність створення спеціалізованого веб-застосунку.

РОЗДІЛ 3
ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ МЕДИЧНОЮ ДОКУМЕНТАЦІЄЮ ТА ЗОБРАЖЕННЯМИ В DICOM-СТАНДАРТІ

В даному розділі проаналізовано та обґрунтовано вибір програмних технологій та інструментів в розробці веб-застосунку для керування медичною документацією та зображеннями в DICOM-стандарті такими як :

- Мова програмування: TypeScript
- Технології для розробки інтерфейсу користувача: React та Next.js
- CSS-фреймворк: Tailwind CSS
- Система керування базами даних: MongoDB
- Рішення для автентифікації: NextAuth.js
- Технології для реалізації DICOM-переглядача: Cornerstone.js
- PACS-сервер: Orthanc

3.1 Аналіз та обґрунтування вибору програмних технологій та інструментів

Вибір правильних програмних технологій є ключовим етапом у розробці будь-якого веб-застосунку, а особливо такого, що працює з чутливими медичними даними та має забезпечувати високу надійність, безпеку та зручність використання.

3.1.1 Мова програмування TypeScript

Для реалізації веб-застосунку було обрано мову програмування TypeScript [51]. TypeScript є надмножиною JavaScript, тобто будь-який правильний JavaScript-код також буде правильним TypeScript-кодом [47, 19]. Весь код TypeScript компілюється у стандартний JavaScript, що гарантує його

виконання у всіх сучасних браузерах та середовищах виконання [47, 19, 7]. Це забезпечує повну сумісність з екосистемою JavaScript, включаючи численні бібліотеки та фреймворки, а також дозволяє інтегрувати TypeScript у існуючі проєкти поступово [47, 19, 7].

Основною причиною вибору TypeScript стала його система статичної типізації. На відміну від динамічної типізації JavaScript, де помилки, пов’язані з типами даних, часто виявляються лише під час виконання програми, TypeScript дозволяє виявляти такі помилки ще на етапі компіляції [47, 19, 53]. Розробник може явно вказувати типи для змінних, параметрів функцій, значень, які вони повертають та полів об’єктів. Компілятор перевіряє коректність використання цих типів по всьому коду, сигналізуючи про потенційні проблеми, такі як присвоєння значення невідповідного типу, виклик неіснуючого методу або передача неправильних аргументів у функцію [53].

Такий підхід значно підвищує надійність коду та якість програмного забезпечення [47, 19, 53]. Раннє виявлення помилок скорочує час та ресурси, що витрачаються на пошук та виправлення багів, особливо у великих та складних системах [47, 19, 53]. Це призводить до більш стабільної роботи програми та зменшує ризик непередбачуваної поведінки під час експлуатації, що є критично важливим у сфері охорони здоров’я [19, 53].

Статична типізація також суттєво покращує читабельність та підтримку коду [47, 19, 53, 7]. Явні анотації типів роблять код більш зрозумілим та документованим, полегшуючи його розуміння іншими розробниками або при поверненні до нього через деякий час [47, 19, 7]. Це особливо важливо при командній розробці, оскільки типи слугують чіткими специфікаціями щодо очікуваних даних між різними частинами системи, зменшуючи ймовірність непорозумінь [47, 53, 7].

TypeScript сприяє підвищенню продуктивності розробників [47, 19]. Сучасні інтегровані середовища розробки використовують інформацію про типи для надання розширених можливостей, таких як точне автодоповнення

коду, швидка навігація по проєкту, безпечний рефакторинг та виявлення помилок у реальному часі [47, 19, 53]. Це прискорює процес написання коду та зменшує кількість рутинних помилок [47, 19]. Екосистема TypeScript включає також інструменти для статичного аналізу коду, які допомагають забезпечувати єдиний стиль коду та дотримуватись найкращих практик [47, 19].

Водночас слід зазначити, що система типів TypeScript, через необхідність збереження повної сумісності з JavaScript, не є абсолютно стійкою [53]. Це означає, що існують певні механізми (тип any, твердження типів, нестрогий доступ до індексів), які можуть дозволити обійти систему типів і призвести до помилок під час виконання [53]. Однак, існують ефективні практики для підвищення надійності типізації в TypeScript (рис 3.1). До них належать використання строгих налаштувань компілятора, мінімізація використання типу any, застосовуючи інструменти статичного аналізу типу typescript-eslint, використання захисників типів замість тверджень типів для перевірки даних під час виконання, застосування бібліотек для валідації зовнішніх даних та використання файлів визначень типів для JavaScript-бібліотек [53].

Отже, TypeScript пропонує значні переваги для розробки надійних, підтримуваних та масштабованих веб-застосунків, особливо у порівнянні з чистим JavaScript. Можливість виявлення помилок на ранніх стадіях, покращена структура коду, підвищена продуктивність розробників та полегшення командної роботи, за умови використання строгих налаштувань та дотримання найкращих практик для забезпечення безпеки типів, роблять TypeScript обґрунтованим вибором для створення складного медичного веб-застосунку.

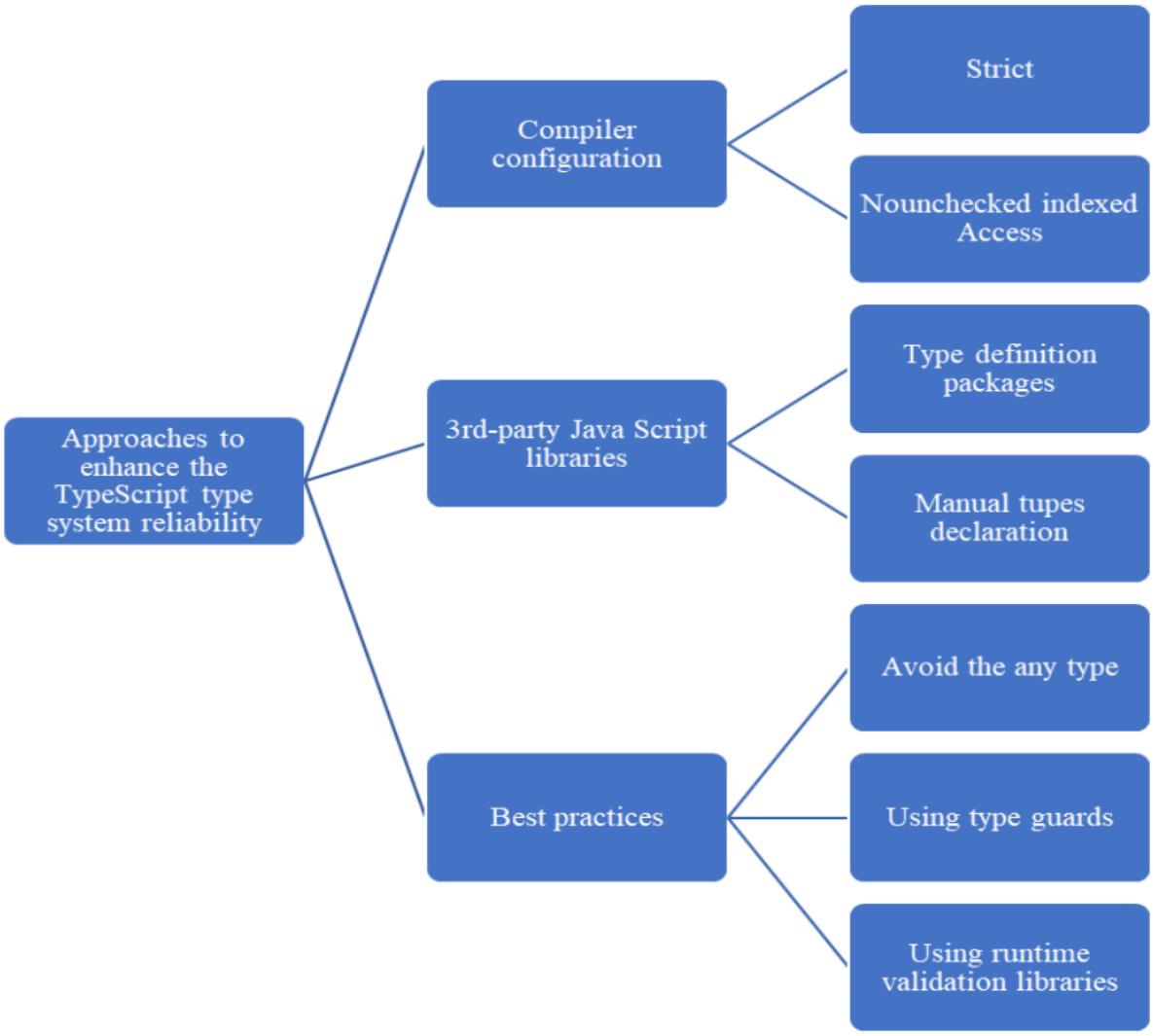

Рисунок 3.1 – Підходи до підвищення надійності системи типів TypeScript [53]

3.1.2 Вибір технологій для розробки інтерфейсу користувача. React та Next.js

Вибір правильних інструментів для розробки інтерфейсу користувача є критично важливим для успіху будь-якого веб-застосунку, особливо в медичній сфері, де вимоги до надійності, продуктивності та зручності використання є надзвичайно високими. Для даного проєкту було прийнято рішення використовувати бібліотеку React як основу для побудови UI-компонентів та фреймворк Next.js, що працює поверх React, для забезпечення додаткових функціональних можливостей, оптимізації продуктивності та спрощення процесу розробки.

Обґрунтування вибору програмного забезпечення для створення інтерфейсу React

React, розроблений компанією Facebook і випущений як відкрите програмне забезпечення у 2013 році, є однією з найпопулярніших JavaScript-бібліотек для створення користувацьких інтерфейсів [5, 40, 20].

Однією з фундаментальних переваг React є його компонентно-орієнтована архітектура. React дозволяє розробникам розбивати складний інтерфейс на невеликі, незалежні та перевикористовувані частини, які називаються компонентами [5, 40, 27]. Кожен компонент інкапсулює власну логіку та розмітку, що робить код більш модульним, організованим та легким для розуміння й підтримки [50, 5, 40, 27]. Можливість перевикористання компонентів значно прискорює процес розробки та забезпечує консистентність інтерфейсу по всьому застосунку [5, 40, 27]. Така модульність також полегшує тестування окремих частин UI та командну роботу над проєктом [5].

Іншою важливою технічною перевагою React є використання віртуального DOM [5, 40, 27, 20]. Замість того, щоб безпосередньо маніпулювати реальним DOM браузера при кожній зміні даних, що є повільною операцією, React спочатку оновлює легковагову копію DOM в пам’яті – віртуальний DOM [5, 27]. Потім React використовує ефективний алгоритм порівняння для визначення мінімальних відмінностей між попереднім та новим станом віртуального DOM [5]. Лише ці зміни потім застосовуються до реального DOM [5, 27]. Результатом є швидший та більш чутливий інтерфейс користувача [5].

React використовує декларативний підхід до програмування інтерфейсів [40]. Розробник описує, як має виглядати інтерфейс для певного стану застосунку, а React сам піклується про ефективне оновлення DOM для досягнення цього стану [46]. Це робить код більш передбачуваним та легшим для налагодження порівняно з імперативним підходом, де потрібно вручну керувати кожною зміною DOM [40].

Обґрунтування вибору фреймворку з відкритим кодом для створення веб-застосунків Next.js

Хоча React надає потужну основу для побудови UI, для створення сучасного, продуктивного та оптимізованого веб-застосунку часто потрібні додаткові можливості, які виходять за рамки базової бібліотеки. Саме тут у гру вступає Next.js – фреймворк з відкритим кодом, створений компанією Vercel, який розроблений спеціально для роботи поверх React [35, 13, 20]. Next.js надає структурований підхід та вирішує багато типових проблем React-розробки «з коробки», значно спрощуючи процес створення складних веб-застосунків [35, 50, 13, 20].

Next.js пропонує зручну систему маршрутизації на основі файлової системи [50, 20]. Маршрути застосунку автоматично створюються на основі структури файлів та папок у директорії app (або pages у старіших версіях) [50]. Це робить структуру проєкту інтуїтивно зрозумілою, спрощує навігацію та управління маршрутами [50]. Фреймворк також підтримує динамічну маршрутизацію, що дозволяє створювати сторінки на основі даних [50].

Інтегровані API шляхи є ще однією важливою можливістю Next.js [35, 50]. Вони дозволяють створювати бекенд-ендпоінти безпосередньо в проєкті Next.js, розміщуючи файли у спеціальній папці app/api [35]. Це спрощує архітектуру для застосунків, яким потрібна певна логіка на стороні сервера (наприклад, для взаємодії з базою даних, автентифікації, інтеграції зі сторонніми сервісами, такими як PACS-сервер), дозволяючи розробляти full-stack застосунки в рамках єдиного фреймворку [50].

Next.js включає в себе вбудовані оптимізації, спрямовані на покращення продуктивності та досвіду розробника. Автоматичне розділення коду відбувається на рівні сторінок, гарантуючи, що для кожної сторінки завантажується лише необхідний JavaScript [35, 41, 50, 13]. Це зменшує розмір початкового бандлу та прискорює завантаження [41, 50, 13]. Next.js також підтримує розділення коду на рівні компонентів за допомогою динамічних імпортів [35, 13]. Оптимізація зображень автоматично змінює

розмір, оптимізує та подає зображення у сучасних форматах, покращуючи швидкість завантаження сторінок [35, 41, 50]. Швидке оновлення забезпечує миттєве відображення змін у коді під час розробки без втрати стану компонентів, що значно підвищує продуктивність розробників [35, 13].

Порівняння з альтернативними frontend-фреймворками

Хоча комбінація React + Next.js була обрана для даного проєкту, важливо розглянути її в контексті інших популярних frontend-фреймворків, таких як Angular, Vue.js та Svelte [5, 27, 20]. Кожен з них має свої сильні та слабкі сторони, і вибір залежить від специфіки проєкту, вимог та досвіду команди розробників [5].

Angular, розроблений Google, є повноцінним, комплексним фреймворком, який пропонує багато вбудованих рішень «з коробки» для маршрутизації, управління станом, HTTP-запитів та іншого [5, 20]. Він використовує TypeScript за замовчуванням та реалізує двонаправлене зв’язування даних. Angular часто вважається більш «жорстким» та «громіздким» порівняно з React, з вищою кривою навчання, але може бути гарним вибором для великих корпоративних застосунків, де потрібна чітка структура та єдиний підхід [5, 20].

Vue.js часто позиціонується як прогресивний фреймворк, що поєднує гнучкість React з деякими зручностями Angular [5, 20]. Він відомий своєю відносно низькою кривою навчання, хорошою документацією та високою продуктивністю [20]. Vue також використовує віртуальний DOM, подібно до React. Його екосистема менша за React, але активно розвивається [5, 20]. Vue може бути чудовим вибором для проєктів середнього розміру або для команд, які шукають простіший старт [20].

Svelte відрізняється від інших фреймворків тим, що він є компілятором [20]. Замість використання віртуального DOM під час виконання, Svelte компілює компоненти у високоефективний імперативний JavaScript-код, який безпосередньо оновлює DOM [20]. Це призводить до дуже малих розмірів бандлів та високої продуктивності під час виконання. Однак його екосистема

та спільнота ще менші порівняно з React, Angular чи Vue, і він може мати крутішу криву навчання через свій унікальний підхід [20].

Для наочності ключові відмінності між фреймворками представлені у табл 3.1.

Таблиця 3.1

Порівняльний аналіз frontend-фреймворків [35, 41, 50, 13, 5, 40, 27, 20]

Критерій	React + Next.js	Angular	Vue.js	Svelte
Тип	Бібліотека + Фреймворк	Комплексний фреймворк	Прогресивний фреймворк	Компілятор
Основна парадигма	Компонентно-орієнтована, Декларативна	Компонентно-орієнтована, MVC/MVVM	Компонентно-орієнтована	Компонентно-орієнтована, Реактивна
Рендеринг	VDOM + SSR/SSG/ISR	Real DOM	VDOM, SSR	Компіляція в імперативний код, без VDOM
Управління станом	Гнучке	Вбудовані рішення	Вбудовані рішення	Вбудовані реактивні змінні
Маршрутизація	Вбудована файлова	Вбудований потужний модуль маршрутизації	Vue Router	Зазвичай зовнішні бібліотеки
Крива навчання	Середня	Висока	Низька	Середня
Екосистема/Спільнота	Дуже велика	Велика	Велика, зростаюча	Менша, зростаюча
Продуктивність	Висока	Висока	Висока	Дуже висока
Гнучкість	Дуже висока	Низька	Висока	Середня
TypeScript	Повна підтримка	Вбудований за замовчуванням	Хороша підтримка	Хороша підтримка

Враховуючи специфіку проєкту, комбінація React та Next.js була визнана оптимальною з кількох причин. По-перше, компонентний підхід React ідеально підходить для створення складних, але модульних медичних інтерфейсів [5, 40, 27]. По-друге, використання віртуального DOM у React та оптимізацій рендерингу у Next.js забезпечує високу продуктивність, що важливо при роботі з великими обсягами даних та інтерактивними інструментами перегляду зображень [35, 41, 50, 5, 40, 27]. По-третє,

можливість використання API шляхів у Next.js дозволяє реалізувати необхідну бекенд-логіку для взаємодії з базою даних та PACS-сервером в рамках єдиного проєкту, спрощуючи архітектуру [35, 50]. По-четверте, велика екосистема та спільнота React/Next.js надають доступ до численних бібліотек та підтримки [5, 40, 27, 20]. До того ж, використання TypeScript у поєднанні з цими технологіями підвищує надійність та керованість кодової бази, що є критичним для медичного застосунку [35, 20]. Хоча інші фреймворки мають свої переваги, обрана комбінація найкраще відповідає балансу між гнучкістю, продуктивністю, екосистемою та можливостями для розробки даного специфічного веб-застосунку.

3.1.3 CSS-фреймворк Tailwind CSS

Вибір підходу до стилізації користувацького інтерфейсу є ще одним важливим аспектом розробки веб-застосунку. Для даного проєкту було обрано CSS-фреймворк Tailwind CSS.

Tailwind позиціонується як utility-first CSS фреймворк [9, 49, 17]. Його основна ідея полягає у наданні великого набору низькорівневих, одноцільових CSS-класів, які застосовуються безпосередньо в HTML-розмітці для стилізації елементів [9, 49, 17, 23]. Замість того, щоб створювати власні семантичні класи і потім описувати їхні стилі в окремому CSS-файлі, розробник комбінує утилітарні класи Tailwind безпосередньо в атрибуті class HTML-елемента [9, 49].

Такий підхід має кілька суттєвих переваг, які обґрунтовують вибір Tailwind CSS для даного проєкту. По-перше, він значно прискорює процес розробки [9, 49, 17, 23]. Розробникам не потрібно постійно вигадувати імена для класів, переключатися між HTML та CSS файлами та писати велику кількість користувацького CSS [9, 49, 17]. Це дозволяє швидко створювати та ітерувати дизайни інтерфейсу [49, 17].

По-друге, Tailwind забезпечує високий рівень гнучкості та кастомізації. На відміну від компонентно-орієнтованих фреймворків, які надають готові, стилізовані компоненти, Tailwind надає утиліти, з яких можна створити

абсолютно будь-який дизайн, не будучи обмеженим стилями фреймворку за замовчуванням [9, 49, 17, 23]. Tailwind можна повністю налаштувати відповідно до вимог проєкту через конфігураційний файл tailwind.config.js [49, 17]. Це дозволяє створити унікальний візуальний стиль для медичного застосунку, не вдаючись до перезапису стилів фреймворку [49].

По-третє, Tailwind сприяє оптимізації кінцевого CSS-коду [49, 17]. Під час процесу збірки проєкту Tailwind автоматично сканує HTML-файли, JavaScript-компоненти та інші шаблони на предмет використаних класів-утиліт і генерує CSS-файл, що містить лише ті стилі, які реально використовуються [49, 17]. Весь невикористаний CSS видаляється [17]. Це призводить до дуже малих розмірів фінального CSS-бандлу, що позитивно впливає на швидкість завантаження сторінок та загальну продуктивність веб-застосунку [49, 17]. Це особливо важливо для медичних застосунків, які можуть бути складними та потребують швидкого відгуку інтерфейсу.

По-четверте, фреймворк має вбудовану підтримку адаптивного дизайну [9, 49, 17]. Дозволяється застосовувати утиліти умовно для різних розмірів екрану за допомогою префіксів. Це робить процес створення адаптивних інтерфейсів інтуїтивним та дозволяє керувати виглядом на різних пристроях безпосередньо в HTML [9, 49, 17]. Також легко реалізувати підтримку станів елементів та темної теми [9, 49].

Однак, підхід Tailwind має і певні недоліки та виклики. Найчастіше згадуваним є потенційне «засмічення» HTML-розмітки великою кількістю класів, що може знизити її читабельність, особливо для складних компонентів [9, 49, 17]. Хоча цю проблему можна частково вирішити шляхом винесення повторюваних наборів утиліт в компоненти або за допомогою директиви @apply в CSS, вона залишається предметом дискусій [49, 17].

Порівняння фреймворка Tailwind з аналогами

Вибір Tailwind стає зрозумілішим при порівнянні з іншими популярними підходами до стилізації. Розглянемо його відмінності від традиційних CSS-фреймворків, таких як Bootstrap. Останній є типовим

представником компонентно-орієнтованих фреймворків, надаючи великий набір готових, стилізованих UI-компонентів [9, 17, 23]. Це дозволяє дуже швидко створювати стандартні інтерфейси, що зручно для прототипів або для розробників-початківців [9, 17, 23]. Проте, такий підхід нав’язує певний візуальний стиль, і кастомізація часто вимагає перезапису CSS, що може призвести до роздутого коду та невикористаних стилів у фінальній збірці [9, 17, 23]. На противагу цьому, Tailwind не надає готових компонентів за замовчуванням, а дає повну свободу дизайну через утиліти, гарантуючи мінімальний розмір кінцевого CSS [9, 49, 17, 23].

Іншою альтернативою є написання «чистого» CSS або використання препроцесорів типу Sass/Less [9]. Цей метод забезпечує максимальний контроль над стилями та потенційно найбільш оптимізований CSS, специфічний для проєкту [9]. Однак, він є найбільш часозатратним, вимагає постійної підтримки консистентності та може призвести до великого обсягу коду, який потрібно підтримувати [9]. Tailwind прагне прискорити цей процес, надаючи готові утиліти та інструменти оптимізації [9, 49, 17].

Існує також підхід CSS-in-JS, де CSS-правила пишуться всередині JavaScript-компонентів. Це покращує інкапсуляцію стилів і полегшує роботу з динамічними стилями. Однак, на відміну від CSS-in-JS, який може виконувати певну роботу зі стилями під час виконання програми, Tailwind генерує статичний CSS-файл ще на етапі збірки проєкту, що зазвичай призводить до кращої продуктивності під час завантаження та роботи сторінки, оскільки браузеру не потрібно виконувати додаткову роботу з обробки стилів через JavaScript [49, 17].

У табл. 3.2 наведено порівняльний огляд Tailwind CSS та Bootstrap як представника компонентних фреймворків.

Таблиця 3.2

Порівняння Tailwind CSS та Bootstrap [9, 49, 17, 23]

Критерій	Tailwind CSS	Bootstrap
Підхід	Utility-First	Component-Based
Основні одиниці	Низькорівневі класи-утиліти	Готові, стилізовані UI-компоненти
Кастомізація	Дуже висока, гнучка	Обмежена, вимагає перезапису стилів
Швидкість розробки	Висока	Дуже висока
Крива навчання	Середня/Висока	Низька
Розмір CSS	Мінімальний	Потенційно великий
HTML-розмітка	Може бути «засміченою» класами	Зазвичай чистіша
Дизайн	Дає повну свободу, не нав’язує стиль	Нав’язує свій візуальний стиль, впізнаваний
Залежність	Залежить від процесу збірки	Може використовуватись через CDN без збірки

Для розробки веб-застосунку керування медичною документацією та DICOM-зображеннями потрібен був підхід, який би забезпечував швидкість розробки кастомного інтерфейсу, високу продуктивність та візуальну консистентність. Компонентні фреймворки, такі як Bootstrap, хоч і прискорюють початковий етап, могли б обмежити гнучкість у створенні специфічних UI-елементів, необхідних для взаємодії з медичними даними та DICOM-переглядачем. Підхід Utility-First, реалізований у Tailwind CSS, дозволяє швидко створювати унікальні дизайни, комбінуючи готові класи, без необхідності написання великої кількості власного CSS. Можливість повної кастомізації теми та гарантія мінімального розміру кінцевого CSS-файлу завдяки автоматичному видаленню невикористаних стилів є суттєвими перевагами для забезпечення високої продуктивності. Хоча існує проблема потенційного засмічення HTML, при використанні з React-компонентами її можна мінімізувати шляхом інкапсуляції стилів у компонентах. Таким чином, Tailwind CSS надає оптимальний баланс між швидкістю розробки, гнучкістю дизайну та оптимізацією продуктивності для завдань даного проєкту.

3.1.4 Система керування базами даних MongoDB

Вибір системи керування базами даних (СКБД) є фундаментальним рішенням при розробці будь-якого веб-застосунку, оскільки від нього залежить спосіб зберігання, організації та доступу до даних. Для даного проєкту було обрано NoSQL документо-орієнтовану базу даних MongoDB.

MongoDB є однією з провідних NoSQL СКБД, яка зберігає дані у вигляді гнучких, JSON-подібних документів (у бінарному форматі BSON), згрупованих у колекції [45, 28]. На відміну від традиційних реляційних баз даних (SQL), MongoDB не вимагає жорстко визначеної схеми таблиць перед збереженням даних [45, 24].

Обґрунтування вибору документної моделі MongoDB

Основною причиною вибору MongoDB стала гнучкість її документної моделі та схеми [32, 45, 28]. Медичні дані, особливо інформація про пацієнтів, візити та консультації, можуть мати різну структуру та еволюціонувати з часом. Документна модель дозволяє зберігати дані зі змінною структурою в одній колекції без необхідності складних та дорогих міграцій схеми, які характерні для реляційних баз даних [45]. Кожен документ може містити вкладені документи та масиви, що дозволяє природно представляти складні ієрархічні дані всередині одного запису [45]. Це спрощує розробку та зберігання даних, які не завжди легко вписуються у строгу табличну структуру SQL [45, 24]. Така гнучкість особливо цінна на етапі розробки та для застосунків, де вимоги до даних можуть змінюватися [45].

Аналіз продуктивності та масштабованості MongoDB

MongoDB розроблена з урахуванням вимог до високої продуктивності та масштабованості, що важливо для медичних застосунків, які можуть працювати з великою кількістю користувачів та даних [45, 24, 28]. Вона підтримує створення різноманітних індексів для прискорення запитів [45, 28]. Правильне використання індексів є критичним для оптимізації вибірки даних, дозволяючи базі даних швидко знаходити потрібні документи без

повного сканування колекції [28]. Агрегаційний фреймворк MongoDB дозволяє виконувати складні операції обробки та аналізу даних безпосередньо на сервері бази даних, що підвищує ефективність [45, 28].

Ключовою перевагою MongoDB є підтримка горизонтального масштабування за допомогою механізму шардингу [45, 28]. Шардинг дозволяє розподіляти великі колекції даних між кількома серверами, що дає змогу системі обробляти значно більші обсяги даних та вищі навантаження запитів, ніж може витримати один сервер [45]. Це забезпечує можливість нарощування потужності системи в міру зростання кількості користувачів та даних застосунку [45]. Хоча реляційні бази даних зазвичай масштабуються вертикально, горизонтальне масштабування часто є більш економічно ефективним та гнучким рішенням для великих навантажень [45]. MongoDB також підтримує реплікацію даних для забезпечення високої доступності та відмовостійкості [45].

Порівняння реляційних баз даних SQL з іншими NoSQL базами даних

Реляційні бази даних (SQL) використовують структуровану модель даних з таблицями, рядками та стовпцями, із заздалегідь визначеною схемою [45]. Вони чудово підходять для даних із чіткими зв’язками та вимогами до транзакційної цілісності [45, 24]. SQL бази даних мають зрілу екосистему, потужні можливості для складних запитів із використанням JOIN-операцій та гарантують високу консистенцію даних [45]. Однак, їхня жорстка схема ускладнює роботу з неструктурованими або напівструктурованими даними та вимагає міграцій при зміні структури [45]. Горизонтальне масштабування у SQL базах даних зазвичай складніше реалізувати порівняно з MongoDB [45]. Для даного проєкту, де структура даних користувачів, візитів та налаштувань може бути різноманітною та змінюватися, а потреба у складних JOIN-операціях між різними основними сутностями не є першочерговою, гнучкість документної моделі MongoDB видається більш доцільною [45]. Хоча MongoDB підтримує транзакції на рівні кількох документів з версії 4.0,

традиційні SQL бази даних все ще вважаються більш надійними для дуже складних транзакційних систем [45]. Слід також зазначити, що хоча MongoDB має свою мову запитів MQL, вона не підтримує SQL нативно, хоча існують інструменти для виконання SQL-подібних запитів [45].

Щодо інших NoSQL баз даних, існують різні типи, кожен зі своїми сильними сторонами [45, 24]. Key-value сховища надзвичайно швидкі для простих операцій читання/запису за ключем, але менш придатні для складних запитів. Колонко-орієнтовані бази даних добре масштабуються для запису великих обсягів даних та аналітичних запитів по великих датасетах, але можуть бути складнішими у використанні та управлінні [45]. Порівняно з іншими NoSQL базами, MongoDB пропонує хороший баланс між гнучкістю моделі даних, потужними можливостями запитів, горизонтальною масштабованістю та відносною простотою використання [45, 24]. Вона є лідером серед документо-орієнтованих баз даних і має велику спільноту та розвинену екосистему [45]. Дослідження показують, що за певних сценаріїв CRUD-операцій MongoDB демонструє високу продуктивність, хоча для складних запитів іноді може поступатися спеціалізованим SQL-рішенням, якщо схема даних добре структурована [24].

У табл. 3.3 представлено узагальнене порівняння MongoDB з реляційними базами даних.

Таблиця 3.3

Порівняння MongoDB та реляційних баз даних [32, 45]

Критерій	MongoDB	Реляційні бази даних
Модель даних	Документо-орієнтована	Таблична
Схема	Гнучка, динамічна	Жорстка, визначена заздалегідь
Структура даних	Добре підходить для ієрархічних, неструктурованих даних	Добре підходить для структурованих даних з чіткими зв’язками
Масштабованість	Горизонтальна	Переважно вертикальна
Запити	Гнучка мова запитів MQL, Aggregation Framework	Стандарт SQL, потужні JOIN-операції
Транзакції (ACID)	Підтримка для кількох документів	Повна підтримка ACID, сильна консистенція

Продовж. табл. 3.3

Критерій	MongoDB	Реляційні бази даних
Гнучкість розробки	Висока, швидка ітерація	Нижча
Консистентність	Зазвичай остаточна узгодженість	Зазвичай сувора узгодженість
Приклади	MongoDB Atlas	PostgreSQL, MySQL, Oracle, SQL Server

Для проєкту веб-застосунку, що працює з різнорідними даними, гнучкість документної моделі MongoDB є суттєвою перевагою [45, 28]. Вона дозволяє легко зберігати та змінювати структури даних без складних міграцій, що прискорює розробку [45]. Можливість зберігати пов'язані дані в одному документі може спростити запити порівняно з необхідністю виконувати JOIN-операції у SQL [45]. Горизонтальна масштабованість забезпечує потенціал для зростання застосунку [45, 28]. MongoDB добре інтегрується з середовищем Node.js (на якому працює Next.js) та має офіційні драйвери, що спрощує взаємодію з базою даних з боку бекенду (API шляхи в Next.js) [45]. Використання таких функцій, як індексація та агрегація, дозволить забезпечити достатню продуктивність для запитів [28]. Враховуючи ці фактори, MongoDB представляється найбільш відповідною СКБД для задоволення потреб даного проєкту у зберіганні даних користувачів, налаштувань та метаданих медичної документації.

3.1.5 Рішення для автентифікації NextAuth

Автентифікація користувачів є невід'ємним компонентом безпеки будь-якого веб-застосунку, особливо медичного, де захист даних пацієнтів та лікарів має найвищий пріоритет. Процес автентифікації передбачає перевірку особи користувача перед наданням доступу до захищених ресурсів системи. Для реалізації автентифікації у даному проєкті, побудованому на Next.js, було обрано спеціалізовану бібліотеку NextAuth.js.

Обґрунтування вибору рішення NextAuth для автентифікації

NextAuth.js – це повноцінне рішення для автентифікації з відкритим кодом, спеціально розроблене для спрощення цього процесу у застосунках

Next.js [36, 48]. Воно абстрагує значну частину складності, пов’язаної з реалізацією безпечних потоків входу, управлінням сесіями та інтеграцією з різними провайдерами автентифікації [48].

Вибір NextAuth.js обґрунтовується кількома ключовими перевагами для даного проєкту. Насамперед, бібліотека вирізняється глибокою інтеграцією з Next.js. Вона створена з урахуванням особливостей фреймворку, включаючи різні стратегії рендерингу та API шляхів [36, 48]. NextAuth.js надає зручні хуки та серверні методи для доступу до інформації про сесію як на клієнті, так і на сервері, що істотно спрощує захист сторінок та API-ендпоінтів [36, 48].

Важливою перевагою є простота реалізації різних методів автентифікації. NextAuth.js «з коробки» підтримує багато стратегій, включаючи автентифікацію за логіном/паролем, що є першочерговою потребою для медичного застосунку, та інтеграцію з популярними OAuth-провайдерами [36, 48]. Це дозволяє легко налаштувати потрібні методи входу для різних категорій користувачів.

Бібліотека реалізує стандартні практики захисту, такі як запобігання CSRF-атакам, використання підписаних і зашифрованих JWT або сесій на основі бази даних, а також безпечні налаштування для cookie [36, 48]. Застосування перевіреного рішення знижує ризики, пов’язані з можливими помилками при самостійній реалізації складних протоколів безпеки [48].

NextAuth.js також керує створенням, перевіркою та оновленням сесій, пропонуючи гнучкі налаштування [48]. Використання SessionProvider робить дані сесії легкодоступними в усьому застосунку через React Context API та хук useSession, оптимізуючи цей процес [36, 48].

Наявність адаптерів для баз даних, зокрема для MongoDB, є ще одним важливим фактором [36, 48]. Це дозволяє без зайвих зусиль зберігати дані користувачів, їхніх акаунтів та сесій безпосередньо у базі даних проєкту, що є необхідним для персистентного зберігання інформації в медичному застосунку [48].

Також NextAuth.js пропонує значну гнучкість та можливості кастомізації. Розробники можуть створювати власні сторінки для процесів автентифікації, а також використовувати функціональні колбеки для втручання у процес автентифікації на різних етапах, наприклад, для додавання специфічних даних користувача до об’єкта сесії чи JWT-токену [36, 48].

Порівняння NextAuth.js з альтернативними підходами

Вибір NextAuth.js здійснювався з урахуванням інших можливих рішень для автентифікації. Сторонні платформи автентифікації, такі як Auth0 чи Okta, надають готову інфраструктуру та широкий функціонал, але можуть бути дорогими при масштабуванні та створювати залежність від постачальника [48]. Passport.js є гнучкою middleware-бібліотекою для Node.js, але її інтеграція в Next.js вимагає більше ручної конфігурації порівняно з NextAuth.js. Власна реалізація автентифікації дає повний контроль, але є надзвичайно складною, часозатратною та несе високі ризики безпеки [48].

У табл. 3.4 наведено порівняння NextAuth.js з основними альтернативами.

Таблиця 3.4

Порівняння підходів до автентифікації у Next.js

Критерій	**NextAuth**	**BaaS / Сторонні сервіси (Auth0, Okta)**	**Passport.js (в Next.js)**	**Власна реалізація**
Тип рішення	Бібліотека (Open Source)	Платформа / Сервіс (SaaS)	Middleware-бібліотека (Open Source)	Повністю кастомний код
Інтеграція з Next.js	Дуже висока, оптимізована	Зазвичай через SDK/API, може потребувати адаптації	Потребує інтеграції в API routes/SSR, менш нативна	Потребує повної реалізації з урахуванням Next.js
Складність реалізації	Низька/Середня	Низька (для базових випадків)	Середня/Висока	Дуже висока
Гнучкість/Контроль	Високий (Open Source, колбеки)	Обмежений конфігурацією сервісу	Дуже високий	Максимальний

Продовж. табл. 3.4

Критерій	NextAuth	BaaS / Сторонні сервіси (Auth0, Okta)	Passport.js (в Next.js)	Власна реалізація
Безпека «з коробки»	Висока (реалізує стандарти)	Дуже висока (спеціалізований сервіс)	Залежить від реалізації, потребує налаштування сесій	Залежить повністю від розробника, високий ризик помилок
Управління БД/ Користувачами	Через адаптери (включаючи MongoDB) або без БД	Зазвичай вбудоване у сервіс	Потребує власної реалізації	Потребує повної власної реалізації
Підтримка провайдерів	Широка (OAuth, Credentials, Email)	Зазвичай дуже широка	Дуже широка	Потребує реалізації для кожного провайдера
Вартість	Безкоштовно	Є безкоштовні тарифи, але може бути дорого при масштабуванні	Безкоштовно	Витрати на розробку та підтримку

NextAuth.js є оптимальним вибором для даного проєкту, оскільки він пропонує збалансоване поєднання простоти використання, гнучкості, безпеки та тісної інтеграції з Next.js. Він дозволяє швидко реалізувати надійну автентифікацію за логіном/паролем, необхідну для медичного застосунку, та легко інтегрується з обраною базою даних MongoDB через адаптер. Використання готового, добре підтримуваного рішення з відкритим кодом знижує ризики безпеки та витрати на розробку порівняно з власною реалізацією. Водночас він надає більше контролю та є економічно вигіднішим рішенням порівняно зі сторонніми BaaS-платформами.

3.2 Технології для реалізації DICOM-переглядача Cornerstone.js та аналіз альтернатив

Ключовим елементом розроблюваного веб-застосунку є можливість перегляду та базової взаємодії з медичними зображеннями у стандарті DICOM безпосередньо у веб-браузері. Це вимагає використання

спеціалізованих програмних інструментів, здатних коректно інтерпретувати дані DICOM та візуалізувати їх за допомогою веб-технологій. Для реалізації цього функціоналу в рамках даного проєкту було обрано бібліотеку Cornerstone.js.

3.2.1 Обґрунтування вибору бібліотеки Cornerstone.js

Cornerstone.js – це легковагова JavaScript бібліотека з відкритим кодом, спеціально розроблена для відображення медичних зображень у веб-середовищі [14, 10]. Вона не прив'язана до конкретного формату даних чи транспортного механізму, що робить її гнучкою та адаптованою [10]. Архітектура Cornerstone.js є модульною і включає ядро, що відповідає за рендеринг зображень на HTML5 Canvas, набір інструментів для інтерактивної взаємодії (панорамування, масштабування, зміна вікна яскравості/контрастності, вимірювання), механізм завантажувачів зображень, серед яких ключовим є cornerstone-wado-image-loader для роботи з DICOM через DICOMweb, та спеціалізований cornerstone-dicom-image-loader, який використовує dicom-parser для підготовки даних до візуалізації [14]. Новіші розробки в екосистемі, такі як cornerstone3D, розширюють можливості бібліотеки для роботи з тривимірними даними [42].

Вибір Cornerstone.js для даного веб-застосунку обґрунтований кількома ключовими перевагами. По-перше, її веб-орієнтованість: бібліотека написана на чистому JavaScript і призначена для роботи безпосередньо у браузері, що ідеально вписується в архітектуру застосунку на базі Next.js та React [14, 10]. По-друге, продуктивність: Cornerstone використовує HTML5 Canvas для рендерингу, а новіші розробки в екосистемі активно використовують WebGL для прискореного 2D та 3D рендерингу, що важливо при роботі з великими серіями зображень [42, 14]. По-третє, основний функціонал: надається необхідний набір інструментів для базового аналізу та маніпуляції зображеннями, що відповідає вимогам проєкту [14]. По-четверте, модульність та інтеграція: будучи бібліотекою, а не готовим застосунком, Cornerstone.js дозволяє глибоко інтегрувати компонент перегляду в

кастомний інтерфейс, створений за допомогою React. Це надає повний контроль над зовнішнім виглядом та логікою взаємодії, на відміну від вбудовування сторонніх готових переглядачів [42, 10]. Вона також добре інтегрується з бекенд-системами, такими як Orthanc, через їх API [10]. По-п’яте, відкритий код та спільнота: бібліотека є безкоштовною, має відкритий вихідний код та активну спільноту розробників, що сприяє її розвитку та підтримці [42, 52, 14]. А також її розширюваність дозволяє створювати власні інструменти та завантажувачі зображень, якщо стандартних недостатньо [14].

3.2.2 Порівняльний аналіз DICOM-переглядачів

Незважаючи на вибір бібліотеки Cornerstone.js як основи для реалізації функціоналу перегляду DICOM у даному проєкті, важливо розуміти контекст існуючих альтернативних рішень. Ринок програмного забезпечення для роботи з медичними зображеннями пропонує широкий спектр інструментів, які можна умовно розділити на веб-орієнтовані та десктопні переглядачі [42, 52, 44, 12].

Веб-орієнтовані DICOM-переглядачі набули значної популярності завдяки зростанню попиту на телемедичні рішення та можливості доступу до зображень з будь-якого пристрою через браузер без необхідності інсталяції [42, 52]. Серед них варто відзначити OHIF Viewer, який часто використовує Cornerstone.js як основу, та є повноцінним відкритим фреймворком з багатим функціоналом, але його вбудовування може обмежувати гнучкість UI [42, 52, 10]. Weasis є кросплатформним відкритим рішенням з хорошими інтеграційними можливостями, але потенційно нижчою продуктивністю веб-версії [42, 44, 12]. DWV – це мінімалістичний відкритий JS/HTML5 переглядач для базових завдань, що може працювати без PACS, але має обмежений функціонал [42, 52]. MedDream – комерційний веб-переглядач, часто пов’язаний із хмарним сервісом DicomLibrary [42, 11]. Існують також інші рішення, як Med3Web, Papaya, IMAIOS Viewer, PostDICOM та платформи для співпраці [42, 52, 11, 12].

Десктопні DICOM-переглядачі, як правило, пропонують найширший функціонал та найкращу продуктивність, особливо для складних 3D маніпуляцій, але вимагають інсталяції та є платформо-залежними [44, 12]. Прикладами є потужні OsiriX MD та Horos [44, 12]; швидкий та функціональний RadiAnt [44, 12]; надзвичайно потужна дослідницька платформа 3D Slicer [44]; та простий безкоштовний MicroDicom [44]. Ключовим недоліком десктопних рішень для даного проєкту є їхня несумісність з вимогою створення універсального веб-застосунку, доступного з будь-якого браузера без інсталяції [42, 44].

Для узагальнення ключових відмінностей між обраною бібліотекою та найбільш релевантними альтернативами наведено скорочену порівняльну табл. 3.5, яка базується на аналізі джерел.

Таблиця 3.5

Порівняльний аналіз Cornerstone.js та альтернативних DICOM-переглядачів [42, 52, 44, 12, 14, 10].

Критерій	Cornerstone.js (бібліотека)	OHIF Viewer (фреймворк/застос.)	Weasis (веб/десктоп)	Horos (десктоп)
Платформа/ Тип	Web (JS бібліотека)	Web (JS фреймворк)	Web/Desktop (Java)	macOS
Ліцензія	MIT (OSS)	MIT (OSS)	EPL/LGPL (OSS)	LGPL (OSS)
Основна технологія	JavaScript, HTML5 Canvas, WebGL	JavaScript (React/Vue), Cornerstone.js	Java, JavaScript	Objective-C/Swift
Ключові 2D інструменти	W/L, Pan/Zoom, Вимір., Анот.	W/L, Pan/Zoom, Вимір., Анот., Cine	W/L, Pan/Zoom, Вимір., Анот., Cine, Фільтри	W/L, Pan/Zoom, Вимір., Анот., Cine
Ключові 3D інструменти	Базові (з cornerstone3D), потребує інтеграції	MPR, MIP, VR, Сегм.	MPR, MIP (базові)	MPR, MIP, VR, Surface Rend., Fusion
Інтеграція (PACS/Web)	Завантажувачі WADO/REST, інтегр. з API (Orthanc)	Інтеграція з PACS (DICOMweb/DIMSE)	Інтеграція з PACS/HIS/RIS (DIMSE/Web)	DIMSE (базовий сервер/клієнт)

Продовж. табл. 3.5

Критерій	Cornerstone.js (бібліотека)	OHIF Viewer (фреймворк/застос.)	Weasis (веб/десктоп)	Horos (десктоп)
Продуктивність	Залежить від реалізації, потенційно висока	Добра, швидкий рендеринг	Може поступатися нативним (веб)	Висока
Основне призначення	Компонент для вбуд. в веб-застосунки	Гнучкий веб-переглядач/фреймворк	Клінічний перегляд, інтеграція	Навчання, дослідження, діагностика

Проаналізувавши альтернативи, можна зробити висновок, що використання саме бібліотеки Cornerstone.js в рамках розроблюваного веб-застосунку на Next.js/React надає оптимальний баланс для досягнення поставлених цілей. На відміну від готових веб-застосунків, Cornerstone.js дозволяє повністю контролювати інтеграцію компонента перегляду в користувацький інтерфейс, забезпечуючи єдиний дизайн та досвід користувача в межах усього застосунку. Це також дозволяє уникнути надлишкового функціоналу, який може бути присутнім у готових рішеннях, і сфокусуватися на реалізації саме тих інструментів, що необхідні для даного проєкту. Порівняно з десктопними програмами, підхід на основі Cornerstone.js забезпечує ключову перевагу – кросплатформність та доступність через веб-браузер без необхідності встановлення ПЗ [42, 44]. Хоча такий підхід вимагає більших зусиль на розробку оточуючого інтерфейсу та логіки активації інструментів, він дозволяє створити кастомізоване та гнучке рішення, що повністю відповідає специфічним вимогам проєкту щодо створення веб-застосунку для роботи з DICOM-зображеннями.

3.3 Аналіз та обґрунтування вибору PACS-серверу

Для ефективного управління медичними зображеннями у стандарті DICOM, особливо в контексті веб-застосунку, який передбачає завантаження,

зберігання та доступ до досліджень, необхідним компонентом архітектури є сервер PACS. Цей сервер виступає центральним сховищем для DICOM-даних, забезпечуючи їх індексацію, зберігання та надання доступу іншим системам, зокрема, розроблюваному веб-переглядачу. У даному проєкті як PACS-рішення було обрано Orthanc.

3.3.1 Обґрунтування вибору Orthanc

Orthanc – це вільний, відкритий, легковаговий сервер DICOM, який може функціонувати як базовий PACS [29, 25]. Він був розроблений з метою спрощення роботи з медичними зображеннями та покращення інтероперабельності, особливо в дослідницьких та специфічних клінічних середовищах, часто виступаючи як доповнення до основного PACS лікарні (загальну схему екосистеми див. рис. 3.2) [29].

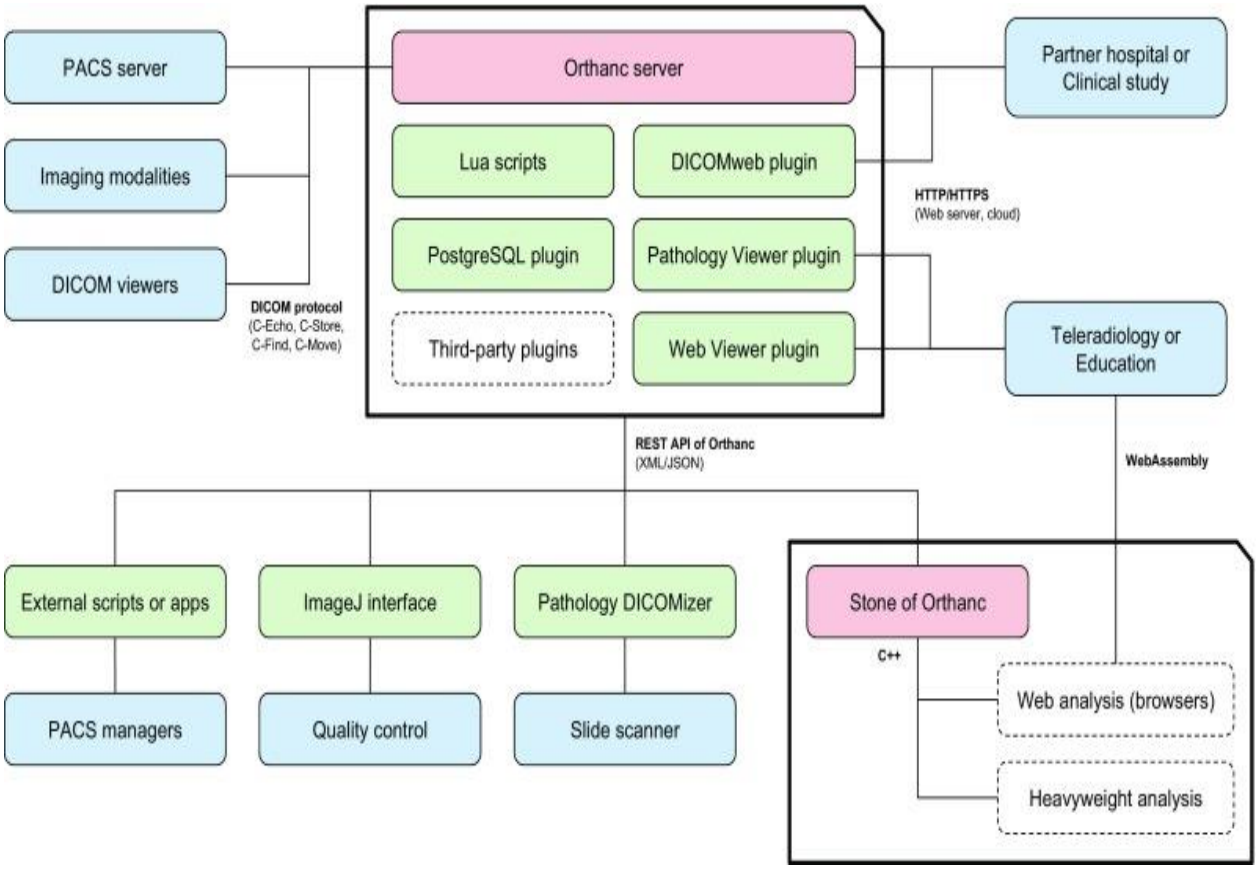

Рисунок 3.2 – Загальна схема екосистеми Orthanc [29]

Orthanc спроєктований як автономний сервер з мінімальними системними вимогами, може працювати на різному обладнанні, не потребує зовнішніх залежностей та постачається з вбудованою базою даних SQLite, що спрощує його встановлення та конфігурацію (архітектуру програмних шарів див. рис. 3.3) [29]. Також перевагами є відкритий код та безкоштовне використання, підкріплене активною спільнотою та доступною комерційною підтримкою [29, 44]. Сервер є кросплатформним, забезпечуючи роботу на Windows, macOS та Linux [29, 44].

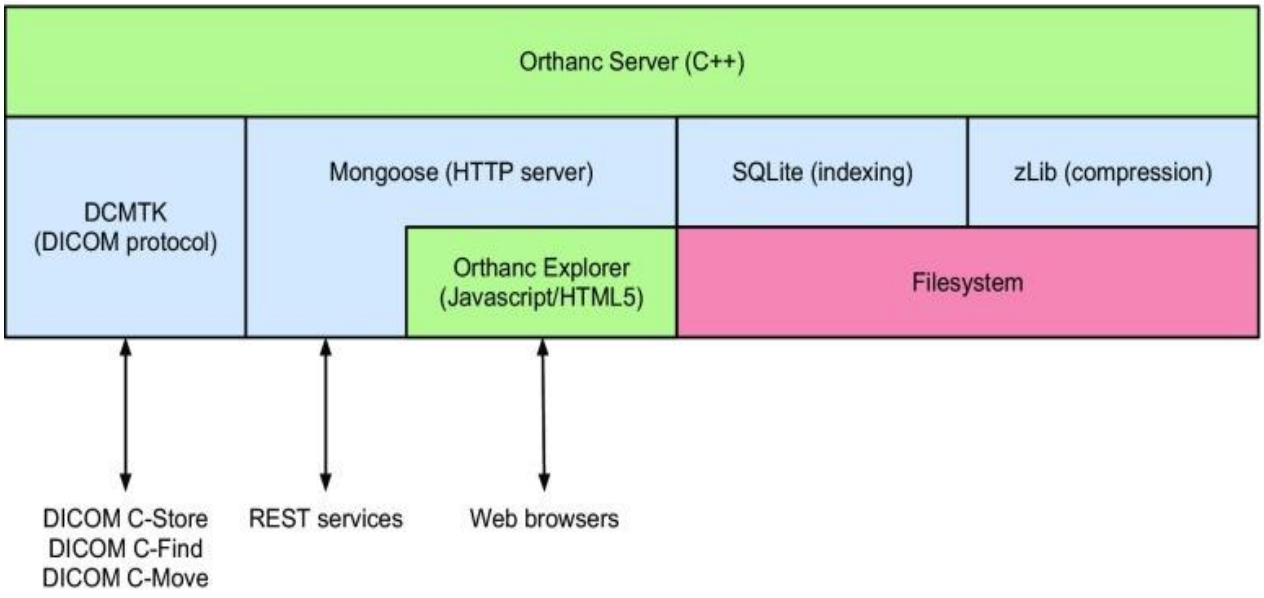

Рисунок 3.3 – Архітектура програмних шарів сервера Orthanc [29]

Критично важливою для даного веб-застосунку є орієнтація Orthanc на веб-інтеграцію через потужний REST API. Цей API дозволяє повністю керувати сервером за допомогою стандартних HTTP-запитів та формату JSON, що значно спрощує інтеграцію з веб-застосунками [38, 10, 29, 25]. Бекенд може взаємодіяти з PACS-сервером, використовуючи звичні веб-технології замість складної реалізації традиційних DICOM-мережевих протоколів [29]. Саме REST API та підтримка DICOMweb уможливлюють реалізацію функціоналу завантаження та доступу до зображень у проєкті.

Також Orthanc забезпечує хорошу сумісність зі стандартом DICOM, зокрема, завдяки використанню бібліотеки DCMTK [29].

3.3.2 Порівняльний аналіз PACS-серверів

Вибір Orthanc також розглядався у порівнянні з іншими класами PACS-рішень, зокрема, з комплексними серверами типу DCM4CHEE та хмарними платформами.

Порівнюючи Orthanc з DCM4CHEE, важливо відзначити їх різне позиціонування [29]. DCM4CHEE, як правило, розглядається як комплексне рішення для великих медичних закладів, здатне замінити комерційні PACS, та має розширену підтримку інтеграційних профілів і взаємодії з лікарняними інформаційними системами через HL7. Orthanc, натомість, часто використовується як більш гнучкий доповнюючий елемент до існуючої інфраструктури або як самостійне рішення для специфічних завдань, досліджень чи невеликих установ. Ключовими перевагами Orthanc у цьому порівнянні є значно простіша інсталяція та конфігурація, менші системні вимоги та сильний акцент на інтеграцію через REST API, що є пріоритетом для розробки веб-орієнтованих застосунків. Хоча DCM4CHEE може пропонувати більше «enterprise» функцій «з коробки», його складність для цілей даного проєкту є надлишковою [29].

Іншою альтернативою є використання хмарних PACS-платформ. Такі сервіси пропонують керовану інфраструктуру з високою масштабованістю та доступністю, знімаючи з користувача завдання адміністрування серверів. Взаємодія з ними зазвичай відбувається через DICOMweb API. Однак, такий підхід пов’язаний з регулярними витратами, що залежать від обсягу збережених даних та інтенсивності використання сервісу, а також створює залежність від конкретного хмарного провайдера. Самостійне розгортання Orthanc надає повний контроль над даними, конфігурацією та витратами, що є більш доцільним для даного проєкту [29, 25].

У табл. 3.6 представлено узагальнене порівняння основних характеристик Orthanc, DCM4CHEE та хмарних PACS-рішень.

Таблиця 3.6

Порівняльний аналіз PACS-серверів Orthanc, DCM4CHEE та хмарних рішень [29]

Критерій	Orthanc	DCM4CHEE	Хмарні PACS-рішення
Тип/Призначення	Легковаговий VNA/PACS, часто як доповнення	Повномасштабний PACS для підприємств	Керований сервіс зберігання/API
Ліцензія	OSS (GPLv3)	OSS (різні ліцензії)	Комерційна
Складність встан.	Низька	Вища	Залежить від провайдера
Основна технологія	C++	Java	Залежить від провайдера
Інтеграція	Акцент на REST API, DICOMweb (плагін)	Акцент на HL7, IHE, DIMSE	Власні API, DICOMweb
Гнучкість/Розшир.	Висока (Lua, плагіни)	Висока, але складніша	Обмежена провайдером
Масштабованість	Добра (особливо з плагіном БД)	Дуже добра для великих лікарень	Дуже висока
Вартість	Безкоштовне ПЗ, витрати на інфраструктуру	Безкоштовне ПЗ, вищі витрати на інфраструктуру	Оплата за використання/зберігання
Контроль/Хостинг	Повний (самостійний хостинг)	Повний (самостійний хостинг)	Обмежений (керований сервіс)

Враховуючи вимоги проєкту до легкої інтеграції з веб-застосунком через стандартні веб-протоколи, необхідність повного контролю над середовищем, переваги відкритого коду та відсутність потреби у складних функціях інтеграції корпоративного рівня, Orthanc є найбільш обґрунтованим вибором як PACS-сервер для даного веб-застосунку. Його архітектура та REST API дозволяють ефективно реалізувати необхідний функціонал зберігання та отримання DICOM-даних з боку Next.js.

Висновки до розділу 3

У даному розділі було проведено комплексний аналіз сучасного стану технологій та інструментальних засобів, що застосовуються при розробці веб-застосунків у медичній сфері. Розглянуто ключові тенденції, такі як

розвиток телемедицини, використання хмарних обчислень та Інтернету медичних речей, а також визначено основні виклики, пов’язані з безпекою даних, інтероперабельністю систем, продуктивністю та зручністю використання медичних веб-рішень.

На основі проведеного аналізу та порівняння альтернатив було сформовано та обґрунтовано вибір конкретних технологій для реалізації програмного продукту. Використання TypeScript забезпечує підвищену надійність та керованість коду завдяки статичній типізації. Фреймворк Next.js обраний за його інтегровані API шляхи та загальну продуктивність, що дозволяє створити сучасний та ефективний користувацький інтерфейс. Стилізація за допомогою Tailwind CSS забезпечує швидкість розробки та гнучкість дизайну. Документо-орієнтована база даних MongoDB забезпечує необхідну гнучкість для зберігання даних користувачів та медичної інформації. Автентифікація реалізована за допомогою NextAuth.js завдяки його тісній інтеграції з Next.js та MongoDB. Ключовими для роботи з DICOM-зображеннями стали бібліотека Cornerstone.js, що надає необхідні інструменти для візуалізації та взаємодії з зображеннями безпосередньо в браузері, та легковаговий PACS-сервер Orthanc, обраний за його відкритість, простоту розгортання та, що особливо важливо, наявність REST API для зручної інтеграції з веб-застосунком. Таким чином, обраний технологічний стек представляє собою сучасне, збалансоване та переважно відкрите рішення, яке відповідає специфічним вимогам проєкту та дозволяє ефективно реалізувати поставлені завдання.

РОЗДІЛ 4
ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ВЕБ-ЗАСТОСУНКУ

У даному розділі детально розглядається інформаційне забезпечення та програмна реалізація розробленого веб-застосунку, призначеного для керування медичною документацією та DICOM-зображеннями. Буде представлено ключові функціональні модулі системи, включаючи механізми автентифікації та розмежування доступу, управління даними пацієнтів та їх візитами, а також інтеграцію з PACS-сервером для обробки DICOM-досліджень. Окрему увагу буде приділено інтегрованому DICOM-переглядачу та його інструментарію. Також у розділі буде продемонстровано типові сценарії роботи користувачів різних ролей (лікаря, консультанта, пацієнта) для ілюстрації практичного застосування функціоналу. Додатково буде наведено діаграму бази даних веб-застосунку та представлено розрахунок економічного ефекту від впровадження розробки, що базується на функціонально-вартісному аналізі, проведеному під час переддипломної практики.

4.1 Діаграма бази даних веб-застосунку

На рис. 4.1 представлена схема бази даних MongoDB, що використовується у веб-застосунку. Ця база даних складається з трьох основних колекцій: User, Visit та Consultation.

Колекція User зберігає загальну інформацію про всіх користувачів системи, включаючи їхні ідентифікатори, імена, електронні адреси, паролі та ролі. Ця колекція є центральною, оскільки дані з неї використовуються для зв’язку з іншими колекціями.

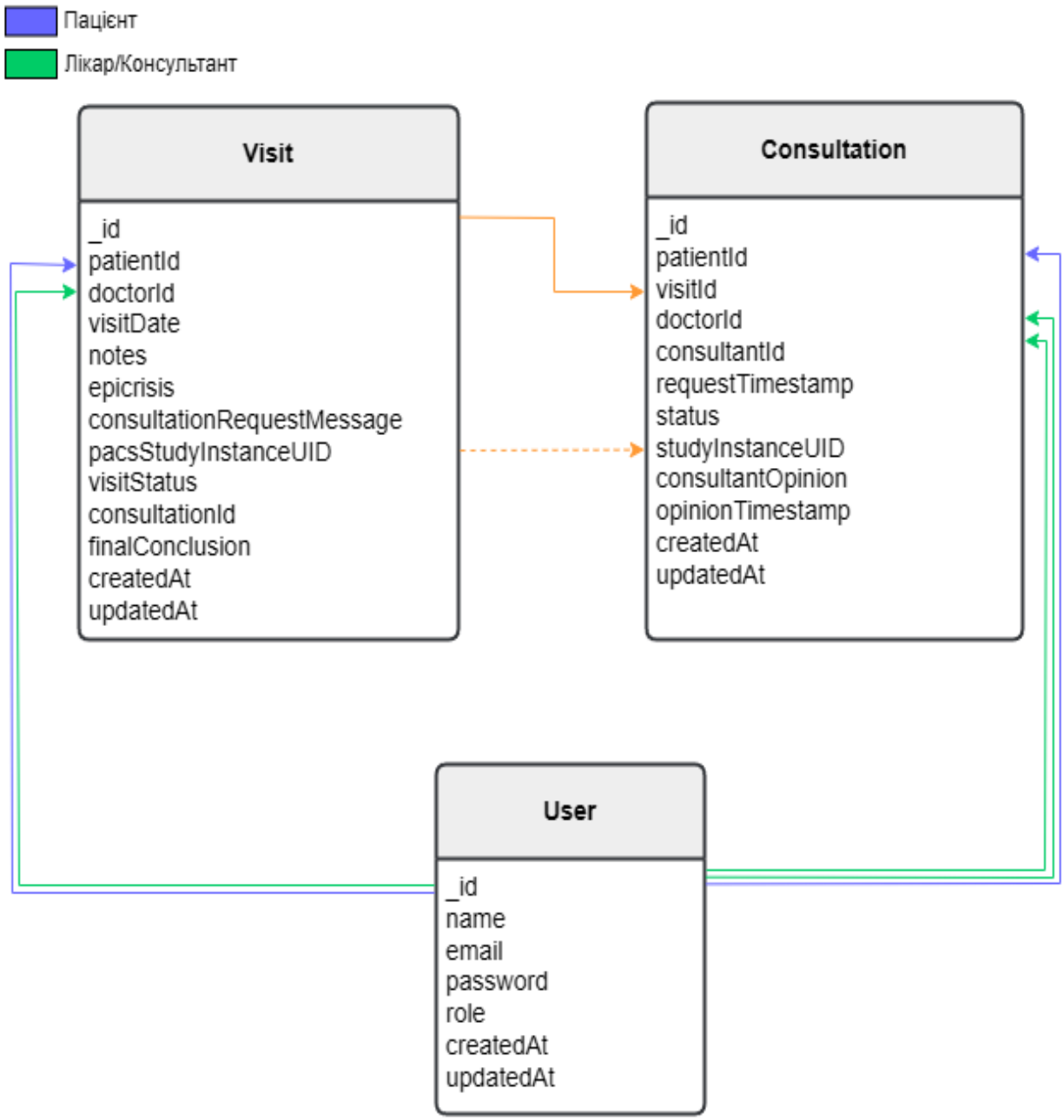

Рисунок 4.1 – Діаграма бази даних

Колекція Visit містить детальну інформацію про візити пацієнтів до лікарів. Зв’язок з колекцією User здійснюється через поля patientId та doctorId. Поле Visit.patientId посилається на User._id користувача з роллю «patient», а Visit.doctorId посилається на User._id користувача з роллю «doctor». Кожен запис у цій колекції представляє окремий візит і включає такі дані, як дата візиту, нотатки, епікриз, статус візиту, а також унікальний

ідентифікатор DICOM-дослідження, якщо таке було проведено під час візиту.

Колекція Consultation зберігає дані про консультації, які пов’язані з певним візитом. Зв’язок з колекцією User реалізований через поля patientId, doctorId, та consultantId. Поле Consultation.patientId посилається на User._id пацієнта. Поля Consultation.doctorId та Consultation.consultantId посилаються на User._id лікарів, що виконують ролі основного лікаря та консультанта відповідно. Зв’язок з колекцією Visit встановлюється через поле Consultation.visitId, яке посилається на Visit._id, пов’язуючи консультацію з конкретним візитом. Існує також логічний зв’язок між колекціями Visit та Consultation через поля Visit.pacsStudyInstanceUID та Consultation.studyInstanceUID. Цей зв’язок вказує на те, що консультація може стосуватися того ж DICOM-дослідження, що й візит, забезпечуючи цілісність даних медичного обстеження.

4.2 Реалізація ключових функціональних модулів

Модулі веб-застосунку відповідають за свою частину роботи, а разом вони забезпечують нормальне функціонування всієї системи. Давайте розглянемо кожен з них.

4.2.1 Система автентифікації та розмежування доступу

Перш за все, щоб дані були в безпеці і кожен бачив тільки те, що йому дозволено, я зробив систему входу та розмежування доступу (рис. 4.2). Все працює досить стандартно: користувач вводить свій логін та пароль. Якщо все вірно, система «впізнає» його і видає спеціальний тимчасовий «ключ» (сесійний токен). З цим токеном користувач може далі працювати із системою. У застосунку є різні ролі – адміністратор, лікар, консультант, пацієнт і для кожного чітко визначені свої права доступу.

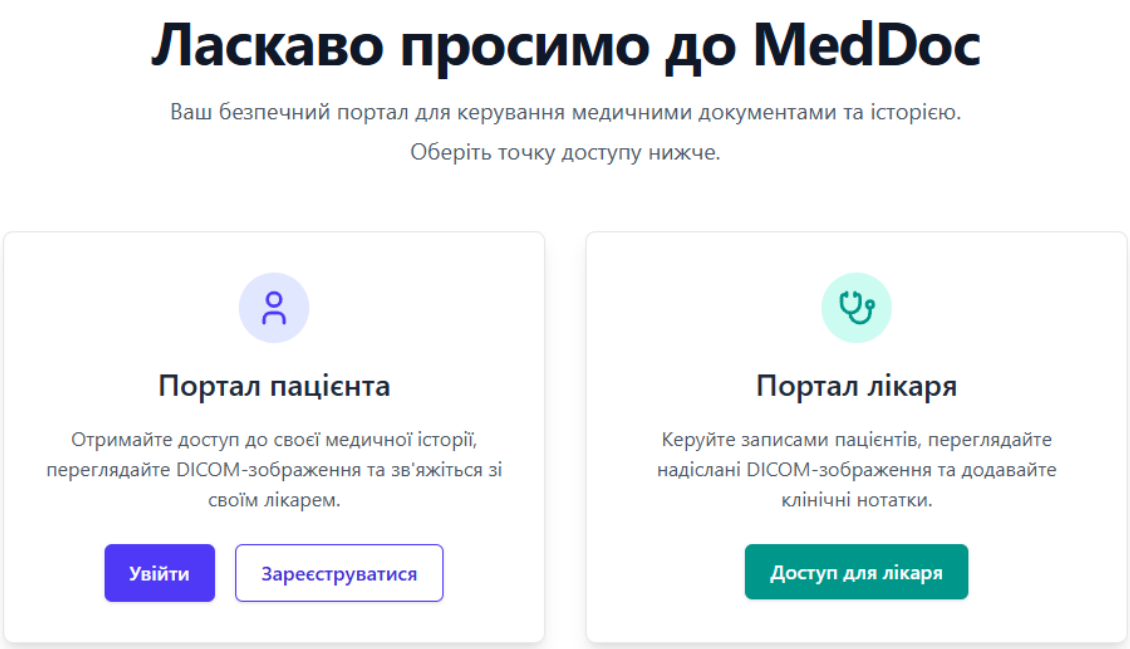

Рисунок 4.2 – Вітальна сторінка

Пацієнт має можливість зареєструватись самостійно, натиснувши кнопку «Зареєструватись» на вітальній сторінці, та заповнивши форму реєстрації (рис. 4.3).

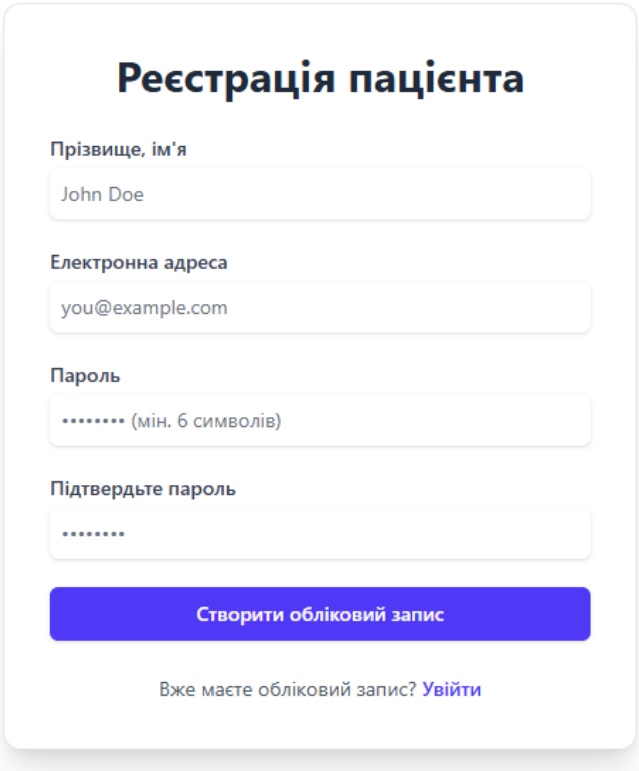

Рисунок 4.3 – Форма реєстрації пацієнта

Передбачені базові перевірки співпадіння паролю (рис. 4.4) та мінімальна кількість символів – 6 (рис. 4.5), з відповідними зрозумілими повідомленнями про помилку для користувача.

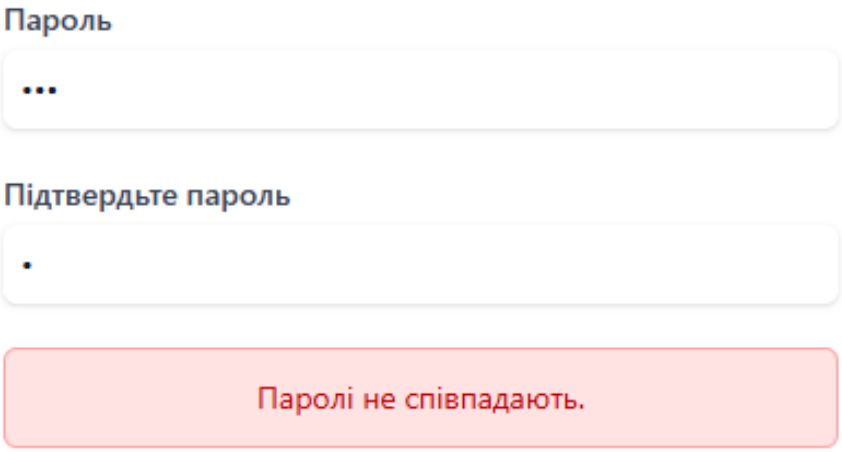

Рисунок 4.4 – Помилка про не співпадіння введених паролів

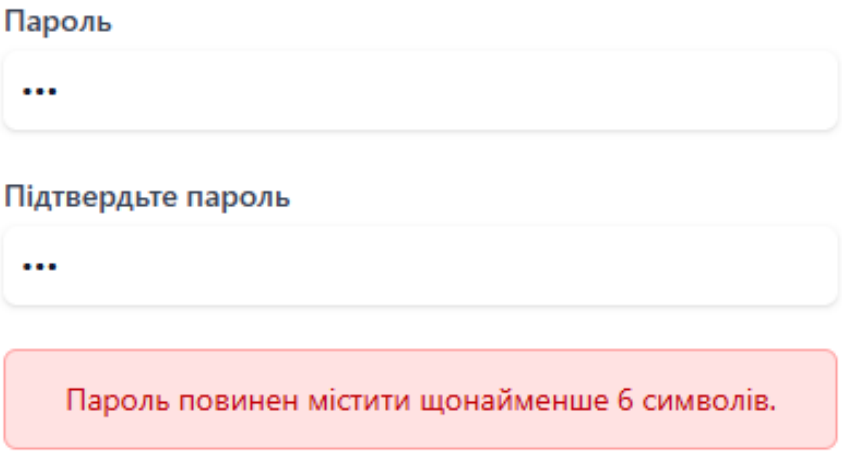

Рисунок 4.5 – Помилка про мінімальну кількість символів

Після успішної реєстрації з’являється відповідне повідомлення (рис 4.6), а також відбувається автоматичне перенаправлення користувача на форму автентифікації (рис 4.7).

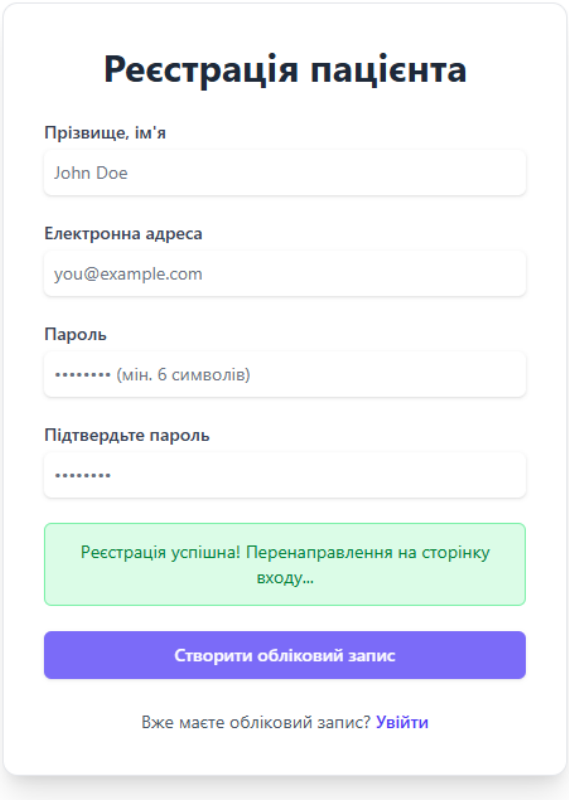

Рисунок 4.6 – Успішна реєстрація

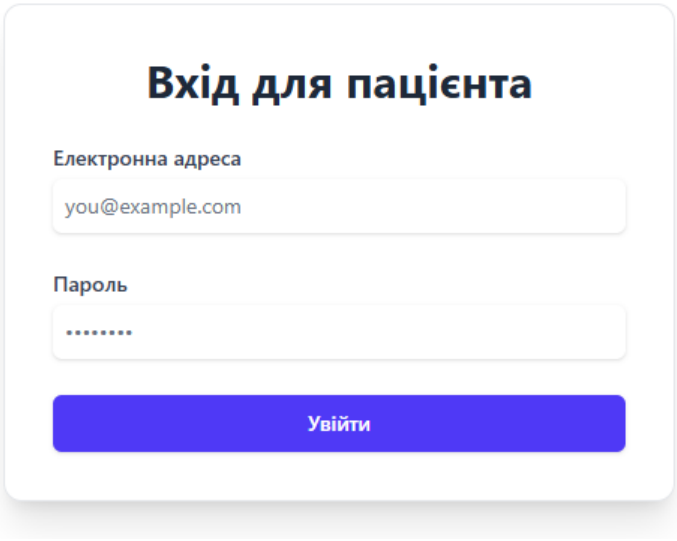

Рисунок 4.7 – Форма автентифікації (логіну) пацієнта

Якщо користувач введе неправильні дані – отримує відповідну помилку (рис. 4.8).

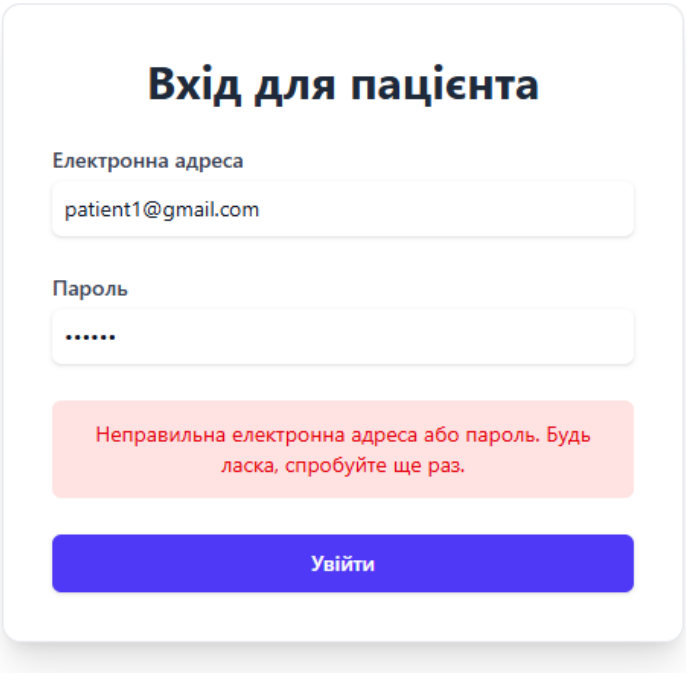

Рисунок 4.8 – Помилка про неправильні введені дані

Так як я зазначав вище – у застосунку присутня система розмежування, а саме є портал для пацієнтів та інший портал для лікарів і консультантів. Тому у відповідних формах входу користувачі отримують наступні помилки (рис. 4.9 - рис. 4.10):

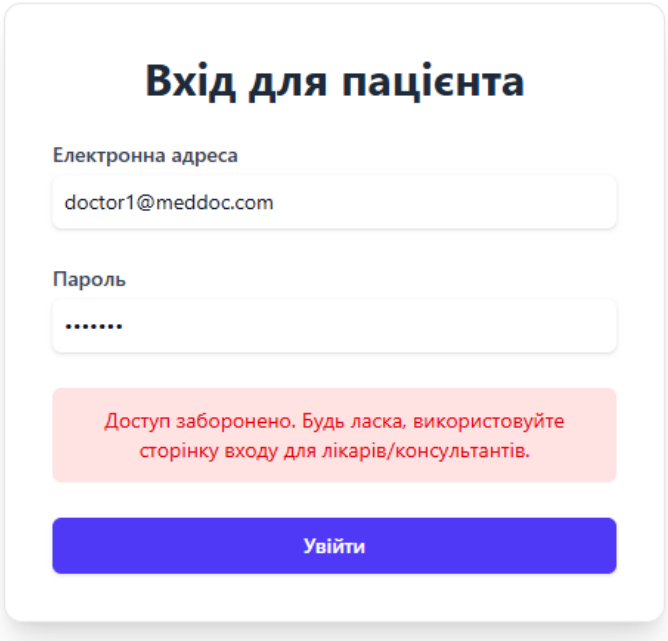

Рисунок 4.9 – Лікар намагається увійти через портал для пацієнтів

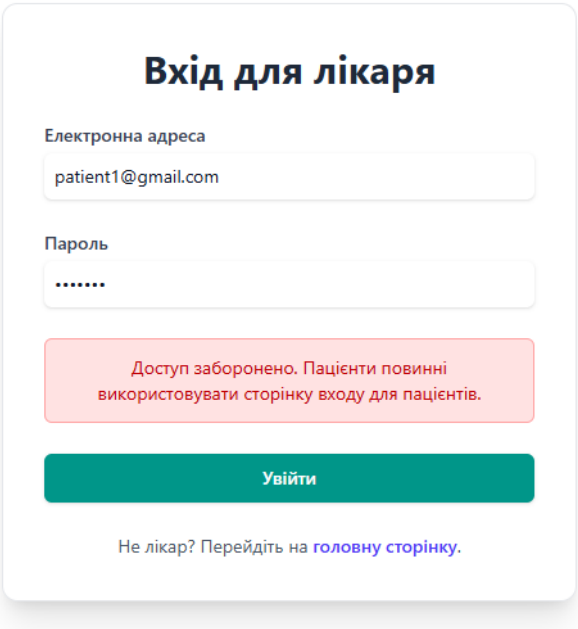

Рисунок 4.10 – Пацієнт намагається увійти через портал для лікарів/консультантів

Лікарі та консультанти, в свою чергу, можуть лише входити через свій портал (рис. 4.11), а реєстрація їх облікових записів відбувається виключно через адміністратора.

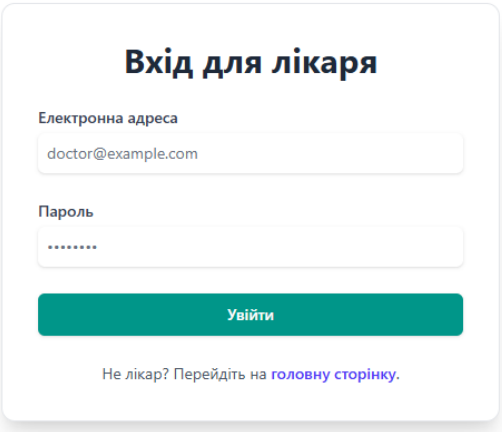

Рисунок 4.11 – Форма автентифікації (логіну) лікаря/консультанта

Адміністратор – це єдиний унікальний обліковий запис, який не може бути створено за допомогою веб-інтерфейсу. Це можливо зробити безпосередньо на рівні взаємодії з базою даних (якщо потенційно необхідно

додати ще такий обліковий запис). В межах цього веб-застосунку на панелі адміністратора знаходиться лише один блок – форма реєстрації лікаря або консультанта (рис. 4.12), хоча, звичайно, ця привілейована роль в майбутньому може мати набагато більше функціоналу.

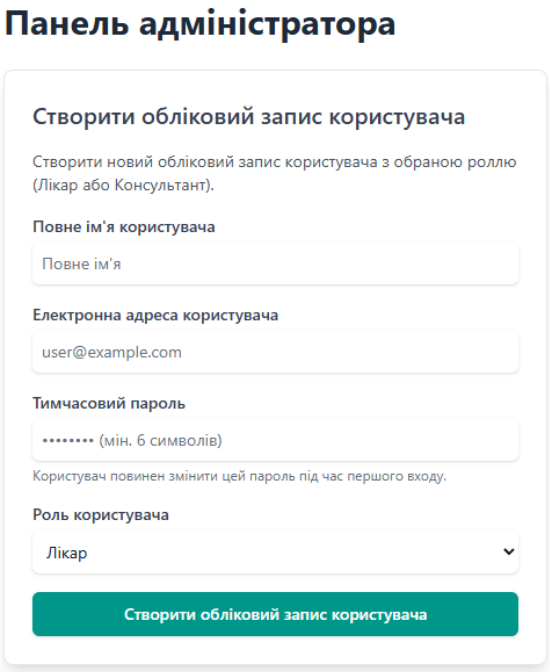

Рисунок 4.12 – Форма реєстрації лікаря/консультанта

4.2.2 Модуль управління пацієнтами

Цей модуль – основний інструмент для лікарів (рис. 4.13). Він дозволяє зручно працювати з базою даних пацієнтів. Можна легко знайти потрібного пацієнта, наприклад, за прізвищем чи номером картки. Коли пацієнта знайдено, відкривається його детальна картка. Там зібрана вся основна інформація: персональні дані, контакти і, головне, медична історія. Лікарі можуть не лише переглядати цю історію, а й доповнювати її – вносити нові діагнози, інформацію про алергії, хронічні хвороби чи проведені маніпуляції. Інтерфейс зроблено так, щоб лікар міг швидко знайти потрібну інформацію, адже в клінічній роботі час часто обмежений. Всі дані пацієнтів зберігаються в базі даних з урахуванням вимог конфіденційності.

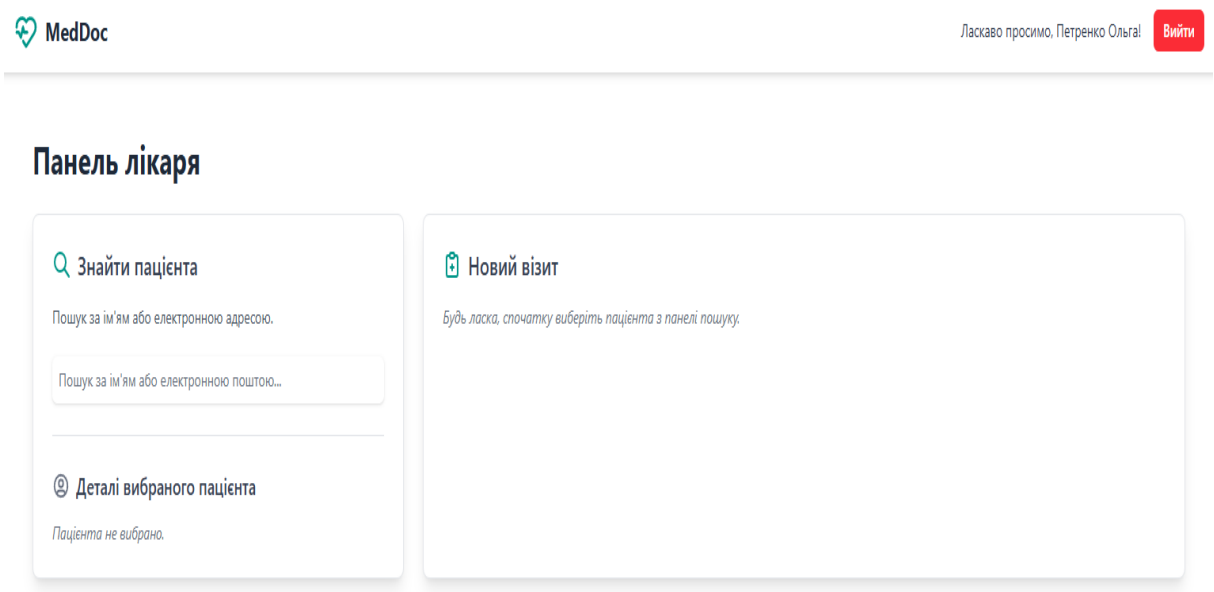

Рисунок 4.13 – Панель лікаря

Лікар може знайти пацієнта як за допомогою електронної пошти (рис. 4.14), так і за персональними даними (рис. 4.15).

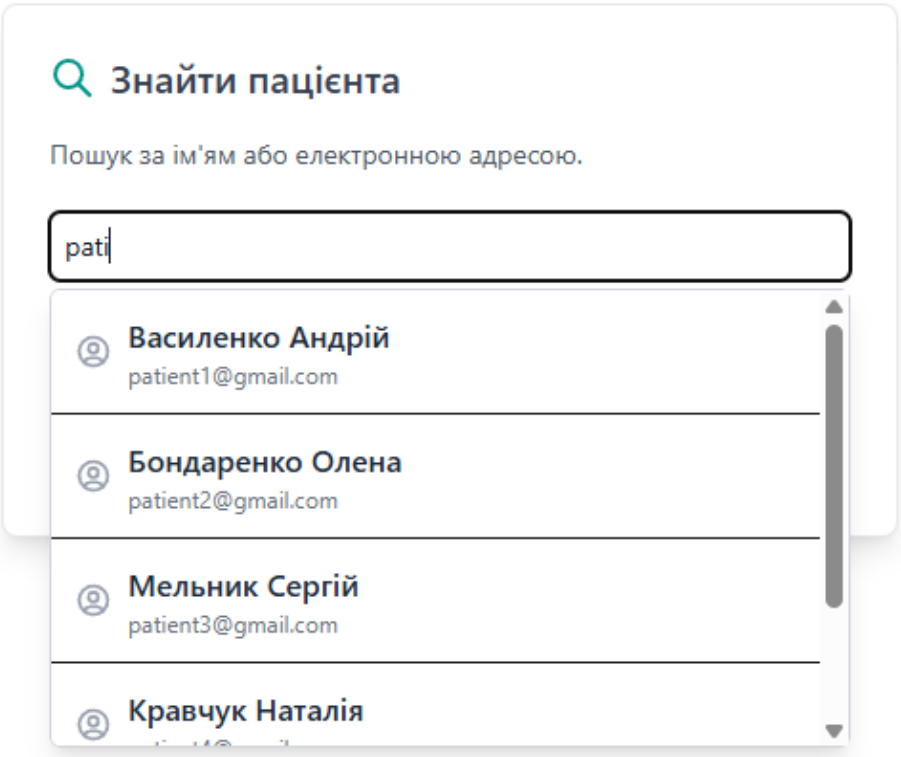

Рисунок 4.14 – Пошук пацієнтів за допомогою електронної пошти

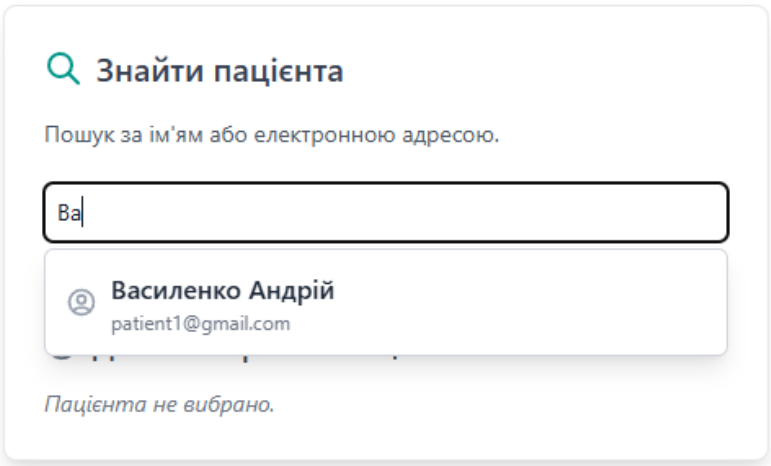

Рисунок 4.15 – Пошук пацієнтів за допомогою персональних даних

Після того як лікар обрав пацієнта зі списку, завантажується уся наявна інформація про нього (рис 4.16).

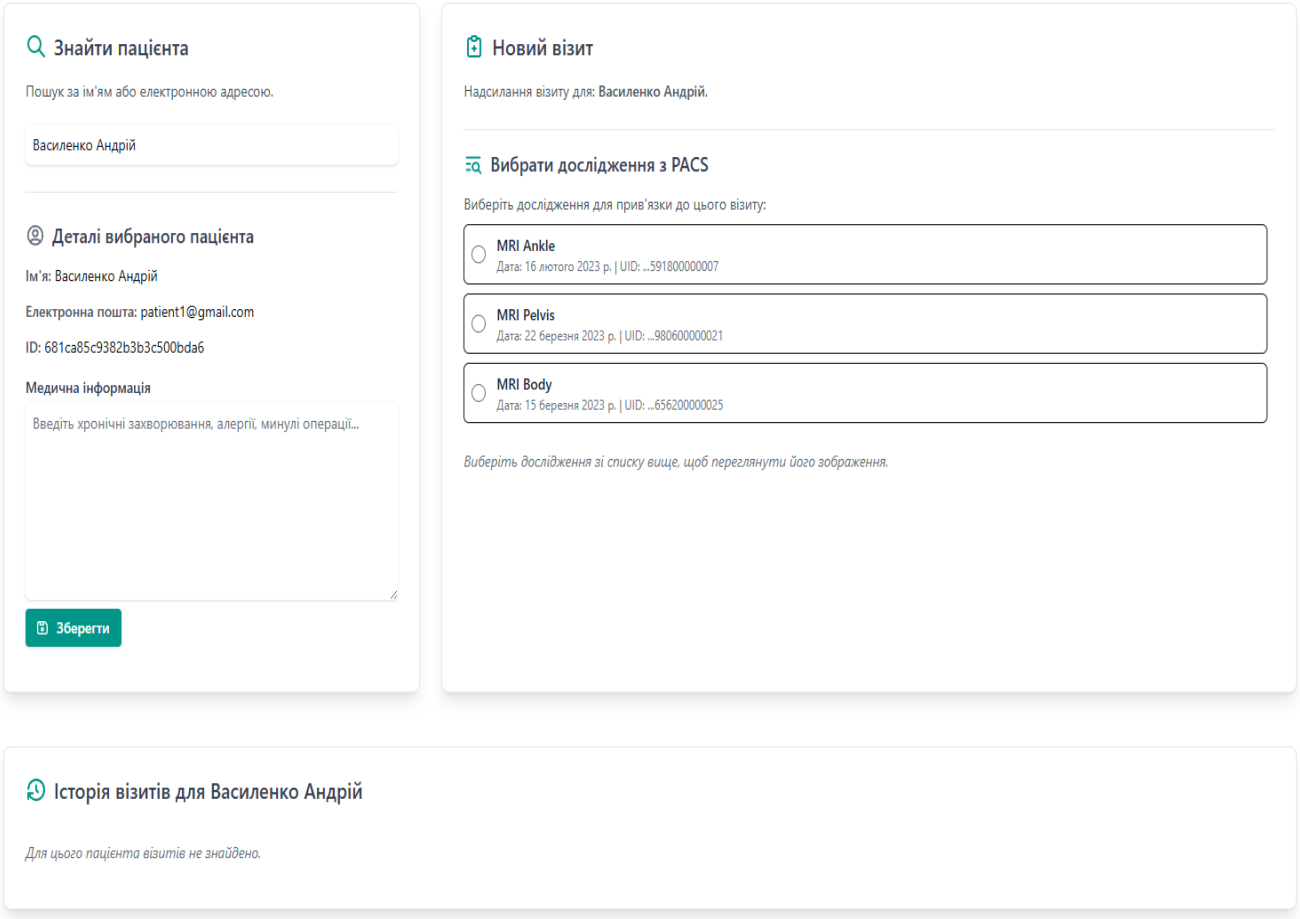

Рисунок 4.16 – Панель лікаря після обрання пацієнта

Розберемо кожен блок окремо: дані про пацієнта, створення візиту, та перегляд історії візитів пацієнта (візити див. 4.2.3 Модуль управління візитами).

Отож, дані про пацієнта (рис. 4.17) відображаються одразу під полем пошуку, кілька статичних полів, а також є поле для введення медичної інформації про пацієнта. Ця інформація може редагуватись виключно лікарем, а також відображається у кабінеті пацієнта.

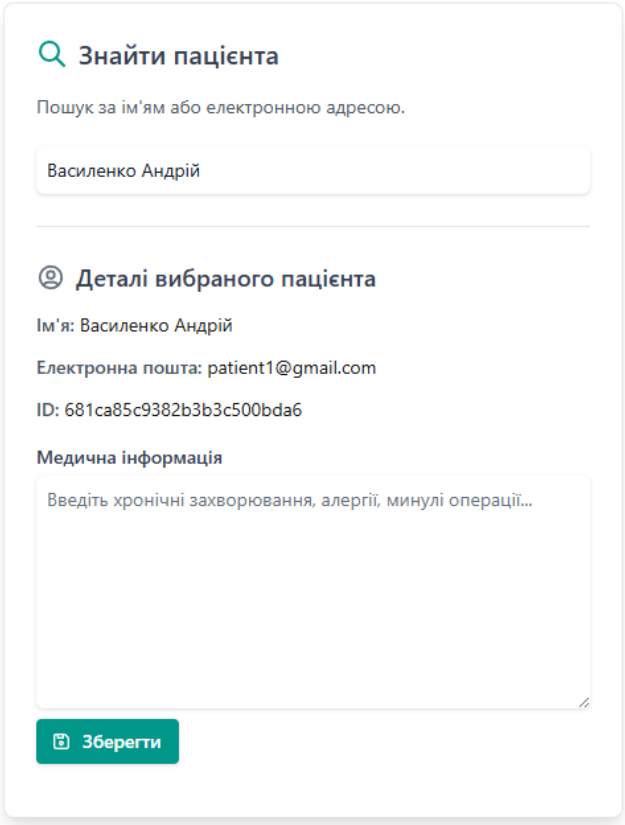

Рисунок 4.17 – Дані пацієнта

4.2.3 Модуль управління візитами

Як тільки лікар обрав пацієнта, із серверу PACS (див. 4.2.4 Модуль завантаження DICOM-досліджень) автоматично підвантажуються усі дослідження останнього. Коли лікар обирає потрібне дослідження (рис. 4.18) – йому розкривається повноцінна багатофункціональна форма (рис. 4.19), в яку входить: переглядач DICOM (див. 4.2.5 Інтегрований DICOM-

переглядач), нотатки до візиту, висновок, а також можливість запитати консультацію (див. 4.2.6 Модуль консультацій).

Рисунок 4.18 – Обране дослідження

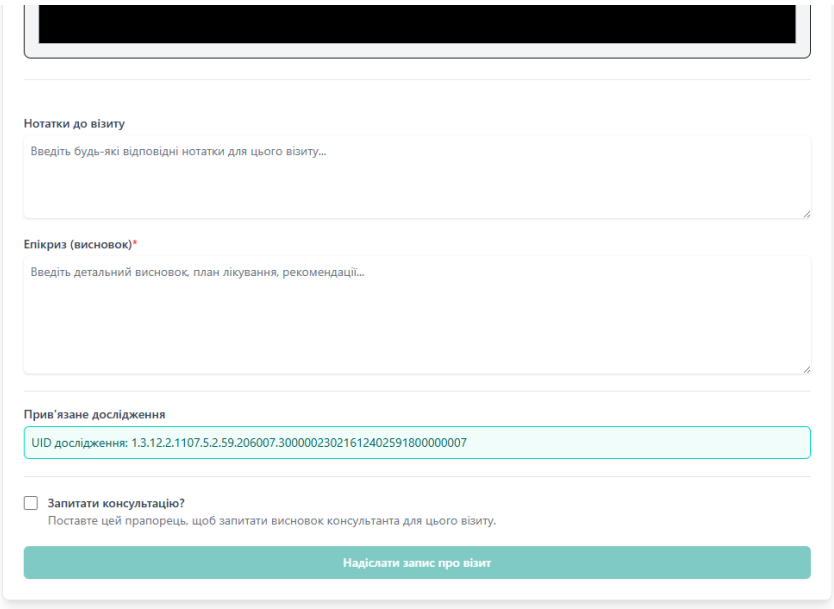

Рисунок 4.19 – Форма запису про візит

Також лікар може переглядати історію візитів пацієнта (рис. 4.20). У цьому блоці показується, які висновки робилися, нотатки, а також є можливість відкрити знімки, що стосуються минулих візитів (рис. 4.21). За допомогою цієї історії лікар швидко та ефективно може пригадати основні моменти лікування пацієнта.

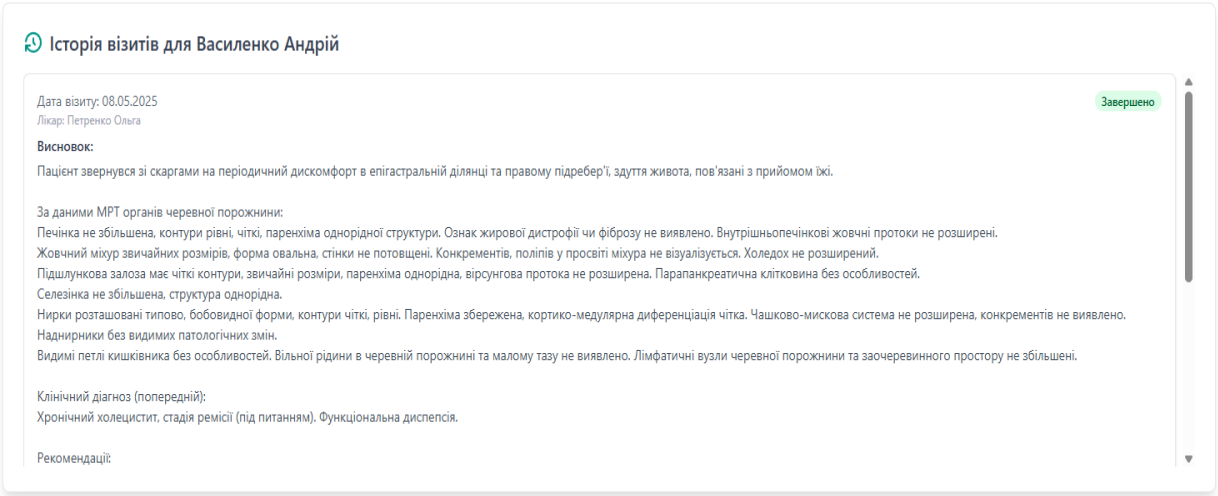

Рисунок 4.20 – Історія візитів пацієнта

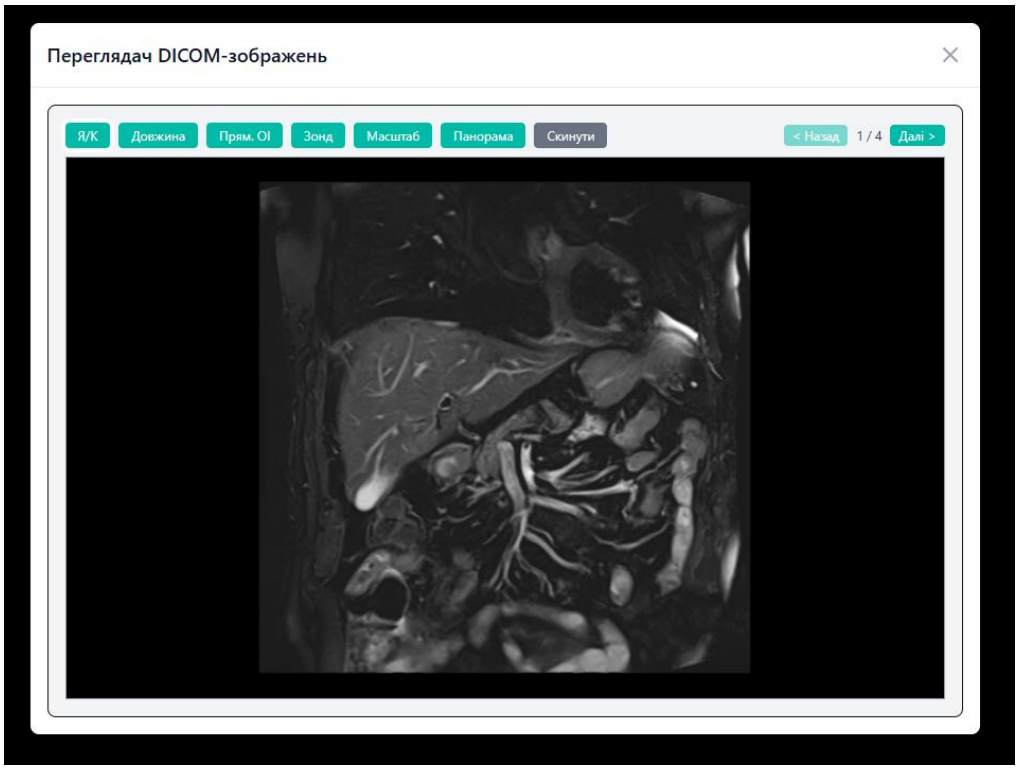

Рисунок 4.21 – Модальне вікно DICOM переглядача зображень минулого візиту

4.2.4 Модуль завантаження DICOM-досліджень

З другого розділу нам відомо, що стандарт DICOM за своєю суттю тісно пов’язаний із системами PACS – саме він дозволяє коректно передавати, зберігати та ідентифікувати ці дані. Для демонстрації взаємодії з PACS-системами, було змодельовано типовий робочий процес, де лікар-діагност після проведення дослідження надсилає отримані DICOM-файли до центрального сховища – системи PACS. У контексті нашого застосунку використовується Orthanc як такий PACS-сервер, де і зберігаються всі діагностичні зображення (рис. 4.22). DICOM стандарт за допомогою своїх вбудованих тегів записує всю метаінформацію, за рахунок чого ми й визначаємо, які дослідження пов’язані із цим пацієнтом.

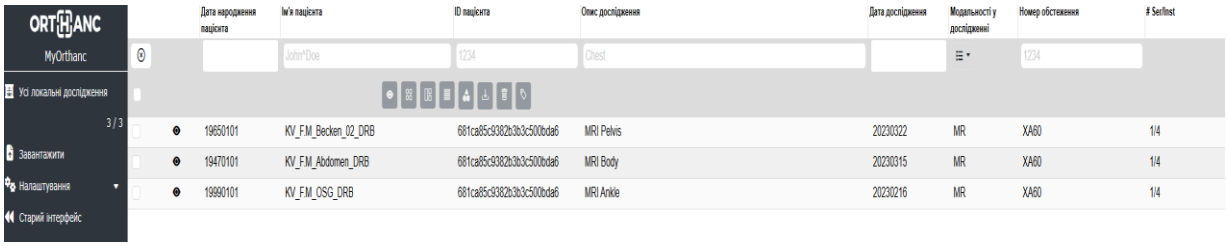

Рисунок 4.22 – Інтерфейс PACS-серверу Orthanc

На даний момент, це єдиний варіант завантажити медичні зображення, щоб потім лікар міг взаємодіяти з ними. Можливо хорошим варіантом буде додати до форми візиту можливість завантажувати файли вручну лікарем, проте я не робив цього функціоналу через відсутність точної інформації чи це слушна ідея. Тут потрібен персоно-орієнтований підхід, завдяки якому можна було безпосередньо в лікаря дізнатись, що варто додати.

4.2.5 Інтегрований DICOM-переглядач

Найважливіша частина роботи зі знімками – це їх перегляд. Тому я інтегрував у веб-застосунок спеціальний DICOM-переглядач (рис. 4.23). Це дозволяє лікарям і консультантам дивитися DICOM-зображення прямо в

браузері, не встановлюючи ніяких додаткових програм на свій локальний пристрій.

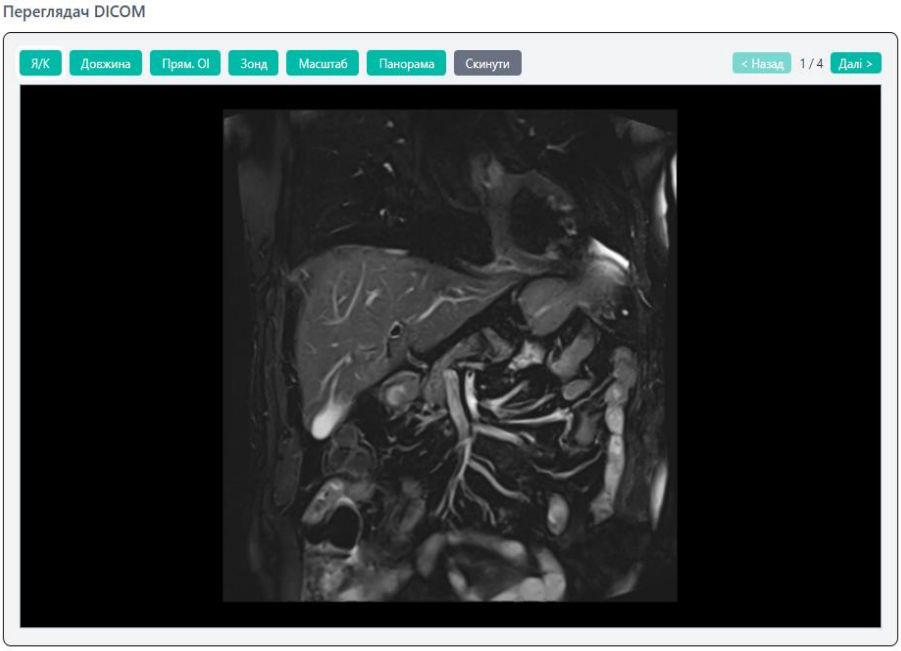

Рисунок 4.23 – DICOM-переглядач

Для реалізації була використана спеціалізована JavaScript-бібліотека Cornerstone.js, яка вміє відображати DICOM-дані та реагувати на дії користувача. Ознайомимось з основними функціями, які потрібні для аналізу вбудованого переглядача:

1) можна змінювати яскравість та контраст (рис. 4.24);

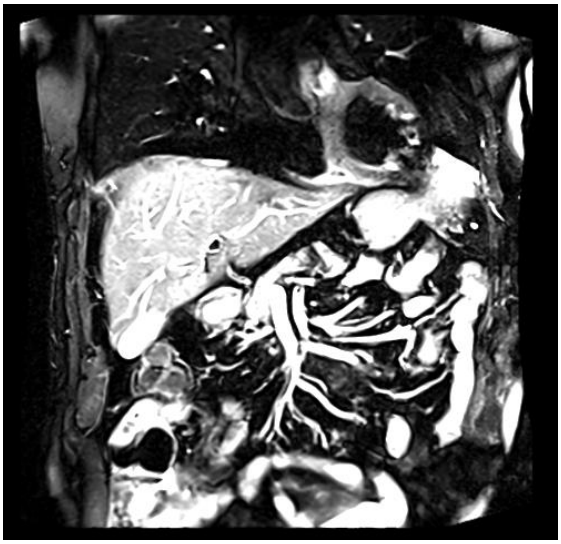

Рисунок 4.24 – підвищена контрастність та затемнення зображення

2) можна наближати та пересувати зображення (рис. 4.25);

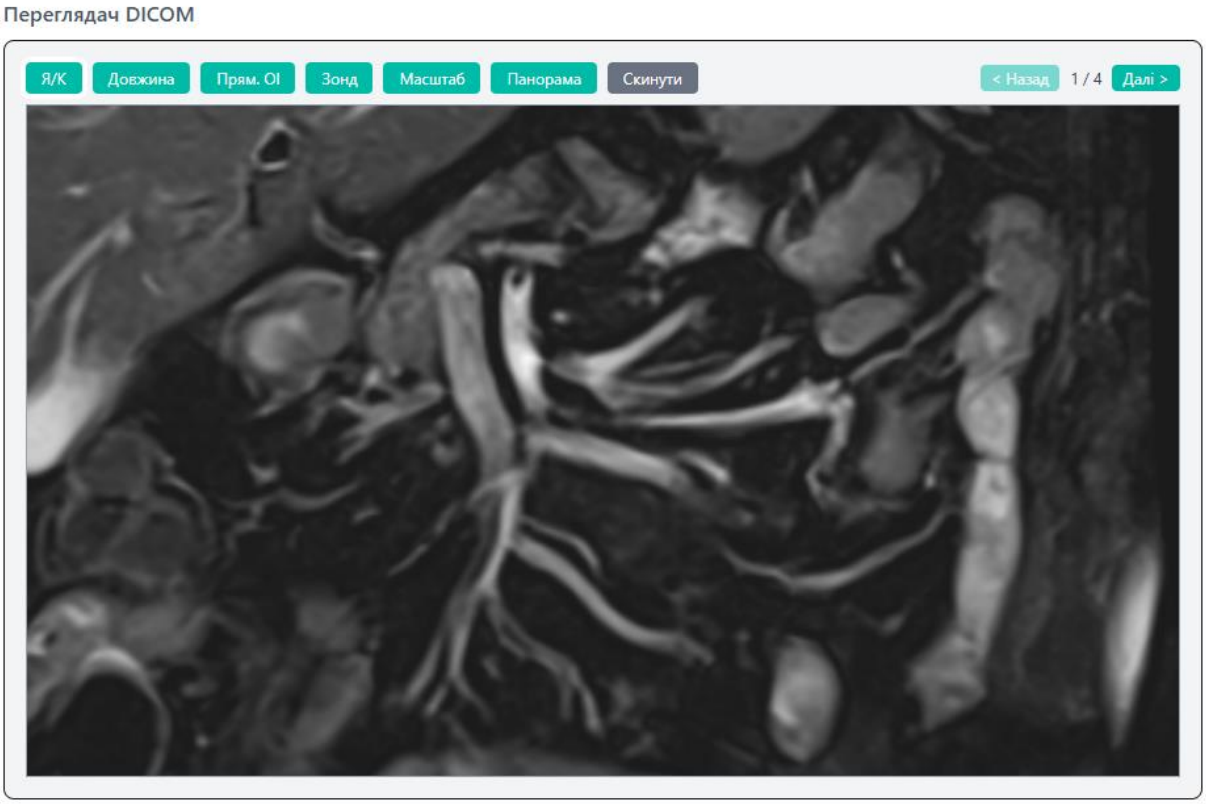

Рисунок 4.25 – збільшене та посунуте зображення

3) можна застосовувати різні вимірювальні інструменти (рис. 4.26),

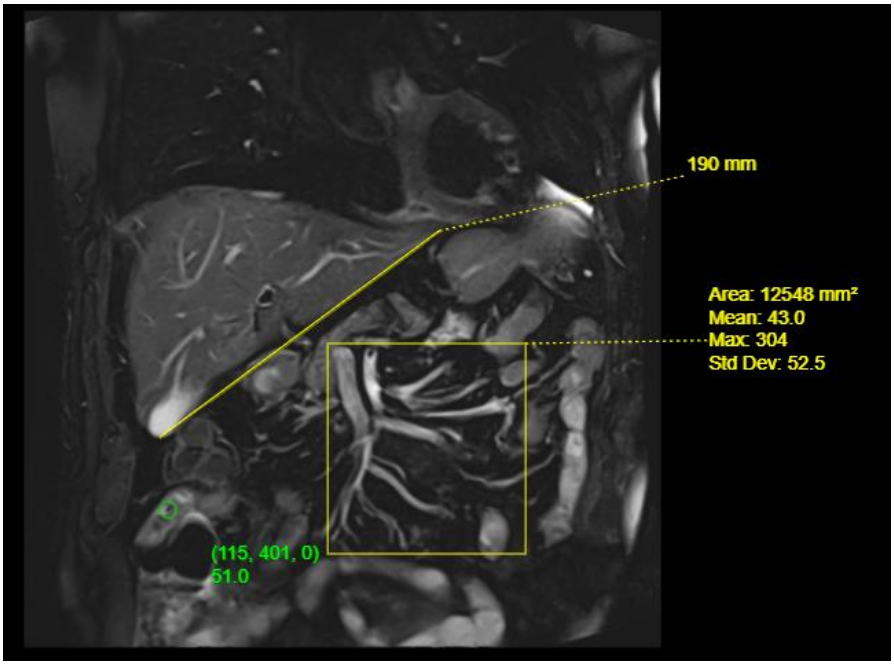

Рисунок 4.26 – вимірювальні інструменти

4) можна перегортати зрізи в серії (рис. 4.27).

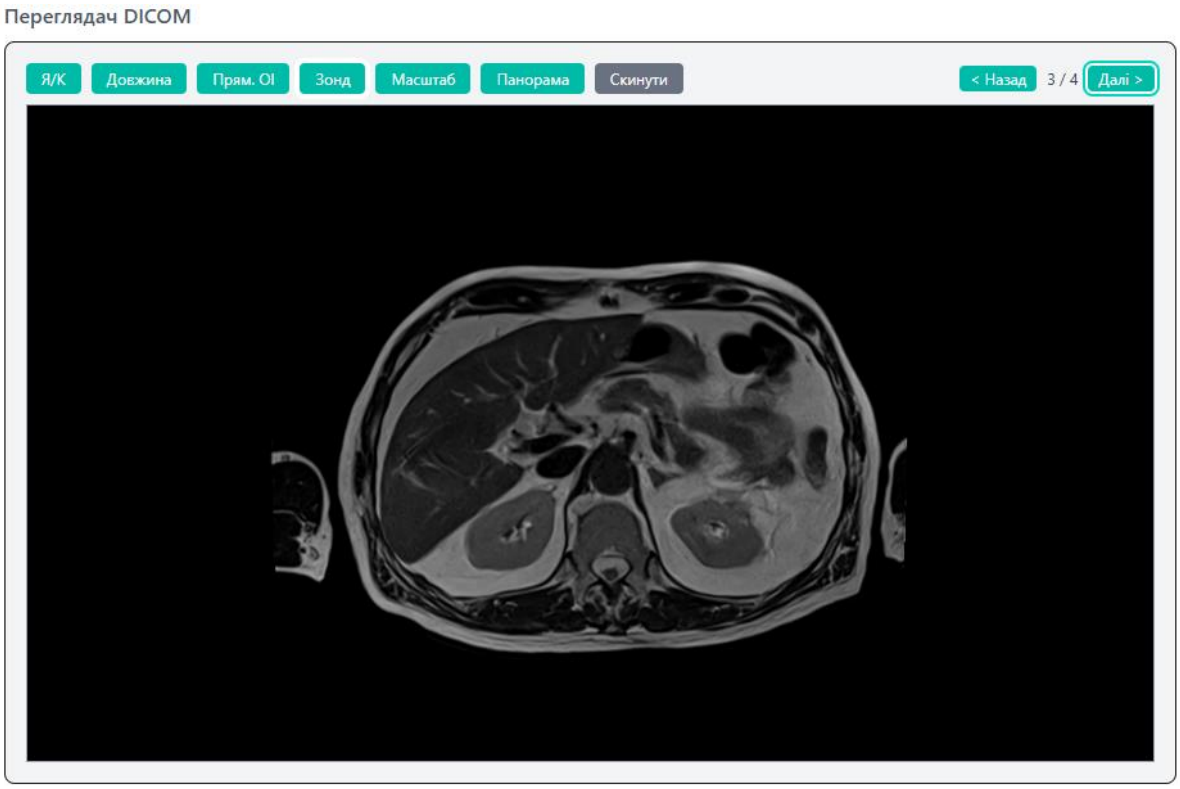

Рисунок 4.27 – перегляд зрізів серії

В комплексі це все створює умови для зручнішої роботи.

4.2.6 Модуль консультацій

Іноді лікарю потрібна думка іншого фахівця. Для таких випадків я зробив модуль консультацій. Лікуючий лікар може створити запит на консультацію для конкретного візиту, обравши потрібного консультанта зі списку (рис. 4.28).

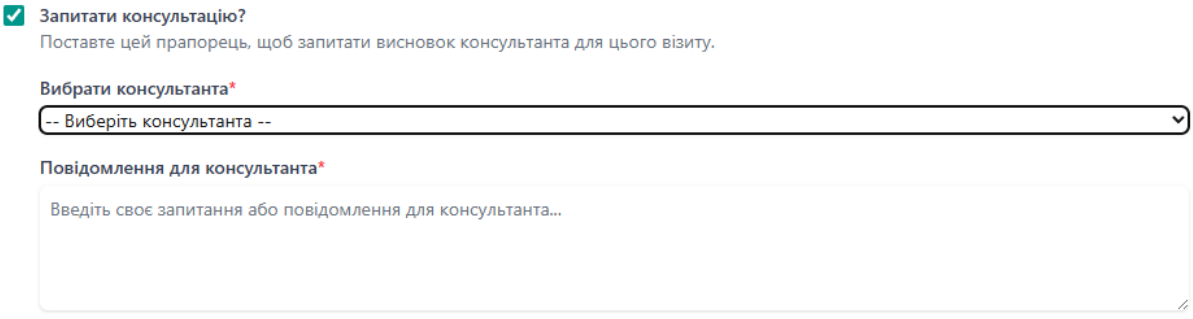

Рисунок 4.28 – Залучення консультанта до візиту

У запит автоматично потрапляє вся необхідна інформація про пацієнта, деталі візиту, результати аналізів та, звичайно, доступ до прикріплених DICOM-досліджень (рис. 4.29).

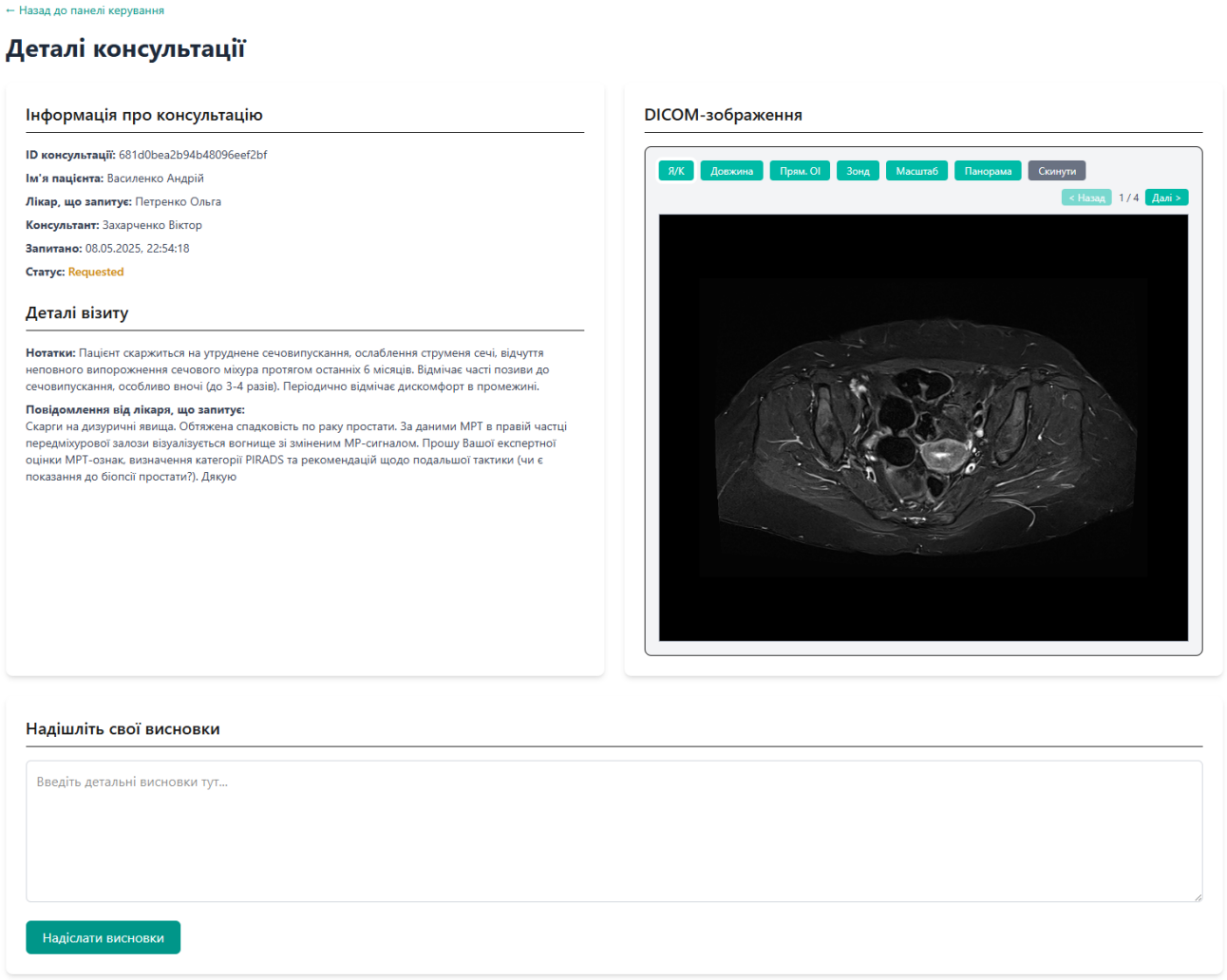

Рисунок 4.29 – Деталі консультації

Консультант бачить цей запит у своєму кабінеті (рис. 4.30). Він може переглянути всі дані, проаналізувати зображення за допомогою того ж вбудованого DICOM-переглядача і написати свій експертний висновок (рис. 4.31).

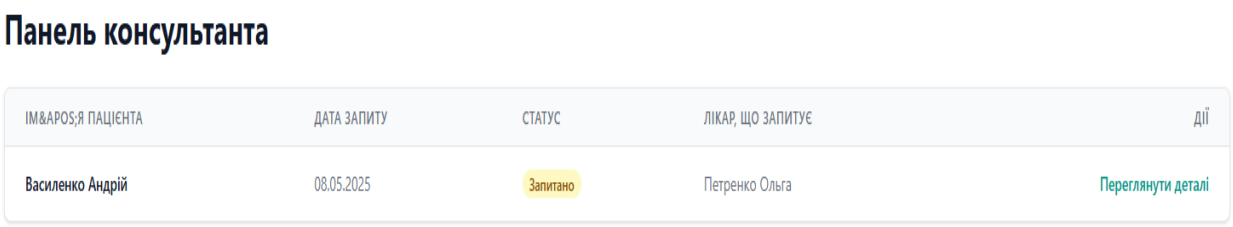

Рисунок 4.30 – Панель консультанта

Рисунок 4.31 – Висновки консультанта

Цей висновок потім стає доступним лікарю, який створив запит – коли він зайде на свою панель, вгорі з’явиться новий блок з доступними до завершення консультаціями (рис. 4.32).

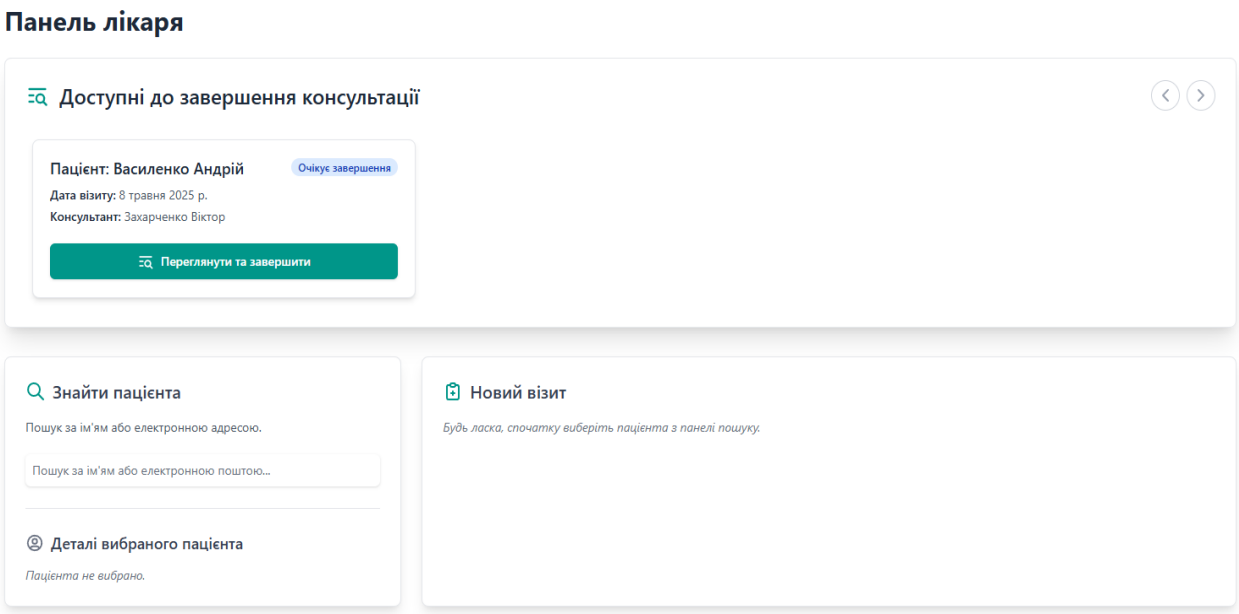

Рисунок 4.32 – Блок доступних до завершення консультацій

Лікар натискає «Переглянути та завершити», після чого для нього відкривається блок «Завершити візит» (рис. 4.33).

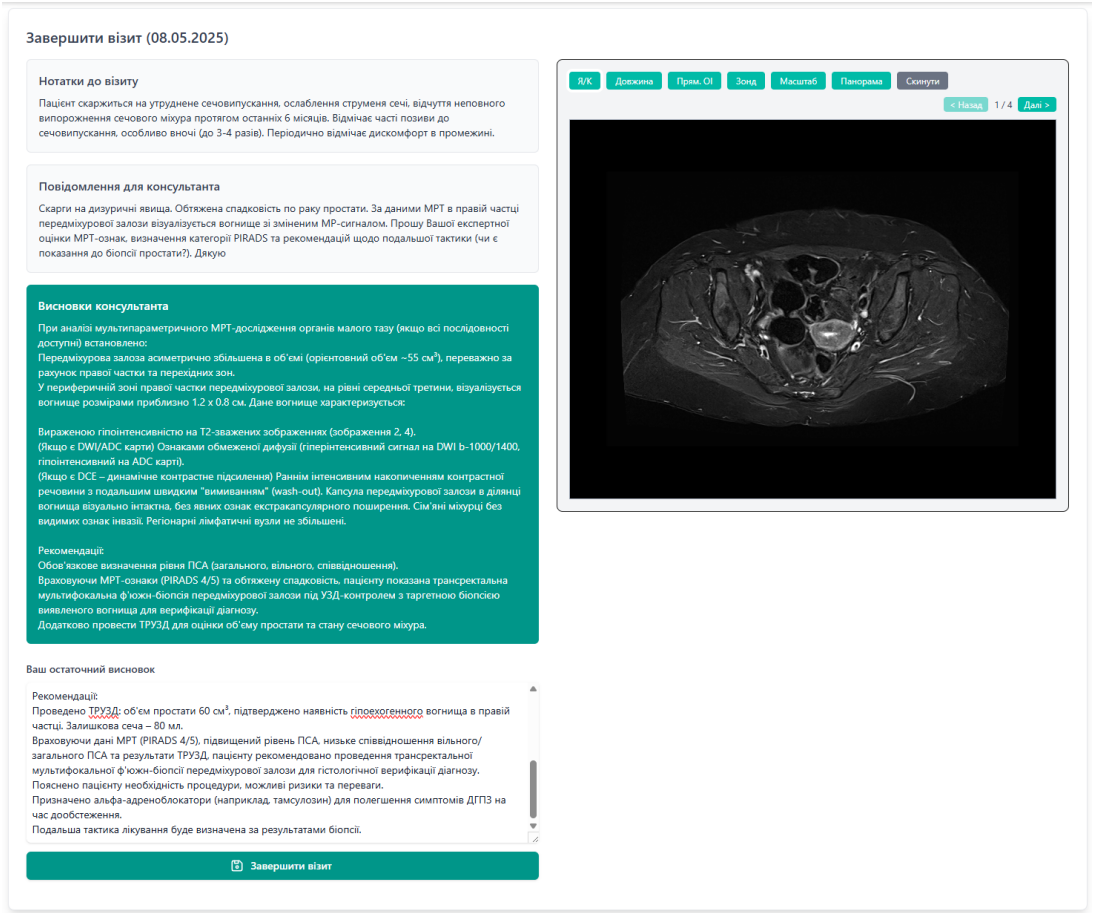

Рисунок 4.33 – Остаточний висновок лікаря після висновку консультанта

Загалом весь процес консультації допомагає приймати кращі рішення у складних клінічних ситуаціях.

4.2.7 Кабінет пацієнта

Після входу в систему пацієнт може переглянути список усіх своїх зареєстрованих візитів до лікарів у хронологічному порядку (рис. 4.34). Пацієнт бачить висновок лікаря, нотатки візиту, а також може переглянути медичні зображення.

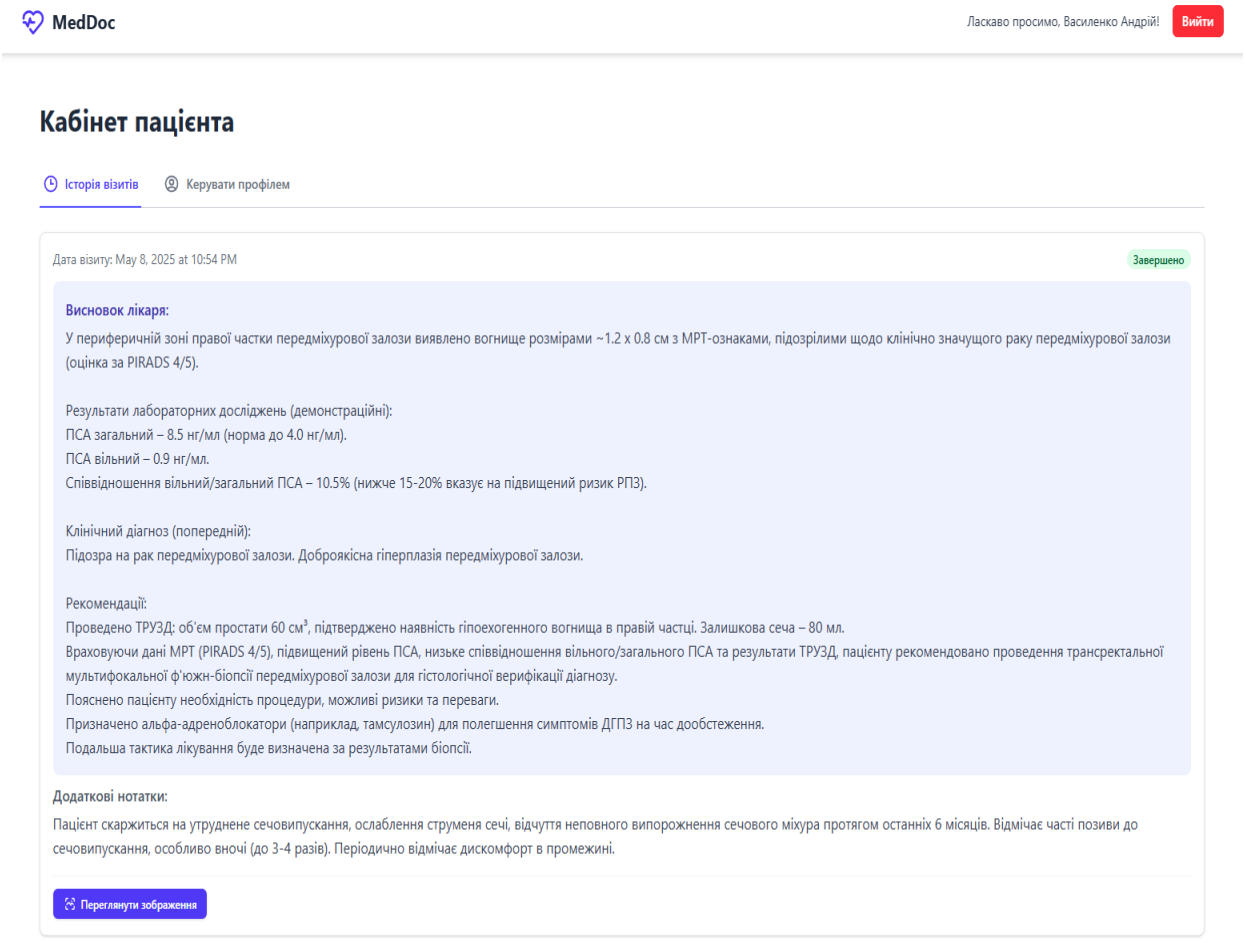

Рисунок 4.34 – Кабінет пацієнта

Також пацієнт може переглянути на вкладці «Керувати профілем» свої персональні дані і медичну інформацію, яку редагує лікар (рис. 4.35).

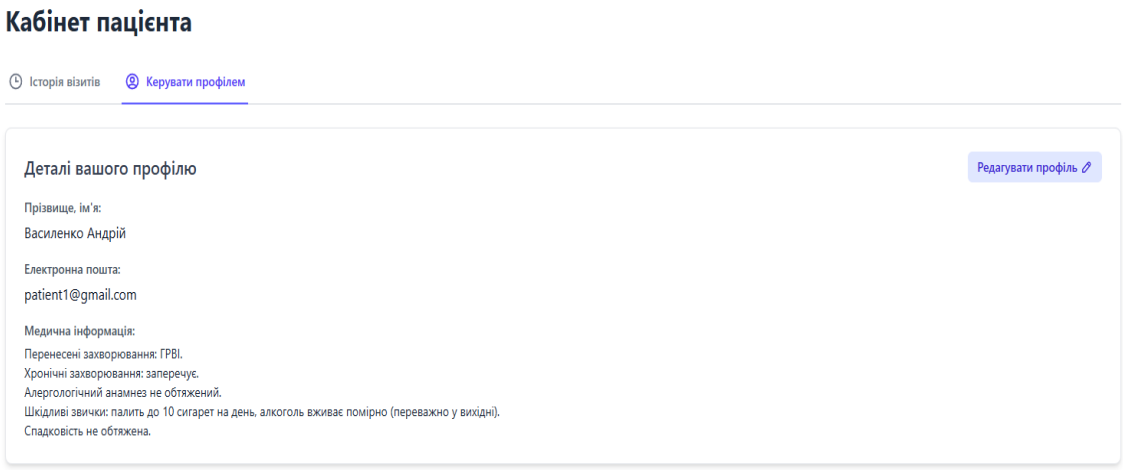

Рисунок 4.35 – Інформація пацієнта

Наразі єдине, що може змінити пацієнт – це прізвище та ім’я (рис. 4.36).

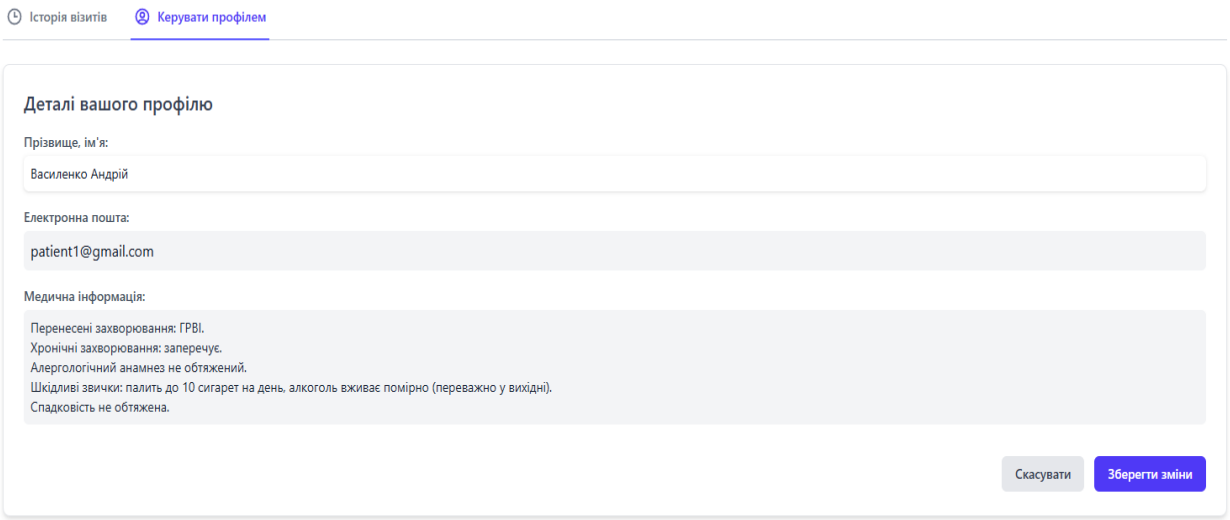

Рисунок 4.36 – Редагування особистих даних пацієнта

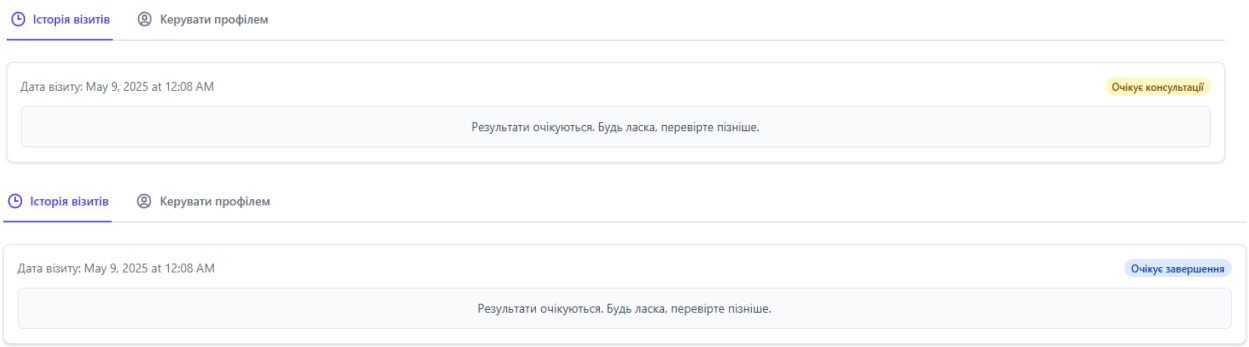

Рисунок 4.37 – Вид візиту, якщо було запитано консультацію

4.3 Демонстрація сценаріїв роботи користувачів

Щоб краще зрозуміти, як працює розроблений веб-застосунок на практиці, розглянемо кілька типових сценаріїв взаємодії для основних ролей користувачів: лікаря, консультанта та пацієнта.

4.3.1 Сценарій 1: робота лікаря

Уявімо робочий день лікаря, який використовує нашу систему.

1) Спочатку лікар заходить до системи, використовуючи свою форму входу (як показано на рис. 4.11).
2) Після входу він потрапляє на свою робочу панель. Тут він одразу може побачити, чи є консультації, які вже опрацьовані консультантами і чекають на його остаточний висновок (як у блоці на рис. 4.32). Якщо такі є, лікар, ймовірно, спочатку перегляне їх. Він натискає «Переглянути та завершити», читає висновок консультанта та заповнює остаточний висновок для візиту у відповідному блоці (рис. 4.33), після чого візит вважається завершеним.
3) Далі, якщо до лікаря прийшов пацієнт, лікар використовує функцію пошуку на своїй панелі (рис. 4.13), щоб знайти картку пацієнта за іменем, прізвищем чи поштою (рис. 4.14, 4.15).
4) Знайшовши потрібного пацієнта у списку, лікар обирає його, і система завантажує всю інформацію (рис. 4.16). Лікар може переглянути або оновити медичну інформацію пацієнта (рис. 4.17).
5) Для створення запису про поточний візит лікар знаходить потрібне діагностичне дослідження (яке, як ми припустили, вже надійшло з PACS-сервера Orthanc і відображається у списку доступних для пацієнта) та обирає його (рис. 4.18). Це відкриває комплексну форму візиту (рис. 4.19).
6) У цій формі лікар може детально проаналізувати DICOM-

зображення за допомогою вбудованого переглядача (рис. 4.23). Він використовує інструменти: змінює яскравість/контраст (рис. 4.24), масштабує та пересуває зображення (рис. 4.25), проводить вимірювання (рис. 4.26), переглядає різні зрізи серії (рис. 4.27).

7) Паралельно з аналізом зображень лікар заповнює текстові поля форми: записує свої нотатки щодо візиту, спостереження, попередній діагноз.
8) Якщо лікар вважає за потрібне отримати думку іншого фахівця, він прямо з цієї ж форми ініціює консультацію, натискаючи відповідну кнопку та обираючи консультанта зі списку (рис. 4.28).
9) Завершення роботи з візитом залежить від того, чи була ініційована консультація. Якщо консультація не потрібна, лікар пише свій остаточний висновок у відповідному полі форми (рис. 4.19) і зберігає/завершує візит. Якщо ж запит на консультацію було створено, лікар зберігає поточний стан візиту і повернеться до нього вже після отримання відповіді від консультанта (крок 2 цього сценарію).

4.3.2 Сценарій 2: робота консультанта

Консультант взаємодіє з системою, коли лікуючий лікар запросив його експертну думку.

1) Консультант входить до системи через той самий портал, що й лікар (рис. 4.11).
2) На його робочій панелі відображається список запитів на консультацію, які були призначені саме йому (рис. 4.30).
3) Він обирає зі списку консультацію, яку хоче опрацювати.
4) Система відображає сторінку з деталями консультації (рис. 4.29). Тут консультант бачить інформацію про пацієнта, дані візиту, нотатки лікуючого лікаря та його питання.
5) Консультант відкриває DICOM-зображення у вбудованому

переглядачі (рис. 4.23) та проводить їх аналіз, використовуючи доступні інструменти (рис. 4.24-4.27).

6) Після аналізу зображень та іншої інформації консультант формулює свій висновок та записує його у спеціальне поле (рис. 4.31).
7) Завершивши написання висновку, консультант натискає кнопку «Надіслати висновки». Після цього запит набуває статус «Висновки надіслано», а лікуючий лікар отримує сповіщення та бачить результат на своїй панелі (рис. 4.32).

4.3.3 Сценарій 3: доступ пацієнта

Пацієнт також може користуватися системою для доступу до своєї медичної історії.

1) Пацієнт входить до системи через портал для пацієнтів (рис. 4.7), використовуючи свої логін та пароль.
2) Після входу він потрапляє до свого особистого кабінету (рис. 4.34). Основна частина кабінету – це список його минулих візитів до лікарів, що були зареєстровані в системі.
3) Пацієнт може переглянути будь-який візит зі списку, фінальний висновок лікаря. Також він може бачити прикріплені до візиту зображення (рис. 4.34).

4.4 Розрахунок економічного ефекту

Під час переддипломної практики було проведено функціонально-вартісний аналіз (ФВА) проекту розробки веб-застосунку для керування медичною документацією та зображеннями у стандарті DICOM. Основною метою цього аналізу було визначення найбільш раціонального варіанту реалізації програмного продукту, що забезпечує оптимальне співвідношення між його технічними характеристиками та економічними витратами на створення.

В ході аналізу було декомпозовано головну функцію системи на ключові підфункції (реалізація Frontend, Backend, обробка DICOM, управління базою даних) та розглянуто альтернативні варіанти їх технічної реалізації з використанням сучасних технологій (Next.js, React, TypeScript, Node.js, MongoDB, PostgreSQL, спеціалізовані DICOM-бібліотеки та інструменти). За допомогою морфологічної карти та позитивно-негативної матриці було сформовано три потенційно життєздатні варіанти архітектури програмного продукту.

Для об'єктивної оцінки було визначено систему ключових параметрів якості: швидкість завантаження DICOM-зображень (X1), час розробки (X2), рівень безпеки даних (X3) та зручність користувацького інтерфейсу (X4). Вагові коефіцієнти цих параметрів були визначені шляхом експертного оцінювання (за участю 7 експертів), яке підтвердило високу узгодженість думок (W = 0.918) та виявило найвищу пріоритетність параметрів безпеки ($K_{в3}$ = 0.325) та зручності інтерфейсу ($K_{в4}$ = 0.268). На основі цих даних було розраховано коефіцієнти технічного рівня (K_K) для кожного варіанту, де найвищий показник продемонстрував Варіант 3 ($K_{К3}$ = 10.225).

Економічний аналіз включав детальну калькуляцію витрат для кожного варіанту, враховуючи скориговану трудомісткість (від 848 до 1083 людино-годин), витрати на оплату праці Senior розробника (з годинною ставкою ≈ 737.5 грн/год), відрахування на ЄСВ (22%), витрати на машинний час (з вартістю ≈ 20.88 грн/год), накладні витрати (67%) та витрати на матеріали (5% від основної ЗП). Розрахункова собівартість розробки ($С_{ПП}$) склала від 1,460,040 грн для Варіанту 1 до 1,874,591 грн для Варіанту 3.

Інтегральна оцінка ефективності варіантів проводилася за допомогою коефіцієнта техніко-економічного рівня. Згідно з розрахунками, найвище значення цього коефіцієнта має Варіант 2 ($К_{ТЕР2} \approx 5.513 * 10^{-6}$), що вказує на найкращий баланс між досягнутим рівнем якості та понесеними

витратами. Незважаючи на найвищу технічну якість Варіанту 3, його значно вища вартість робить його менш ефективним у співвідношенні «якість/ціна».

Таким чином, на підставі проведеного ФВА, Варіант 2 (Next.js frontend, окремий Node.js backend, JS DICOM бібліотеки, MongoDB) рекомендовано як оптимальний для реалізації даного програмного продукту. Детальний кошторис витрат для обраного варіанту показав, що повна вартість розробки, включаючи прибуток та ПДВ, становить близько 2,250,000 грн.

Проведений аналіз підтвердив цінність застосування методу ФВА для прийняття обгрунтованих технічних та економічних рішень на ранніх стадіях проектування складних програмних систем, дозволивши обрати найбільш збалансований варіант розробки веб-застосунку

Висновки до розділу 4

У даному розділі було детально висвітлено процес програмної реалізації веб-застосунку та продемонстровано його ключові функціональні характеристики. Здійснено огляд основних системних модулів: системи автентифікації з рольовим розмежуванням доступу, компонентів для управління даними пацієнтів та їхніх візитів, механізмів інтеграції з PACS-сервером Orthanc для роботи з DICOM-дослідженнями, а також інтегрованого веб-переглядача DICOM на основі Cornerstone.js, що включає базові інструменти аналізу зображень

Для наочної демонстрації практичного застосування реалізованого функціоналу було представлено типові сценарії взаємодії користувачів різних ролей (лікаря, консультанта та пацієнта). Ці сценарії охоплюють послідовність дій, починаючи від входу в систему та пошуку пацієнта, до створення запису про візит, аналізу DICOM-зображень, формування запиту на консультацію та її надання, а також перегляду медичної історії. Наведені приклади підтверджують відповідність розробленого програмного продукту функціональним вимогам щодо управління медичною документацією та

DICOM-даними, визначеним у рамках практичної роботи.

В роботі розглянуто розрахунок економічного ефекту. За темою дипломної роботи було проведено функціонально-вартісний аналіз (ФВА) проекту розробки веб-застосунку для керування медичною документацією та зображеннями у стандарті DICOM. Основною метою цього аналізу було визначення найбільш раціонального варіанту реалізації програмного продукту, що забезпечує оптимальне співвідношення між його технічними характеристиками та економічними витратами на створення. Детальні розрахунки надані в розділі звіту з переддипломної практики.

РОЗДІЛ 5
ЗАХИСТ ІНФОРМАЦІЇ

Робота з медичною інформацією вимагає особливої уваги до її захисту. Збереження конфіденційності даних пацієнтів, забезпечення їх цілісності та доступності для авторизованих користувачів – це ключові пріоритети для розробленого веб-застосунку, які розглянуті в п.2.1.2 даної роботи. Щоб досягти потрібного рівня безпеки, було застосовано як технічні рішення безпосередньо в програмі, так і організаційні правила для користувачів. Далі розглянемо ці заходи детальніше.

5.1 Технічні засоби захисту інформації в системі

Під час розробки застосунку було впроваджено кілька важливих технічних засобів для захисту даних.

Для контролю доступу до системи ми використали бібліотеку NextAuth.js. Вона відповідає за надійну перевірку користувачів при вході в систему за логіном та паролем. Важливо, що паролі користувачів ніколи не зберігаються у відкритому вигляді. При реєстрації чи зміні пароля система перетворює їх на хеш за допомогою криптографічного алгоритму. Це одностороння операція, тобто відновити оригінальний пароль з хешу практично неможливо, навіть якщо хтось отримає доступ до бази даних.

Після того, як користувач успішно увійшов, NextAuth.js керує його сесією. Для цього використовуються безпечні механізми, наприклад, JWT-токени, які зберігаються у спеціальних httpOnly cookie, або сесії, що зберігаються на сервері в базі даних. Це дозволяє системі «пам’ятати» користувача під час його роботи.

Фундаментальним принципом розмежування доступу є розподіл користувачів за ролями: адміністратор, лікар, консультант та пацієнт. Для

кожної ролі визначено специфічний набір прав, при цьому інформація про роль користувача зберігається в сесійних даних. При спробі доступу до певної сторінки або виконання дії система здійснює перевірку ролі користувача, дозволяючи або блокуючи операцію відповідно до його повноважень. Таким чином забезпечується контрольований доступ до функціоналу та даних.

Важливим заходом безпеки є шифрування даних у процесі їх передачі. Обмін інформацією між клієнтським браузером та веб-сервером здійснюється виключно за протоколом HTTPS, що гарантується наявністю SSL/TLS сертифіката на сервері. Використання протоколу HTTPS гарантує криптографічний захист усього обсягу переданої інформації, охоплюючи запити, відповіді сервера, облікові дані користувачів та медичні відомості. Такий підхід суттєво ускладнює можливість перехоплення та розшифрування цих даних під час їх передачі через інтернет-канали.

Щодо безпеки даних безпосередньо під час їх знаходження в базі MongoDB, крім застосованого хешування паролів, сам програмний продукт може не передбачати власних додаткових алгоритмів шифрування. Однак, варто зазначити, що сучасні хмарні платформи для баз даних, наприклад MongoDB Atlas, нерідко включають функціонал шифрування даних на рівні фізичного сховища. Це забезпечує шифрування інформації на дисках сервера, що слугує додатковим рівнем захисту від неавторизованого доступу до самого серверного обладнання.

Те, що ми використовуємо сучасні інструменти типу Next.js та React, вже трохи захищає від популярних веб-атак. Наприклад, React сам слідкує за тим, щоб в HTML не потрапило нічого зайвого, і це допомагає уникнути атак через скрипти (XSS). А коли працюємо з MongoDB через спеціальні інструменти як Mongoose, то вони перевіряють дані за нашими правилами, що ускладнює NoSQL-ін’єкції.

Також одна з важливих дій – це завжди перевіряти все, що користувач вводить у різні поля. Причому перевіряти і на його комп’ютері, щоб він

одразу бачив помилку, і обов’язково на нашому сервері, перш ніж ці дані кудись підуть чи збережуться. Так ми не дамо потрапити в систему неправильним чи шкідливим даним.

Також всі API-маршрути, які працюють з чутливими даними або виконують важливі дії, захищені. Перш ніж виконати запит, система перевіряє, чи користувач автентифікований і чи має він необхідну роль та права доступу.

5.2 Організаційно-правові аспекти та політика безпеки для користувачів

Технічних засобів недостатньо, якщо самі користувачі не дотримуються правил безпеки. Для персоналу, що працює з цим веб-застосунком, мають діяти чіткі вимоги.

По-перше, це конфіденційність облікових даних. Кожен співробітник повинен мати свій унікальний логін та надійний пароль. Передавати ці дані комусь іншому або використовувати чужий акаунт – неприпустимо. Паролі слід обирати складні та періодично змінювати. При будь-якій підозрі на компрометацію акаунту потрібно негайно інформувати системного адміністратора. Важливо завжди виходити з системи після завершення роботи, особливо на комп’ютерах спільного користування.

По-друге, це безпека безпосередньо на робочому місці. Комп’ютер, з якого здійснюється доступ до системи, має бути захищений паролем операційної системи. При відході від робочого місця екран потрібно блокувати. Не можна залишати без нагляду роздруковані медичні документи чи носії інформації.

По-третє, треба дуже серйозно ставитися до лікарської таємниці і правил роботи з інформацією. Заглядати в особисті дані пацієнтів та їхні медичні картки можна тільки тоді, коли це справді потрібно для роботи, і тільки якщо система дає на це право. Копіювати, скидати на флешку чи

пересилати цю інформацію через якісь незахищені канали – категорично заборонено. Якщо хтось розголосить дані пацієнта без дозволу – це серйозне порушення.

І, останнє, всі дії в системі мають відповідати чинному законодавству України, зокрема Закону «Про захист персональних даних» та іншим актам, що регулюють обіг медичної інформації.

Висновки до розділу 5

У даному розділі основну увагу приділено заходам, які було вжито для захисту інформації в межах розробленого веб-застосунку. Описано ключові технічні аспекти: систему автентифікації та авторизації, що базується на ролях та реалізована з допомогою NextAuth.js із застосуванням хешування паролів; шифрування даних під час їх передачі через HTTPS; можливості шифрування даних безпосередньо при зберіганні на рівні системи управління базами даних. Також у даному розділі представлено аналіз способів протидії поширеним загрозам веб-безпеки, серед яких фігурують атаки типу міжсайтового скриптингу та ін’єкції NoSQL-коду. Зменшення потенційних ризиків у цьому контексті забезпечується шляхом застосування вбудованих засобів захисту, наявних у використаних програмних фреймворках, та через обов’язкову перевірку всіх даних, що надаються користувачами системи. Важливим аспектом є акцентування на необхідності реалізації відповідних організаційних заходів та безумовного дотримання користувачами чинних регламентів інформаційної безпеки. Це й стосується забезпечення належного рівня конфіденційності облікових даних, підтримання безпеки на робочих місцях, а також суворого виконання етичних норм медичної практики та законодавчих вимог щодо обігу персональної інформації.

ЗАГАЛЬНІ ВИСНОВКИ

Результати роботи

У ході виконання дипломної роботи було успішно розроблено веб-застосунок, призначений для ефективного керування медичною документацією та зображеннями у стандарті DICOM. Ключовим результатом є створення функціонального програмного продукту, що використовує сучасний стек технологій, включаючи TypeScript, React та Next.js для розробки клієнтської частини, Next.js API маршрути для реалізації серверної логіки, систему управління базами даних MongoDB, бібліотеку NextAuth.js для забезпечення автентифікації, а також PACS-сервер Orthanc та бібліотеку Cornerstone.js для роботи з DICOM-зображеннями.

Розроблений застосунок надає комплексний функціонал, зокрема можливість керування даними пацієнтів та їхніми медичними візитами. Важливим аспектом є реалізована система автентифікації та розмежування прав доступу, що підтримує чотири різні ролі користувачів: адміністратора, відповідального за створення облікових записів медичного персоналу; лікаря, який веде пацієнтів, створює записи про візити та може запитувати консультації; консультанта, що надає експертні висновки; та пацієнта, який має доступ до перегляду власної медичної історії та зображень. Суттєвим досягненням є інтеграція DICOM-переглядача безпосередньо у веб-інтерфейс, що дозволяє авторизованим користувачам переглядати DICOM-зображення та використовувати базові інструменти для їх аналізу, такі як зміна яскравості та контрасту, масштабування, панорамування, вимірювання та навігація по серіях зображень, без необхідності встановлення стороннього програмного забезпечення. Забезпечено надійну взаємодію з PACS-сервером Orthanc для процесів завантаження, зберігання та отримання DICOM-досліджень. Також реалізовано модуль консультацій, який полегшує отримання лікарями експертної думки від інших фахівців.

Значну увагу було приділено питанням безпеки: впроваджено хешування паролів, забезпечено передачу даних через захищений протокол HTTPS, реалізовано захист API-маршрутів та розмежування доступу на основі ролей. Працездатність системи та її відповідність поставленим вимогам підтверджено шляхом тестування основних функціональних модулів та демонстрації типових сценаріїв роботи користувачів.

Висновки

У ході виконання дипломної роботи було успішно вирішено актуальну проблему сучасної медицини, пов’язану з розрізненістю медичної документації та діагностичних зображень. Головна мета проєкту – створення та програмна реалізація інтегрованого веб-застосунку для оптимізації роботи з клінічними даними – була повністю досягнута. Виконання всіх ключових завдань, визначених на етапах теоретичного аналізу, проєктування та практичної реалізації, включно з розробкою функціональних модулів, інтеграцією з PACS-системою та забезпеченням безпеки даних, дозволило створити комплексне та завершене програмне рішення, що відповідає сучасним вимогам до медичних інформаційних систем.

Практична значущість розробленого веб-застосунку полягає у його здатності суттєво оптимізувати робочі процеси в медичних установах. Надаючи єдину точку доступу до розрізнених раніше даних пацієнта, включаючи історію візитів, результати консультацій та діагностичні DICOM-зображення з можливістю їх зручного перегляду, система сприяє швидшому та більш обґрунтованому прийняттю клінічних рішень. Це, у свою чергу, позитивно впливає на своєчасність постановки діагнозу та загальну якість медичної допомоги.

Перспективи подальшого розвитку та вдосконалення системи є досить широкими. Пріоритетними напрямками є реалізація передачі консультанту DICOM-зображень з поточними лікарськими вимірюваннями та анотаціями для оптимізації спільної роботи фахівців; суттєве розширення

інструментарію вбудованого DICOM-переглядача, зокрема додавання функцій для роботи з тривимірними реконструкціями, розширеного анотування медичних зображень та можливості динамічного порівняння різних діагностичних досліджень одного пацієнта. Окрім цього, актуальним є дослідження та подальша інтеграція елементів штучного інтелекту для автоматизованої підтримки прийняття клінічних рішень та інтелектуального аналізу зображень. Успішна реалізація цих напрямків дозволить ще більше підвищити цінність розробленого програмного продукту та його конкурентоспроможність на ринку медичних інформаційних технологій, підтверджуючи вагомий внесок даної роботи у цифровізацію сфери охорони здоров’я.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Al Khatib, I., Shamayleh, A., Ndiaye, M. Healthcare and the Internet of Medical Things: Applications, Trends, Key Challenges, and Proposed Resolutions [Електронний ресурс] // Informatics. – 2024. – Vol. 11, № 3. – Article ID: 47. – Режим доступу до ресурсу: https://www.mdpi.com/2227-9709/11/3/47
2. Alharbe, N. Ontological Augmentation and Analytical Paradigms for Elevating Security in Healthcare Web Applications [Електронний ресурс] // Scalable Computing: Practice and Experience. – 2024. – Vol. 25, № 1. – P. 157–165. – Режим доступу до ресурсу: https://www.researchgate.net/publication/377173598_Ontological_Augmentation_and_Analytical_Paradigms_for_Elevating_Security_in_Healthcare_Web_Applications
3. Andriole, K. P. Picture archiving and communication systems: past, present, and future [Електронний ресурс] // Journal of Medical Imaging. – 2023. – Vol. 10, № 6. – Article ID: 061405. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC10754358/
4. Balykina, A. Охорона здоров’я у хмарах. Хмарні технології у медицині [Електронний ресурс] / A. Balykina // SIM-Networks. – 2024. – Режим доступу до ресурсу: https://www.sim-networks.com/ukr/blog/cloud-technologies-for-medicine
5. Bhatt, D., Parekh, K., Minat, M., Patel, B. ReactJS: A Comprehensive Analysis of its Features, Performance, and Suitability for Modern Web Development [Електронний ресурс] // International Journal of Scientific Research in Engineering and Management (IJSREM). – 2023. – Vol. 7, № 9. – P. 1–6. – Режим доступу до ресурсу: https://ijsrem.com/download/reactjs-a-comprehensive-analysis-of-its-features-performance-and-suitability-for-modern-web-development/

6. Bidgood, W. D. Jr., Horii, S. C., Prior, F. W., Van Syckle, D. E. Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging [Електронний ресурс] // Journal of the American Medical Informatics Association. – 1997. – Vol. 4, № 3. – P. 199–212. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC61235/
7. BlueCoding. Examining TypeScript's Advantages for Big JavaScript Projects [Електронний ресурс] // BlueCoding. – 2024. – Режим доступу до ресурсу: https://www.bluecoding.com/post/exploring-the-benefits-of-typescript-for-large-scale-javascript-projects
8. Bria, A. DICOM vs PACS – Understanding the Benefits and Differences in Medical Imaging [Електронний ресурс] / A. Bria ; мед. рец. A. Costache // Medical Imaging Technology Blog. – 2025. – Режим доступу до ресурсу: https://blog.medicai.io/en/dicom-vs-pacs-exploring-the-differences/
9. Budiman, M. A. A Comprehensive Introduction to Tailwind CSS [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@alifm2101/a-comprehensive-introduction-to-tailwind-css-36bc9cb81a1c
10. Calisto, F. M. Using CornerstoneJS and Orthanc to Support Deep Learning Projects [Електронний ресурс] // Medium. – 2019. – Режим доступу до ресурсу: https://medium.com/oppr/using-cornerstonejs-and-orthanc-to-support-deep-learning-projects-c9675819c33a
11. Chane, M. What are the best Free DICOM Viewer to Open your Medical images? [Електронний ресурс] // M. Chane. – 2019. – Режим доступу до ресурсу: https://www.imaios.com/en/resources/blog/5-best-dicom-viewer
12. Clarke, C. DICOM / PACS viewing software: The best free software solutions for DICOM images [Електронний ресурс] // Radiology Cafe. – 2023. – Режим доступу до ресурсу: https://www.radiologycafe.com/radiology-trainees/dicom-viewers/

13. CodeWalnut. A Complete Guide On Benefits Of Using Next.js [Електронний ресурс] // CodeWalnut. – 2024. – Режим доступу до ресурсу: https://www.codewalnut.com/learn/benefits-of-using-next-js
14. Cornerstone.js – офіційна документація [Електронний ресурс] // Cornerstone.js. – [б. д.]. – Режим доступу до ресурсу: https://www.cornerstonejs.org/docs/getting-started/overview/
15. Coventry, P. A., Bower, P., Blakemore, A., Baker, L., Hann, M., Paisley, A., Renwick, C., Li, J., Ugajin, A., Gibson, M. Comparison of active treatments for impaired glucose regulation: a Salford Royal Foundation Trust and Hitachi collaboration (CATFISH): study protocol for a randomized controlled trial [Електронний ресурс] // Trials. – 2016. – Vol. 17, № 1. – Article ID: 424. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC5000461/
16. Denk Giebel, G., Speckemeier, C., Abels, C., Plescher, F., Börchers, K., Wasem, J., Blase, N., Neusser, S. Problems and Barriers Related to the Use of Digital Health Applications: Scoping Review [Електронний ресурс] // Journal of Medical Internet Research. – 2023. – Vol. 25. – Режим доступу до ресурсу: https://www.jmir.org/2023/1/e43808/
17. Dixit, N., Shrivastava, V., Pandey, A., Sharma, R. Revolutionizing Web Design with Tailwind CSS: A Comprehensive Exploration [Електронний ресурс] // International Journal of Research Publication and Reviews. – 2024. – Vol. 5, Issue 5. – P. 4198–4204. – Режим доступу до ресурсу: https://ijrpr.com/uploads/V5ISSUE5/IJRPR27677.pdf
18. Doran S.J., Al Sa'd M., Petts J.A., et al. Integrating the OHIF Viewer into XNAT: Achievements, Challenges and Prospects for Quantitative Imaging Studies [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC8875191/
19. Emmanni, P. S. The Role of TypeScript in Enhancing Development with Modern JavaScript Frameworks [Електронний ресурс] // International Journal of Science and Research (IJSR). – 2021. – Vol. 10, № 1. – Режим

доступу до ресурсу: https://www.researchgate.net/publication/380361058_The_Role_of_TypeScript_in_Enhancing_Development_with_Modern_JavaScript_Frameworks

20. Gamela, A., Rebelo, G. Next.js vs React: What Are the Differences? [Електронний ресурс] // Imaginary Cloud. – 2024. – Режим доступу до ресурсу: https://www.imaginarycloud.com/blog/next-js-vs-react
21. Golubev, A., Bogatencov, P., Secrieru, G. DICOM data processing optimization in medical information systems [Електронний ресурс] // Scalable Computing: Practice and Experience. – 2018. – Vol. 19, № 2. – P. 189–201. – Режим доступу до ресурсу: https://scispace.com/pdf/dicom-data-processing-optimization-in-medical-information-10c6gdw8sz.pdf
22. Google Cloud. Ambra Health: Accelerating healthcare research with anonymized medical imaging data [Електронний ресурс]. – [б. д.]. – Режим доступу до ресурсу: https://cloud.google.com/customers/ambra-health
23. Gopalakrishnan, R. Tailwind CSS vs Bootstrap: Tactical Web Development Frameworks [Електронний ресурс] // Spritle. – 2024. – Режим доступу до ресурсу: https://www.spritle.com/blog/tailwind-vs-bootstrap-tactical-web-development-frameworks/
24. Győrödi, C., Dumşe-Burescu, D. V., Zmaranda, D. R., Gyorodi, R. A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management [Електронний ресурс] // Big Data and Cognitive Computing (BDCC). – 2022. – Vol. 6, Issue 2. – P. 49. – Режим доступу до ресурсу: https://www.researchgate.net/publication/360401893_A_Comparative_Study_of_MongoDB_and_Document-Based_MySQL_for_Big_Data_Application_Data_Management
25. Haidar, A., Aly, F., Holloway, L. PDCP: A Set of Tools for Extracting, Transforming, and Loading Radiotherapy Data from the Orthanc Research PACS [Електронний ресурс] // Software. – 2022. – Vol. 1, № 2. – P. 215–

222. – Режим доступу до ресурсу: https://www.mdpi.com/2674-113X/1/2/9

26. HELSI – офіційний сайт [Електронний ресурс] // HELSI. – [б. д.]. – Режим доступу до ресурсу: https://helsi.me/about
27. Holivets, S. Advantages and Benefits of React JS: The Essential Tool for Modern Developers [Електронний ресурс] // MaybeWorks. – 2025. – Режим доступу до ресурсу: https://maybe.works/blogs/react-js-advantages
28. Jindal, A., Shrivastava, V., Pandey, A. Benefits of Use of MongoDB as Database: A Comprehensive Study [Електронний ресурс] // International Journal of Research Publication and Reviews. – 2024. – Vol. 5, Issue 4. – P. 2559–2562. – Режим доступу до ресурсу: https://ijrpr.com/uploads/V5ISSUE4/IJRPR24918.pdf
29. Jodogne, S. The Orthanc Ecosystem for Medical Imaging [Електронний ресурс] // Journal of Digital Imaging. – 2018. – Vol. 31, № 2. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC5959835/
30. Josefsson, U., Berg, M., Koinberg, I., Hellström, A.-L., Jenholt Nolbris, M., Ranerup, A., Sparud Lundin, C., Skärsäter, I. Person-centred web-based support – development through a Swedish multi-case study [Електронний ресурс] // BMC Medical Informatics and Decision Making. – 2013. – Vol. 13. – Article ID: 119. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC4015484/
31. Mokhtari M.E.A. Easy Peasy Medical Imaging with OHIF: Create a Free DICOM Viewer [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://pycad.medium.com/easy-peasy-medical-imaging-with-ohif-7596b3d8244d
32. MongoDB – офіційна документація [Електронний ресурс] // MongoDB. – [б. д.]. – Режим доступу до ресурсу: https://www.mongodb.com/docs/atlas/ https://www.mongodb.com/solutions/industries/healthcare

33. Natalia. Медичний застосунок: фінансові та репутаційні перспективи для сфери охорони здоров'я [Електронний ресурс] / Natalia // EDGELAB. – 2023. – Режим доступу до ресурсу: https://edgelab.com.ua/blog/medychnyj-zastosunok-finansovi-ta-reputaczijni-perspektyvy-dlya-sfery-ohorony-zdorovya/
34. National Electrical Manufacturers Association. DICOM Standard [Електронний ресурс]. – Режим доступу до ресурсу: https://www.dicomstandard.org/
35. Next.js – офіційна документація [Електронний ресурс] // Next.js. – [б. д.]. – Режим доступу до ресурсу: https://nextjs.org/docs
36. NextAuth.js – офіційна документація [Електронний ресурс] // NextAuth.js. – [б. д.]. – Режим доступу до ресурсу: https://next-auth.js.org/getting-started/introduction
37. OHIF Viewer – офіційна документація [Електронний ресурс] // OHIF Viewer. – [б. д.]. – Режим доступу до ресурсу: https://docs.ohif.org/
38. Orthanc – офіційна документація [Електронний ресурс] // Orthanc. – [б. д.]. – Режим доступу до ресурсу: https://www.orthanc-server.com/static.php?page=documentation
39. Padmanabhan A. OHIF Viewer [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://devopedia.org/ohif-viewer
40. Parmar, A. Why Use React in 2025 for Your Business App Development? [Електронний ресурс] // Glorywebs. – 2025. – Режим доступу до ресурсу: https://www.glorywebs.com/blog/why-use-reactjs
41. Patel, V. Analyzing the Impact of Next.JS on Site Performance and SEO [Електронний ресурс] // International Journal of Computer Applications Technology and Research. – 2023. – Vol. 12, № 10. – P. 24–27. – Режим доступу до ресурсу: https://doi.org/10.7753/IJCATR1210.1004
42. Pereira, H., Romero, L., Faria, P. M. Web-Based DICOM Viewers: A Survey and a Performance Classification [Електронний ресурс] // Journal of Imaging Informatics in Medicine. – 2024. – Режим доступу до ресурсу:

https://www.springermedizin.de/web-based-dicom-viewers-a-survey-and-a-performance-classificatio/50060790

43. PR Newswire. Ambra Health surpasses 10 billion images under management [Електронний ресурс] // PR Newswire. – 2021. – Режим доступу до ресурсу: https://www.prnewswire.com/news-releases/ambra-health-surpasses-10-billion-images-under-management-301233074.html
44. PYCAD. Top 7 DICOM Viewer Software Solutions [Електронний ресурс] // PYCAD. – [б. д.]. – Режим доступу до ресурсу: https://pycad.co/dicom-viewer-software/
45. Rathore, M., Bagui, S. S. MongoDB: Meeting the Dynamic Needs of Modern Applications [Електронний ресурс] // Encyclopedia. – 2024. – Vol. 4, Issue 4. – P. 1433–1453. – Режим доступу до ресурсу: https://www.mdpi.com/2673-8392/4/4/93
46. React – офіційна документація [Електронний ресурс] // React. – [б. д.]. – Режим доступу до ресурсу: https://react.dev/reference/react
47. Robinson, E. Why Use TypeScript in 2025? [Електронний ресурс] // Caisy. – 2024. – Режим доступу до ресурсу: https://caisy.io/blog/why-use-typescript
48. Saleem M. A Complete Guide to Next.js Authentication [Електронний ресурс] // ButterCMS Blog. – 2025. – Режим доступу до ресурсу: https://buttercms.com/blog/nextjs-authentication-tutorial/
49. Singh, L. Designing a Modern Library Website using Tailwind CSS [Електронний ресурс] // International Journal of Information Movement. – 2024. – Vol. 9, Issue I. – P. 20–29. – Режим доступу до ресурсу: https://www.ijim.in/files/2024/May%202024/Vol%209%20Issue%20I%2020-29%20Paper%204%20Lakhpat%20Singh.pdf?_t=1721199937
50. Srivastava, S., Shukla, H., Landge, N., Srivastava, A., Jindal, D. A Comprehensive Review of Next.js Technology: Advancements, Features, and Applications [Електронний ресурс] // SSRN. – 2024. – Режим

доступу до ресурсу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4831070

51. TypeScript – офіційна документація [Електронний ресурс] // TypeScript. – [б. д.]. – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/
52. Wadali, J. S., Sood, S. P., Kaushish, R., Syed-Abdul, S., Khosla, P. K., Bhatia, M. Evaluation of Free, Open-source, Web-based DICOM Viewers for the Indian National Telemedicine Service (eSanjeevani) [Електронний ресурс] // Journal of Digital Imaging. – 2020. – Vol. 33, № 6. – P. 1499–1513. – Режим доступу до ресурсу: https://pmc.ncbi.nlm.nih.gov/articles/PMC7728946/
53. Zemskov, M. Reliability of the Type System in TypeScript in Software Development [Електронний ресурс] // Asian Journal of Research in Computer Science. – 2025. – Vol. 18, № 2. – P. 50–59. – Режим доступу до ресурсу: https://doi.org/10.9734/ajrcos/2025/v18i2561
54. Дія.Бізнес. Великий гід по Helsi: як зареєструватись і чим може допомогти [Електронний ресурс] // Дія.Бізнес. – [б. д.]. – Режим доступу доступу до ресурсу: https://bf.diia.gov.ua/articles/velikij-gid-po-helsi-yak-zareyestruvatis-i-chim-mozhe-dopomogti
55. Коломієць, Р. О. Отримання, перетворення та обробка біосигналів: Навчальний посібник для студентів спеціальності 163 «Біомедична інженерія» [Електронний ресурс] // Р. О. Коломієць, Т. М. Нікітчук, Д. С. Морозов. – Житомир: Державний університет «Житомирська політехніка», 2024. – 294 с. – Режим доступу до ресурсу: https://learn.ztu.edu.ua/pluginfile.php/359723/mod_resource/content/0/%D0%9E%D0%9F%D1%82%D0%B0%D0%9E%D0%91%D0%A1.pdf