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

Комп'ютерно-інформаційних систем і програмної інженерії

(повна назва факультету)

Програмної інженерії

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

(назва освітнього ступеня)

на тему: Проєктування та розробка програмного забезпечення Веб-орієнтованої системи для комп'ютерного зору для Інтерактивного аналізу рентгенівських знімків легень

Виконав(ла): студент(ка) IV курсу, групи СП-43

спеціальності 121 – Інженерія програмного забезпечення

(шифр і назва спеціальності)

_____ Юрчишина В.А

(підпис) (прізвище та ініціали)

Керівник _____ Петрик М.Р

(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М

(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р

(підпис) (прізвище та ініціали)

Рецензент _____ _____

(підпис) (прізвище та ініціали)

Тернопіль
2026

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

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

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)

« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Юрчишиній Вікторії Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Проєктування та розробка програмного забезпечення веб-орієнтованої комп'ютерного зору для інтерактивного аналізу рентгенівських знімків легень

Керівник роботи д.ф-м.н, професор Петрик Михайло Романович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 06 » квітня 2026 року № 4/9-171

2. Термін подання студентом завершеної роботи 22.06.2026

3. Вихідні дані до роботи обраний технологічний стек (React, FastAPI, YOLOv8), відкриті медичні датасети рентгенограм грудної клітки, наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
Розділ 1 Аналіз вимог до програмної системи
Розділ 2 Проєктування та розробка програмної системи
Розділ 3 Тестування, впровадження та підтримка
Розділ 4 Безпека життєдіяльності, основи охорони праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи, актуальність дослідження та мета проєкту. 2. Архітектура медичної вебплатформи MedVision та схема клієнт-серверної взаємодії (REST API).
3.Структура і схема бази даних вебплатформи. 4.Архітектура згорткової нейронної мережі YOLOv8 для сегментації та детекції патологій. 5.Принцип роботи методів пояснювального штучного інтелекту (XAI) на основі алгоритму Grad-CAM. 6. Діаграма прецедентів (Use Case) та структура користувацьких інтерфейсів (React). 7.Результати навчання моделей, графіки лосів (Loss) та метрик точності (mAP, IoU). 8.Демонстрація інтерфейсу користувача 9. Тестування. 10. Загальні висновки виконання роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	к.т.н., доц. Мариненко С.Ю		
Нормоконтроль	Стоянов Ю.М. к.т.н., доц. каф. ПІ		

7. Дата видачі завдання 6 квітня 2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Збір інформації за темою дослідження, патентний пошук та аналіз сучасних PACS-систем та методів інтеграції комп'ютерного зору в медичну діагностику. Підготовка вступу та першого розділу (Розділ 1 Аналіз вимог до програмної системи).		
2.	Обґрунтування вибору технологічного стеку розробки. Проєктування клієнт-серверної архітектури вебплатформи MedVision та схеми бази даних. Оформлення другого розділу пояснювальної записки (Розділ 2 Проєктування та розробка програмної системи).		
3.	Навчання згорткової нейромережі YOLOv8 на медичних датасетах, інтеграція методів пояснювального ШІ. Розробка бекенду на FastAPI та веб-інтерфейсу на React		
4.	Проведення комплексного модульного та інтеграційного тестування системи, аналіз метрик точності моделей (Розділ 3 Тестування, впровадження та підтримка).		
5.	Розробка рішень з безпеки життєдіяльності та охорони праці (Розділ 4).		
6.	Формулювання загальних висновків, оформлення списку використаних джерел та додатків.		
7.	Перевірка роботи на плагіат, отримання відгуку керівника і подача диплому на нормоконтроль.		
8.	Попередній захист кваліфікаційної роботи бакалавра		
9.	Захист кваліфікаційної роботи бакалавра		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра виконана Юрчишиною Вікторією Андріївною, студенткою групи СП-43 Тернопільського національного технічного університету імені Івана Пулюя, на тему «Проєктування та розробка програмного забезпечення веб-орієнтованої системи комп'ютерного зору для інтерактивного аналізу рентгенівських знімків легень». Робота має обсяг 70 сторінок, включає 11 рисунків, 4 таблиці, 3 додатків та бібліографію з 22 джерел.

Метою роботи є розробка програмного забезпечення для веб-орієнтованої системи для автоматизованого аналізу рентгенівських знімків легень із використанням методів комп'ютерного зору та штучного інтелекту. Система призначена для підвищення ефективності первинної діагностики шляхом автоматичного виявлення патологій на рентгенівських знімках, візуалізації зон ураження та надання користувачу довідкової медичної інформації.

У роботі розроблено програмну систему MedVision, архітектура якої включає клієнтську частину, серверний застосунок, базу даних та модуль штучного інтелекту для аналізу медичних зображень. Реалізовано механізми реєстрації та авторизації користувачів, завантаження рентгенівських знімків, автоматичного аналізу зображень, локалізації патологій, формування медичної довідки та збереження історії виконаних досліджень.

Для реалізації програмної системи використано технології React, ASP.NET Core Web API, Entity Framework Core, Microsoft SQL Server, Python та FastAPI. Для виявлення патологій на рентгенівських знімках застосовано модель комп'ютерного зору YOLOv8, яка дозволяє здійснювати локалізацію патологічних змін та визначати їх тип із відповідним рівнем достовірності.

Ключові слова роботи: комп'ютерний зір, штучний інтелект, аналіз медичних зображень, рентгенівські знімки легень, YOLOv8, веб-орієнтована система, React, ASP.NET Core, медична діагностика, MedVision.

ABSTRACT

Bachelor's qualification thesis completed by Viktoriia Andriivna Yurchyshyna, a student of group SP-43 at Ivan Puluj Ternopil National Technical University, on the topic: "Design and Development software of a Web-Oriented Computer Vision System for Interactive Analysis of Lung X-ray Images". The thesis consists of 70 pages, includes 11 figures, 4 tables, 3 appendices, and a bibliography containing 22 references.

The aim of the thesis is to develop a web-oriented system for automated analysis of lung X-ray images using computer vision and artificial intelligence methods. The system is designed to improve the efficiency of primary diagnostics through automatic detection of pathologies in X-ray images, visualization of affected areas, and provision of medical reference information to users.

The thesis presents the development of the MedVision software system, whose architecture includes a client-side application, a server-side application, a database, and an artificial intelligence module for medical image analysis. Mechanisms for user registration and authentication, X-ray image uploading, automatic image analysis, pathology localization, generation of medical reference information, and storage of analysis history have been implemented.

The software system was developed using React, ASP.NET Core Web API, Entity Framework Core, Microsoft SQL Server, Python, and FastAPI technologies. The YOLOv8 computer vision model was applied for pathology detection in lung X-ray images, enabling localization of pathological findings and identification of their types with an associated confidence level.

Keywords: computer vision, artificial intelligence, medical image analysis, lung X-ray images, YOLOv8, web-oriented system, React, ASP.NET Core, medical diagnostics, MedVision.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД — база даних.

ДБН — державні будівельні норми.

ДСН — державні санітарні норми.

ЕОМ — електронно-обчислювальна машина.

ЗВО — заклад вищої освіти.

AI / ШІ — штучний інтелект.

КТ — комп’ютерна томографія.

МРТ — магнітно-резонансна томографія.

НАПБ — нормативний акт з пожежної безпеки.

НПАОП — нормативно-правовий акт з охорони праці.

ПЗ — програмне забезпечення.

ПЗВ — пристрій захисного вимкнення.

API (Application Programming Interface) — програмний інтерфейс застосунку.

ASP.NET — безкоштовна вебплатформа з відкритим вихідним кодом для створення вебзастосунків та сервісів від компанії Microsoft.

B2B (Business-to-Business) — комерційна взаємодія між компаніями.

CNN (Convolutional Neural Network) — згорткова нейронна мережа.

CPU (Central Processing Unit) — центральний процесор.

DTO (Data Transfer Object) — шаблон проєктування, об'єкт перенесення даних між шарами системи.

GPU (Graphics Processing Unit) — графічний процесор.

HTTP (Hypertext Transfer Protocol) — протокол передачі гіпертексту.

JSON (JavaScript Object Notation) — текстовий формат обміну даними, що базується на JavaScript.

PACS (Picture Archiving and Communication System) — медична система архівації та передачі зображень.

PE (Protective Earth) — захисне заземлення (захисний нульовий провідник).

SQL (Structured Query Language) — мова структурованих запитів для роботи з базами даних.

SRP (Single Responsibility Principle) — принцип єдиної відповідальності.

TN-S — тип системи заземлення, де робочий та захисний нулі розділені по всій довжині.

URL (Uniform Resource Locator) — єдиний вказівник ресурсу в інтернеті.

Web API — вебтехнологія для обміну даними між клієнтом та сервером через HTTP.

XAI (Explainable Artificial Intelligence) — порозумілий/пояснювальний штучний інтелект (стосується ваших теплових карт та Grad-CAM).

YOLOv8 (You Only Look Once version 8) — сучасна модель глибокого навчання для детекції об'єктів у реальному часі.

ВСТУП

Прогрес у сфері інформаційних технологій та нейромереж відкриває можливості для їх ефективного застосування у спеціалізованих галузях людської діяльності. Однією з галузей, де сучасні цифрові технології мають особливо важливе значення, є медицина. Використання програмних систем для підтримки прийняття рішень, автоматизованого аналізу медичних даних та діагностики захворювань дозволяє підвищити ефективність роботи медичного персоналу, зменшити ймовірність помилок та покращити якість медичного обслуговування пацієнтів.

Захворювання органів дихальної системи належать до найбільш поширених причин погіршення здоров'я населення у світі. Своєчасне виявлення патологічних змін у легенях має вирішальне значення для успішного лікування та зменшення ризику розвитку ускладнень. Одним із найпоширеніших методів діагностики захворювань органів грудної клітки є рентгенографічне дослідження. Рентгенівські знімки дозволяють виявляти широкий спектр патологій, зокрема пневмонію, фіброз легень, плевральний випіт, кардіомегалію, набряк легень та інші захворювання.

Незважаючи на високу інформативність рентгенографії, процес аналізу рентгенівських знімків потребує значного досвіду та кваліфікації лікаря-рентгенолога. В умовах великої кількості досліджень та обмежених ресурсів медичних закладів виникає потреба у використанні автоматизованих засобів підтримки діагностики, які можуть допомогти спеціалістам швидко виявляти потенційні патології та звертати увагу на найбільш важливі ділянки зображення.

Сучасні методи комп'ютерного зору та машинного навчання демонструють високі результати при розв'язанні задач аналізу медичних зображень. Особливої популярності набули глибокі нейронні мережі, які здатні автоматично визначати складні закономірності у візуальних даних та забезпечувати високу точність класифікації. Використання моделей штучного інтелекту для аналізу

рентгенівських знімків відкриває нові можливості щодо автоматизації процесів первинної діагностики та створення інтелектуальних медичних систем.

Актуальність теми роботи зумовлена необхідністю створення сучасних програмних засобів, що дозволяють здійснювати автоматизований аналіз рентгенівських знімків легень із використанням технологій штучного інтелекту та комп'ютерного зору. Розробка таких систем сприяє підвищенню ефективності медичних досліджень, скороченню часу аналізу зображень та покращенню доступності сучасних діагностичних технологій.

Об'єктом дослідження є процес автоматизованого аналізу рентгенівських знімків легень із використанням методів комп'ютерного зору та штучного інтелекту.

Предметом дослідження є методи, алгоритми та програмні засоби розробки веб-орієнтованих систем для виявлення патологій на рентгенівських знімках легень.

Метою роботи є проєктування та розробка веб-орієнтованої системи комп'ютерного зору для інтерактивного аналізу рентгенівських знімків легень, яка забезпечує автоматичне виявлення патологій, візуалізацію результатів аналізу та надання користувачеві додаткової медичної інформації.

Для досягнення поставленої мети необхідно виконати такі завдання:

1) провести аналіз предметної області та існуючих програмних рішень для автоматизованої діагностики захворювань легень;

2) дослідити сучасні методи комп'ютерного зору та машинного навчання для аналізу медичних зображень;

3) спроєктувати архітектуру веб-орієнтованої системи;

4) розробити клієнтську та серверну частини програмного забезпечення;

5) реалізувати інтеграцію моделі штучного інтелекту для автоматичного виявлення патологій на рентгенівських знімках;

6) реалізувати механізми збереження історії досліджень та формування медичної довідкової інформації;

7) провести тестування розробленої системи та оцінити результати її роботи.

Практичне значення роботи полягає у створенні функціональної веб-орієнтованої системи MedVision, яка дозволяє автоматизувати процес аналізу рентгенівських знімків легень, здійснювати виявлення патологій за допомогою технологій штучного інтелекту та надавати користувачам додаткову інформацію щодо виявлених захворювань. Результати роботи можуть бути використані як основа для подальшого розвитку інтелектуальних медичних інформаційних систем та систем підтримки прийняття рішень у сфері охорони здоров'я.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

У даному розділі проведено аналіз предметної області, розглянуто особливості діагностики захворювань органів грудної клітки за рентгенівськими знімками, досліджено існуючі програмні рішення для автоматизованого аналізу медичних зображень та сучасні методи комп'ютерного зору. На основі проведеного аналізу сформовано основні вимоги до веб-орієнтованої системи комп'ютерного зору для інтерактивного аналізу рентгенівських знімків легень.

1.1 Аналіз предметної області

Медична діагностика є однією з найважливіших складових сучасної системи охорони здоров'я. Від своєчасного та точного виявлення захворювань значною мірою залежить ефективність лікування пацієнтів та прогноз перебігу хвороби. Одним із найбільш поширених методів діагностики захворювань органів грудної клітки є рентгенографічне дослідження.

Рентгенографія дозволяє отримувати зображення внутрішніх органів за допомогою рентгенівського випромінювання та використовується для виявлення широкого спектра патологій легень і серцево-судинної системи. Серед основних переваг методу можна виділити відносну доступність, швидкість проведення дослідження та високу інформативність під час діагностики багатьох захворювань [1].

Щоденно в медичних закладах виконується значна кількість рентгенологічних досліджень, що створює додаткове навантаження на лікарів-рентгенологів. Аналіз кожного знімка потребує уважного вивчення особливостей анатомічних структур, оцінки можливих патологічних змін та формування медичного висновку. Навіть досвідчені спеціалісти можуть витрачати значний час на аналіз великої кількості досліджень.

Одним із перспективних напрямків розвитку сучасної медицини є використання методів штучного інтелекту (ШІ) та комп'ютерного зору для автоматизованого аналізу медичних зображень.

Штучний інтелект значно впливає на радіологію, покращуючи якість і швидкість діагностики. Сучасні алгоритми глибокого навчання (переважно згорткові нейронні мережі) здатні автоматично аналізувати медичні зображення – комп'ютерної томографії (КТ), магнітно-резонансної томографії (МРТ), рентгенів та мамограм – і виявляти патологічні зміни. Вони навчаються на величезних масивах даних, що дає змогу розпізнавати тонкі ознаки хвороб, які людське око може пропустити. Наприклад, команда Стенфордського університету навчила нейромережу на 112 тисячах рентгенів грудної клітки, щоб точно діагностувала захворювання. В результаті ці системи допомагають зменшити кількість пропущених випадків, прискорити опис знімків та знизити навантаження на лікарів [2].

Системи комп'ютерного зору здатні аналізувати цифрові медичні зображення, виявляти характерні ознаки патологій та визначати області можливих уражень. Такі системи не замінюють лікаря, але можуть використовуватись як інструмент підтримки прийняття рішень, що дозволяє скоротити час аналізу досліджень та підвищити точність первинної діагностики.

Для реалізації подібних систем необхідно забезпечити взаємодію між програмними компонентами, моделями штучного інтелекту та користувачами. Одним із найбільш зручних способів надання доступу до таких можливостей є використання веб-орієнтованих систем, які дозволяють працювати із сервісом через веб-браузер без встановлення додаткового програмного забезпечення.

Таким чином, створення веб-орієнтованої системи комп'ютерного зору для інтерактивного аналізу рентгенівських знімків легень є актуальним завданням, спрямованим на підвищення ефективності медичної діагностики та розширення можливостей застосування штучного інтелекту в медицині.

1.2 Аналіз існуючих програмних рішень

Останніми роками технології штучного інтелекту та комп'ютерного зору активно впроваджуються в медичну сферу для автоматизації процесів аналізу медичних зображень. Існує низка програмних рішень та наукових проєктів, спрямованих на виявлення патологій на рентгенівських знімках органів грудної клітки.

Одним із найвідоміших рішень є система CheXNet, розроблена дослідниками Стенфордського університету. Вона базується на використанні глибоких нейронних мереж для автоматичного аналізу рентгенівських знімків грудної клітки та виявлення ознак пневмонії. За результатами досліджень точність моделі в окремих випадках була співставною з результатами роботи лікарів-рентгенологів. Система продемонструвала ефективність застосування методів глибокого навчання для автоматизованої медичної діагностики [3].

Іншим відомим проєктом є ChestX-ray14, створений Національним інститутом здоров'я США (NIH). Проєкт являє собою масштабний набір рентгенівських знімків грудної клітки, що містить інформацію про 14 різних патологій. Даний набір даних широко використовується для навчання та тестування моделей комп'ютерного зору в галузі медичної діагностики [4]. (Один із похідних датасетів ChestX-ray14 використовувався під час навчання моделі для створення системи MedVision)

Серед сучасних комерційних рішень варто відзначити платформу Qure.ai, яка використовує технології штучного інтелекту для аналізу рентгенівських знімків та комп'ютерної томографії. Система дозволяє автоматично виявляти патології легень, формувати попередні висновки та допомагати медичним працівникам у процесі діагностики. Основною перевагою рішення є інтеграція в клінічні процеси та підтримка великої кількості медичних досліджень.

Також широкого поширення набули дослідницькі проєкти на основі моделей детекції об'єктів, зокрема YOLO. Такі системи дозволяють не лише

класифікувати наявність патології, а й визначати її локалізацію безпосередньо на рентгенівському знімку. Це забезпечує більш наочне представлення результатів аналізу та підвищує рівень довіри користувачів до отриманих висновків.

Проведений аналіз показав, що існуючі програмні рішення демонструють високу ефективність у задачах автоматизованого аналізу медичних зображень. Водночас більшість із них є або науковими дослідницькими проєктами, або комерційними системами з обмеженим доступом. Крім того, не всі рішення забезпечують одночасно локалізацію патологій, збереження історії досліджень, інтерактивний веб-інтерфейс та надання довідкової медичної інформації. Саме тому виникає необхідність розробки власної веб-орієнтованої системи MedVision, яка поєднує можливості автоматичного виявлення патологій, їх візуалізації та інтерактивної взаємодії з користувачем.

Для наочного представлення результатів аналізу існуючих програмних комплексів та наукових проєктів було проведено їх порівняльне дослідження за ключовими функціональними характеристиками (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз існуючих рішень для аналізу рентгенограм

Критерій порівняння	**CheXNet (Stanford)**	**ChestX-ray14 (NIH)**	**Qure.ai (qXR)**	**Проєкт MedVision**
Основне призначення	Наукова модель класифікації	Відкритий датасет та baseline	Комерційна медична система	Веб-орієнтована система
Доступність для користувача	Тільки вихідний код (GitHub)	Тільки набір даних для навчання	Комерційні B2B ліцензії	Відкритий веб-інтерфейс
Тип аналізу зображення	Мультилейбл класифікація	Класифікація патологій	Детекція та сегментація	Детекція (локалізація) об'єктів

Продовження таблиці 1.1

1	2	3	4	5
Візуалізація результатів	Теплові карти Grad-CAM	Відсутня	Bounding boxes, звіти	Чіткі рамки локалізації патолгій
Збереження історії пацієнта	Відсутнє	Відсутнє	Інтеграція з PACS лікарні	Авторизований профіль з БД
Основна технологія ШІ	DenseNet-121 (CNN)	Різні архітектури CNN	Власні закриті моделі	YOLOv8 (Real-time detection)

1.3 Аналіз методів комп'ютерного зору та штучного інтелекту

Комп'ютерний зір є одним із напрямків штучного інтелекту, що займається розробкою методів автоматичного аналізу та обробки цифрових зображень. Основною метою комп'ютерного зору є отримання корисної інформації із зображень та відеоданих, а також автоматичне розпізнавання об'єктів, їх класифікація та локалізація. Сучасні технології комп'ютерного зору широко застосовуються в медицині, транспорті, промисловості та системах безпеки [5].

Для аналізу медичних зображень найбільшого поширення набули методи глибокого навчання. На відміну від класичних алгоритмів обробки зображень, моделі глибокого навчання здатні самостійно визначати найбільш важливі ознаки об'єктів без необхідності ручного налаштування параметрів. Це дозволяє досягати високої точності при розв'язанні задач класифікації та детекції об'єктів.

Одним із найпоширеніших типів моделей для аналізу зображень є згорткові нейронні мережі (Convolutional Neural Networks, CNN). Дані мережі використовують спеціальні згорткові шари для автоматичного виділення характерних ознак зображення та формування високорівневого представлення

даних. Завдяки цьому CNN демонструють високу ефективність під час аналізу рентгенівських знімків та інших медичних зображень [6].

Для розв'язання задачі локалізації патологій у даній роботі було обрано модель YOLOv8. Дана модель належить до класу алгоритмів детекції об'єктів у реальному часі та забезпечує одночасне визначення класу об'єкта і його координат на зображенні. Основною перевагою YOLOv8 є висока швидкість роботи, точність виявлення об'єктів та можливість використання для аналізу великих наборів медичних зображень [7].

На відміну від моделей, що виконують лише класифікацію знімків, алгоритми детекції дозволяють визначати конкретні області локалізації патологічних змін. Це забезпечує більш наочне представлення результатів аналізу та дозволяє користувачу візуально оцінити ділянки, які були визначені моделлю як потенційно патологічні.

Отже, методи комп'ютерного зору та глибокого навчання є ефективним інструментом для автоматизованого аналізу рентгенівських знімків. Використання моделі YOLOv8 дозволяє реалізувати не лише класифікацію патологій, а й їх локалізацію, що є важливою складовою сучасних медичних інформаційних систем.

1.4 Характеристика розроблюваної системи MedVision

На основі проведеного аналізу предметної області та існуючих програмних рішень було прийнято рішення щодо розробки власної веб-орієнтованої системи MedVision, призначеної для автоматизованого аналізу рентгенівських знімків легень із використанням технологій комп'ютерного зору та штучного інтелекту.

Система MedVision розробляється як багатокомпонентний програмний продукт, що поєднує веб-інтерфейс користувача, серверну частину, базу даних та окремий модуль штучного інтелекту для аналізу медичних зображень. Основним призначенням системи є допомога користувачам у виявленні патологічних змін на

рентгенівських знімках органів грудної клітки та надання результатів аналізу у зручному візуальному вигляді.

Користувач взаємодіє із системою через веб-браузер, що дозволяє працювати із програмою без встановлення додаткового програмного забезпечення. Після завантаження рентгенівського знімка система автоматично передає його до модуля штучного інтелекту, який виконує аналіз зображення та визначає наявність можливих патологій. Результати обробки повертаються користувачу у вигляді висновку, рівня достовірності прогнозу та візуалізації ділянок, де були виявлені патологічні зміни.

Особливістю системи є наявність довідкового модуля, який автоматично формує коротку медичну інформацію щодо виявлених патологій. Користувач може ознайомитися з описом захворювання, характерними ознаками, можливими методами лікування та рекомендованими додатковими обстеженнями.

Крім того, система забезпечує збереження історії проведених досліджень, що дозволяє переглядати результати попередніх аналізів та отримувати детальну інформацію щодо кожного виконаного дослідження.

Тобто, система MedVision поєднує можливості сучасних веб-технологій та методів штучного інтелекту для створення зручного інструменту автоматизованого аналізу рентгенівських знімків легень.

1.5 Формування вимог до системи

На основі проведеного аналізу предметної області, існуючих програмних рішень та сучасних методів комп'ютерного зору було сформовано вимоги до веб-орієнтованої системи для інтерактивного аналізу рентгенівських знімків легень.

Основною метою системи є забезпечення автоматизованого аналізу рентгенівських знімків із використанням методів штучного інтелекту та надання користувачам зручного інструменту для отримання результатів дослідження. Система повинна забезпечувати можливість завантаження рентгенівських знімків,

їх автоматичної обробки та відображення результатів аналізу в зручній для користувача формі.

Функціональні вимоги до вимог системи:

1) реєстрація та авторизація користувачів;

2) завантаження рентгенівських знімків легень через веб-інтерфейс;

3) автоматичний аналіз зображень за допомогою моделі штучного інтелекту - після завантаження знімка система повинна автоматично передавати його на обробку модулю комп'ютерного зору;

4) визначення типу патології та рівня достовірності результату;

5) локалізація патологічних змін на рентгенівському знімку - користувач повинен отримувати візуалізацію ділянок зображення, на яких були виявлені ознаки патології;

6) відображення результатів аналізу та сформованого висновку;

7) надання довідкової інформації щодо виявлених патологій - користувачеві повинна надаватися додаткова інформація про виявлені захворювання, включаючи їх опис, характерні ознаки, можливі методи лікування та рекомендовані додаткові обстеження;

8) збереження історії виконаних досліджень - система повинна автоматично зберігати результати проведених аналізів у базі даних для подальшого перегляду;

9) перегляд детальної інформації про попередні аналізи - користувач повинен мати доступ до раніше виконаних досліджень із можливістю перегляду вихідного знімка, результатів аналізу, локалізації патологій та довідкової інформації.

10) підтримка рольової моделі користувачів (Doctor, Student);

11) верифікація користувачів за номером професійного або студентського посвідчення;

12) надання доступу до функціональності системи відповідно до статусу верифікації користувача.

Нефункціональних вимог системи:

1) зручний та зрозумілий користувацький інтерфейс - система повинна забезпечувати інтуїтивно зрозумілу навігацію та простоту використання для користувачів без спеціальної технічної підготовки;

2) забезпечення швидкої обробки рентгенівських знімків - час аналізу повинен бути мінімальним та не створювати значних затримок під час роботи користувача із системою;

3) можливість роботи через сучасні веб-браузери - програмний продукт повинен коректно функціонувати у поширених браузерах без необхідності встановлення додаткового програмного забезпечення;

4) забезпечення збереження даних користувачів та результатів аналізу - система повинна гарантувати цілісність і доступність інформації, що зберігається у базі даних;

5) масштабованість програмного рішення для подальшого розширення функціональності - архітектура системи повинна дозволяти додавання нових можливостей без суттєвих змін існуючих компонентів;

6) модульна архітектура системи, що спрощує супровід та модернізацію програмного забезпечення - окремі компоненти системи повинні бути незалежними між собою, що забезпечить зручність тестування, підтримки та подальшого розвитку програмного продукту.

Для реалізації поставлених вимог було прийнято рішення використовувати архітектуру клієнт-серверного типу. Клієнтська частина реалізується у вигляді веб-застосунку на основі React, серверна частина розробляється з використанням ASP.NET Core Web API, а для зберігання даних використовується база даних Microsoft SQL Server. Аналіз рентгенівських знімків виконується окремим сервісом на Python із використанням моделі комп'ютерного зору YOLOv8.

Таким чином, сформовані вимоги визначають основні функціональні можливості та архітектурні особливості системи MedVision і є основою для подальшого проєктування та реалізації програмного продукту.

2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ MEDVISION

У даному розділі розглянуто процес проєктування та розробки веб-орієнтованої системи MedVision для інтерактивного аналізу рентгенівських знімків легень. Описано архітектуру програмної системи, структуру бази даних, програмні компоненти та особливості реалізації модуля штучного інтелекту. Також наведено опис розробленого користувацького інтерфейсу та принципів взаємодії між окремими компонентами системи.

2.1 Загальна архітектура системи MedVision

Система MedVision розроблена відповідно до багаторівневої клієнт-серверної архітектури, яка забезпечує розподіл функціональних обов'язків між окремими компонентами програмного забезпечення. Такий підхід дозволяє підвищити гнучкість системи, спростити її супровід та забезпечити можливість подальшого розширення функціональності.

Під час проєктування системи було розглянуто декілька можливих архітектурних підходів, серед яких монолітна архітектура, мікросервісна архітектура та розподілена Web API архітектура. Для вибору найбільш доцільного рішення було проведено їх порівняння за ключовими критеріями, результати якого наведено в таблиці 2.1.

Таблиця 2.1. Порівняльний аналіз архітектурних підходів до побудови системи

Критерій порівняння	**Монолітна архітектура**	**Мікросервісна архітектура**	**Розподілена Web API архітектура (обрана)**
Швидкість розробки	Висока на початкових етапах	Низька (потребує налаштування інфраструктури)	Середня (оптимальна для інтеграції Python/.NET)

Продовження таблиці 2.1

1	2	3	4
Масштабованість	Обмежена (масштабується весь застосунок)	Висока (кожен сервіс окремо)	Висока (сервіс II та бізнес-логіка масштабуються незалежно)
Складність розгортання	Низька	Дуже висока	Середня
Ефективність використання GPU/CPU	Низька (конфлікт середовищ виконання)	Висока	Висока (FastAPI ізольовано використовує ресурси під YOLOv8)

Як видно з таблиці 2.1, монолітна архітектура забезпечує швидку розробку на початкових етапах, проте має обмежені можливості масштабування та ускладнює інтеграцію компонентів, реалізованих різними технологіями. Мікросервісний підхід забезпечує максимальну гнучкість та масштабованість, однак потребує складної інфраструктури та значних витрат на підтримку.

З огляду на особливості проєкту MedVision було обрано розподілену Web API архітектуру, яка дозволяє ефективно поєднати серверну частину на платформі ASP.NET Core та модуль штучного інтелекту, реалізований засобами Python і FastAPI. Такий підхід забезпечує незалежний розвиток окремих компонентів системи, спрощує їх супровід та створює умови для подальшого масштабування програмного продукту.

Отже, архітектура системи MedVision складається з чотирьох основних компонентів:

- клієнтської частини (Frontend);
- серверної частини (Backend);

- бази даних;
- модуля штучного інтелекту для аналізу рентгенівських знімків.

Клієнтська частина реалізована у вигляді веб-застосунку з використанням бібліотеки React. Вона забезпечує взаємодію користувача із системою, включаючи реєстрацію та авторизацію, завантаження рентгенівських знімків, перегляд результатів аналізу, роботу з історією досліджень та отримання довідкової інформації щодо виявлених патологій.

Серверна частина системи реалізована на платформі ASP.NET Core Web API. Вона відповідає за обробку запитів від клієнтського застосунку, взаємодію з базою даних, збереження інформації про користувачів та результати аналізу, а також організацію обміну даними між веб-застосунком та модулем штучного інтелекту.

Для зберігання інформації використовується система керування базами даних Microsoft SQL Server. У базі даних зберігаються відомості про зареєстрованих користувачів, результати аналізу рентгенівських знімків, шляхи до збережених файлів та історія виконаних досліджень.

Аналіз рентгенівських знімків виконується окремим сервісом, реалізованим мовою Python із використанням фреймворку FastAPI. Основу модуля штучного інтелекту становить модель комп'ютерного зору YOLOv8, яка виконує автоматичне виявлення патологій на рентгенівських знімках та визначає їх локалізацію.

Під час роботи системи користувач завантажує рентгенівський знімок через веб-інтерфейс. Після цього зображення передається на сервер ASP.NET Core, який виконує необхідну обробку та надсилає файл до сервісу штучного інтелекту. Модель YOLOv8 аналізує зображення, визначає наявні патології та формує результати аналізу. Отримані дані повертаються до серверної частини, зберігаються в базі даних та відображаються користувачу через веб-інтерфейс.

Запропонована архітектура дозволяє незалежно розвивати окремі компоненти системи, забезпечує зручну інтеграцію технологій штучного інтелекту та створює основу для подальшого масштабування програмного продукту.

Важливою особливістю розробленої системи є використання окремого сервісу штучного інтелекту, який функціонує незалежно від основного веб-застосунку. Це дозволяє ефективно розподіляти обчислювальні ресурси та забезпечує можливість модернізації алгоритмів аналізу медичних зображень без зміни інших компонентів системи. Крім того, модульна структура архітектури спрощує тестування окремих компонентів та підвищує надійність роботи програмного забезпечення.

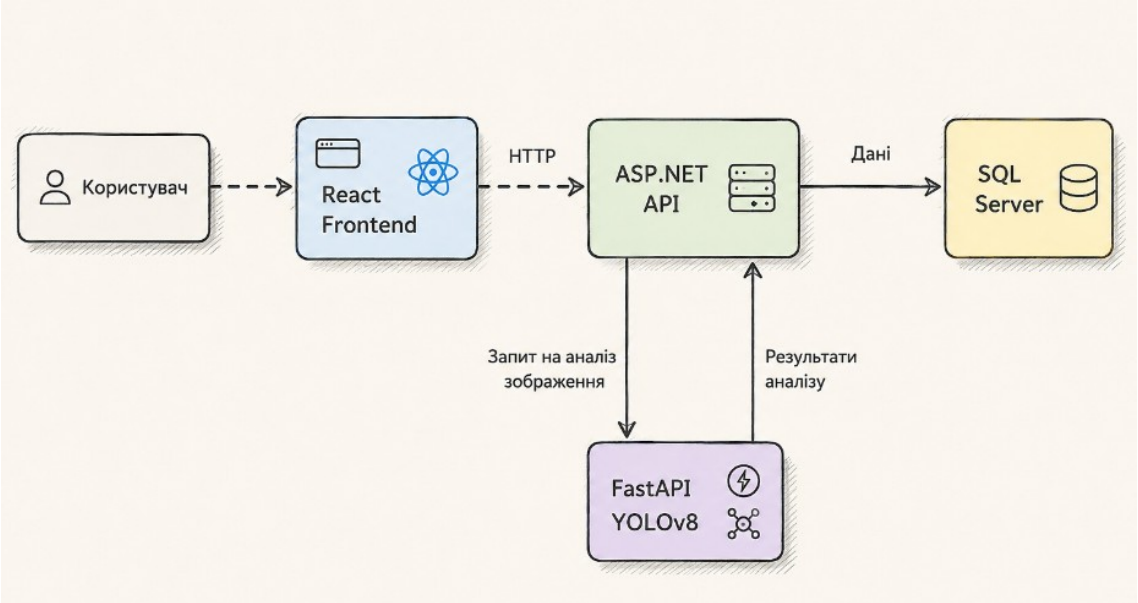

Рисунок 2.1 — Загальна архітектура системи MedVision

Отже, обрана архітектура забезпечує ефективну взаємодію між користувачем, веб-застосунком, серверною частиною, базою даних та модулем штучного інтелекту. Використання багаторівневої клієнт-серверної архітектури дозволяє підвищити масштабованість системи та спрощує її подальшу модернізацію.

2.2 Проєктування бази даних

База даних є важливою складовою системи MedVision, оскільки забезпечує зберігання інформації про користувачів та результати виконаних досліджень.

Використання централізованого сховища даних дозволяє організувати довготривале збереження результатів аналізу, забезпечити доступ до історії досліджень та реалізувати механізми авторизації користувачів.

Для реалізації сховища даних було обрано систему керування базами даних Microsoft SQL Server. Вибір Microsoft SQL Server зумовлений декількома факторами. По-перше, дана система керування базами даних забезпечує високу продуктивність та надійність під час роботи з реляційними даними. По-друге, SQL Server має тісну інтеграцію з платформою .NET та технологією Entity Framework Core, що значно спрощує розробку серверної частини застосунку. Крім того, система підтримує механізми резервного копіювання, захисту даних та керування доступом, які є важливими для програмних продуктів, що працюють із медичною інформацією. Також використання SQL Server дозволяє легко масштабувати систему у разі збільшення кількості користувачів та обсягу збережених даних.

Під час проєктування бази даних було виділено дві основні сутності: користувач та аналіз рентгенівського знімка. Сутність користувача містить інформацію, необхідну для автентифікації та авторизації в системі. Сутність аналізу рентгенівського знімка використовується для збереження результатів роботи моделі штучного інтелекту та інформації про виконані дослідження.

Таблиця Users призначена для зберігання облікових записів користувачів. До її основних атрибутів належать унікальний ідентифікатор користувача, адреса електронної пошти, хеш пароля, роль користувача, номер посвідчення або студентського квитка та ознака успішного проходження верифікації.

Використання хешування паролів дозволяє підвищити рівень безпеки та запобігти зберіганню конфіденційних даних у відкритому вигляді. Додаткові атрибути Role, VerificationNumber та IsVerified забезпечують підтримку рольової моделі системи та механізму підтвердження статусу користувача.

Таблиця XrayAnalyses використовується для збереження результатів аналізу рентгенівських знімків. У ній зберігаються шляхи до завантажених зображень та згенерованих теплових карт, результат класифікації, рівень достовірності

прогнозу, текстовий висновок, інформація про виявлені патології та дата виконання аналізу.

Між таблицями реалізовано зв'язок типу «один до багатьох», відповідно до якого один користувач може виконати необмежену кількість аналізів рентгенівських знімків. Кожний запис про аналіз належить лише одному користувачу. Такий підхід дозволяє забезпечити персональне зберігання історії досліджень та спрощує пошук необхідної інформації.

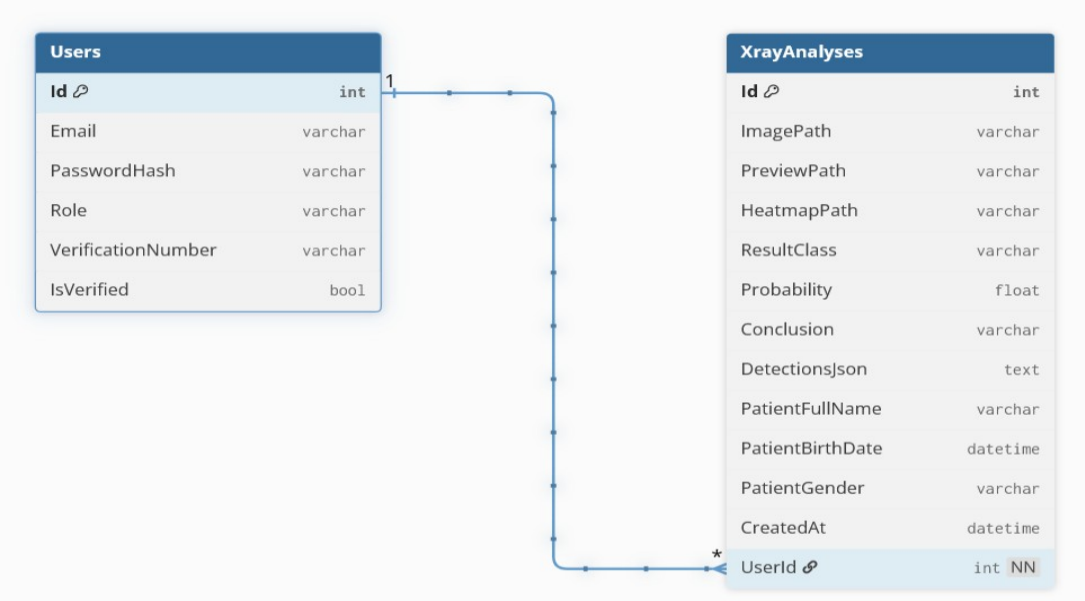

Рисунок 2.2 — ER-діаграма бази даних системи MedVision

Особливу увагу слід приділити таблиці XrayAnalyses, яка використовується для зберігання результатів аналізу рентгенівських знімків. Кожен запис у таблиці відповідає окремому дослідженню, виконаному користувачем.

Основні атрибути таблиці мають таке призначення:

- Id – унікальний ідентифікатор аналізу;
- ImagePath – шлях до завантаженого рентгенівського знімка;
- HeatmapPath – шлях до зображення з візуалізацією локалізації патологій, сформованого моделлю штучного інтелекту;
- ResultClass – загальний результат аналізу, що вказує на наявність або

відсутність патологічних змін;

- Probability – рівень достовірності прогнозу моделі у відсотках;
- Conclusion – текстовий висновок, сформований на основі результатів аналізу;
- DetectionsJson – структурована інформація про виявлені патології та їх характеристики у форматі JSON;
- CreatedAt – дата та час виконання аналізу;
- UserId – зовнішній ключ, що визначає користувача, якому належить дане дослідження.

Запропонована структура таблиці дозволяє зберігати як результати роботи моделі штучного інтелекту, так і допоміжну інформацію, необхідну для формування історії досліджень та детального перегляду результатів аналізу.

Розроблена структура бази даних орієнтована на реалізацію основного функціоналу системи MedVision та підтримує механізми автентифікації, авторизації й верифікації користувачів.

На поточному етапі система підтримує дві категорії користувачів: лікарів та студентів. Для кожного користувача в базі даних додатково зберігається його роль (Role), номер посвідчення або студентського квитка (VerificationNumber), а також результат проходження верифікації (IsVerified).

Під час реєстрації користувач вказує свою роль та ідентифікаційний номер. Серверна частина системи виконує перевірку наданих даних за допомогою внутрішнього реєстру верифікації. Результат перевірки автоматично зберігається в базі даних і використовується для підтвердження статусу користувача.

Такий підхід дозволяє підвищити достовірність інформації про користувачів системи та створює основу для подальшого розширення функціональних можливостей програмного продукту.

У перспективі подальшого розвитку програмного продукту можливе впровадження додаткових ролей користувачів, зокрема лікарів-рентгенологів, пацієнтів та адміністраторів системи. У такому випадку структура бази даних

може бути розширена новими сутностями для зберігання медичної інформації про пацієнтів, даних про медичних працівників, журналів дій користувачів та механізмів розмежування прав доступу.

Крім того, перспективним напрямком модернізації є нормалізація даних про виявлені патології. На даний момент інформація про них зберігається у полі DetectionsJson, що забезпечує простоту реалізації та достатню гнучкість під час роботи із результатами аналізу. У майбутніх версіях системи можливим є виділення окремих таблиць для патологій, медичних висновків та додаткових діагностичних даних, що дозволить підвищити масштабованість та функціональні можливості системи.

2.3 Розробка серверної частини системи

Серверна частина системи MedVision реалізована з використанням платформи ASP.NET Core Web API. Основним призначенням серверного застосунку є забезпечення взаємодії між клієнтською частиною системи, базою даних та модулем штучного інтелекту. Сервер виконує обробку HTTP-запитів, збереження даних, керування користувачами та передачу рентгенівських знімків до сервісу аналізу медичних зображень.

Крім цього, серверна частина реалізує механізми реєстрації, авторизації та верифікації користувачів. Система підтримує дві ролі користувачів — лікар та студент. Під час реєстрації виконується перевірка номера посвідчення або студентського квитка за допомогою внутрішнього реєстру верифікації, після чого результат перевірки зберігається в базі даних.

Для роботи із базою даних використовується технологія Entity Framework Core, яка забезпечує об'єктно-реляційне відображення даних та спрощує виконання операцій створення, читання, оновлення та видалення записів. Взаємодія із базою даних реалізована через контекст даних AppDbContext та відповідні сутності предметної області.

До складу серверної частини входять моделі даних, DTO-об'єкти, контролери API та механізми взаємодії із сервісом штучного інтелекту. Такий підхід дозволяє забезпечити розділення відповідальності між окремими компонентами програмного забезпечення та підвищує зручність супроводу системи.

2.3.1 Застосування принципів об'єктно-орієнтованого проєктування

Під час розробки серверної частини системи MedVision використовувалися сучасні підходи до об'єктно-орієнтованого проєктування, що дозволило забезпечити гнучкість архітектури, зручність супроводу програмного забезпечення та можливість його подальшого розвитку.

Одним із основних принципів, які були враховані під час розробки системи, є принцип єдиної відповідальності (Single Responsibility Principle, SRP). Відповідно до даного принципу кожен програмний компонент повинен виконувати лише одну функцію. У розробленій системі функціональність розподілена між окремими компонентами. Контролери відповідають за обробку HTTP-запитів та формування відповідей клієнту, моделі описують структуру даних предметної області, DTO-об'єкти використовуються для передачі даних між клієнтською та серверною частинами системи, а контекст бази даних забезпечує взаємодію із сховищем даних.

Для реалізації взаємодії між окремими компонентами застосунку використовується механізм впровадження залежностей (Dependency Injection), який є невід'ємною складовою платформи ASP.NET Core. Даний підхід дозволяє зменшити зв'язність між програмними модулями та забезпечує можливість їх незалежного тестування та модифікації. Реєстрація необхідних сервісів виконується централізовано під час конфігурації застосунку в файлі Program.cs.

Важливу роль у структурі програмного забезпечення відіграє використання шаблону DTO (Data Transfer Object). Для передачі даних між шарами системи

були створені спеціалізовані класи LoginDto, RegisterDto, DetectionDto, AiPredictionResponse та XrayAnalysisResponseDto. Використання DTO дозволяє передавати лише необхідні дані, приховуючи внутрішню структуру моделей бази даних та зменшуючи обсяг інформації, що передається між компонентами системи.

Архітектура серверної частини також реалізує принцип розділення відповідальності (Separation of Concerns). Бізнес-логіка обробки аналізів, взаємодія з базою даних, робота з моделями штучного інтелекту та обробка HTTP-запитів виконуються окремими компонентами системи. Такий підхід спрощує супровід програмного забезпечення та дозволяє незалежно модернізувати окремі модулі без внесення значних змін до інших частин проєкту.

Застосування принципів об'єктно-орієнтованого проєктування дозволило створити зрозумілу та масштабовану архітектуру серверної частини системи MedVision, що відповідає сучасним вимогам до розробки веб-орієнтованих програмних продуктів.

2.4 Розробка модуля аналізу медичних зображень

Модуль аналізу медичних зображень є ключовим компонентом системи MedVision, оскільки саме він забезпечує автоматичне виявлення патологій на рентгенівських знімках легень. Для реалізації даного модуля було використано мову програмування Python та сучасні засоби розробки систем штучного інтелекту.

Основним завданням модуля є обробка рентгенівських знімків, виявлення патологічних змін, визначення їх локалізації та формування результатів аналізу для подальшого відображення користувачу через веб-інтерфейс системи.

2.4.1 Обґрунтування вибору технологій

Для реалізації модуля аналізу медичних зображень було обрано мову програмування Python. Дане рішення обумовлене широким використанням Python у сфері машинного навчання та комп'ютерного зору, а також наявністю великої кількості бібліотек для роботи з нейронними мережами та медичними зображеннями.

Для створення сервісу аналізу було використано фреймворк FastAPI. Його перевагами є висока швидкодія, підтримка асинхронної обробки запитів та зручна інтеграція із сучасними моделями штучного інтелекту.

Як основний алгоритм аналізу медичних зображень було обрано модель YOLOv8. На відміну від класичних моделей класифікації, YOLOv8 дозволяє не лише визначати наявність патології, а й локалізувати область її розташування на рентгенівському знімку. Це забезпечує більш наочне представлення результатів аналізу та підвищує інформативність системи.

2.4.2 Підготовка та навчання моделі YOLOv8

Для реалізації автоматичного виявлення патологій на рентгенівських знімках легень було використано модель детекції об'єктів YOLOv8. Навчання моделі виконувалося на основі відкритого медичного набору даних VinBigData Chest X-ray Abnormalities Detection, який містить рентгенівські знімки грудної клітки з розміченими областями патологічних змін [8].

На етапі підготовки даних було виконано завантаження зображень та анотацій, що містять інформацію про тип патології та координати відповідних областей ураження.

Для забезпечення сумісності з архітектурою YOLOv8 координати прямокутних областей локалізації патологій були перетворені у формат YOLO, який використовує координати центру об'єкта та його відносні розміри. Після

цього набір даних було розділено на навчальну та валідаційну вибірки у співвідношенні 80% до 20%.

Навчання моделі здійснювалося із використанням фреймворку Ultralytics YOLOv8. В якості базової моделі було використано YOLOv8n, яка характеризується невеликим розміром та високою швидкістю виконання. Основні параметри навчання наведено в таблиці 2.6.

Таблиця 2.2 — Основні параметри навчання моделі

Параметр	**Значення**
Архітектура моделі	YOLOv8n
Кількість епох	30
Розмір зображення	256×256
Batch size	16
Навчальна вибірка	80%
Валідаційна вибірка	20%

Після завершення процесу навчання фреймворк Ultralytics автоматично зберіг найкращу версію моделі у файлі best.pt. Вибір моделі здійснювався на основі показників якості на валідаційній вибірці. Саме цей файл використовується під час виконання прогнозування та виявлення патологій у розробленій системі.

2.4.3 Реалізація сервісу аналізу зображень

Модуль аналізу реалізовано як окремий веб-сервіс на основі FastAPI. Такий підхід дозволяє відокремити процеси обробки медичних зображень від основного веб-застосунку та забезпечує незалежний розвиток компонентів системи.

Після отримання рентгенівського знімка сервіс виконує завантаження навченої моделі YOLOv8 та здійснює аналіз зображення. У процесі роботи модель

визначає наявні патології, координати зон ураження та рівень достовірності кожного прогнозу.

Результати аналізу формуються у вигляді структурованого набору даних, який містить перелік виявлених патологій, їх характеристики та рівень впевненості моделі. Додатково формуються два варіанти візуалізації результатів аналізу: теплова карта локалізації патологічних змін та зображення з нанесеними межами виявлених патологій

2.4.4 Алгоритм аналізу рентгенівських знімків

Процес аналізу рентгенівських знімків складається з декількох послідовних етапів. Спочатку користувач завантажує знімок через веб-інтерфейс системи. Після цього файл передається до сервісу аналізу медичних зображень, де виконується його попередня обробка, що включає перевірку формату, нормалізацію даних та підготовку зображення до подальшого аналізу. Наступним етапом є обробка знімка за допомогою моделі штучного інтелекту YOLOv8, яка здійснює пошук характерних ознак патологічних змін у легенях..

На основі отриманих результатів визначаються виявлені патології, формується медичний висновок та створюється теплова карта локалізації патологічних змін. Додатково система розраховує рівень достовірності прогнозу для кожної виявленої патології, що дозволяє оцінити впевненість моделі у прийнятому рішенні. Після завершення аналізу результати повертаються до серверної частини системи, де зберігаються в базі даних разом із супровідною інформацією про дослідження. Надалі користувач може переглядати історію виконаних аналізів, відкривати детальну інформацію про кожне дослідження та повторно ознайомлюватися з отриманими результатами..

Рисунок 2.3 — Activity Diagram аналізу знімка

На рисунку 2.3 представлено Activity Diagram аналізу рентгенівського знімка, що реалізований у модулі штучного інтелекту системи MedVision.

2.4.5 Формат обміну даними між серверною частиною та модулем штучного інтелекту

Для забезпечення взаємодії між серверною частиною системи та модулем штучного інтелекту використовується обмін даними через HTTP-запити. Серверний застосунок ASP.NET Core Web API виконує роль посередника між клієнтською частиною та сервісом аналізу медичних зображень, реалізованим на базі FastAPI. Такий підхід дозволяє ізолювати модуль штучного інтелекту від інших компонентів системи та забезпечує можливість його незалежного розвитку й масштабування.

Після завантаження користувачем рентгенівського знімка файл надходить до серверної частини системи. Сервер виконує перевірку отриманих даних та формує HTTP-запит до сервісу FastAPI. Передача зображення здійснюється у форматі multipart/form-data, що дозволяє передавати бінарні файли через HTTP-протокол.

Після завершення аналізу сервіс FastAPI формує відповідь у форматі JSON. Використання формату JSON забезпечує зручність передачі структурованих даних між різними програмними платформами та спрощує подальшу обробку результатів аналізу серверною частиною системи.

Лістинг 2.1 — Приклад відповіді сервісу аналізу наведено нижче:


```
    "id": 128,
      "imagePath": "/uploads/0efd2579-1f73-4255-a3c0-
d9374019b21c.dicom",
      "resultClass": "Abnormal",
      "probability": 55.34,
      "conclusion": "Виявлено патології.",
      "createdAt": "2026-06-12T06:44:08.8112493Z",
      "heatmapPath": "/predictions/pred_a9575d00-99a2-494b-ac92-
ba716a1a2f86.jpg",
      "detectionsJson": "[{"..."}]",
      "detections": [
        {
          "className": "Cardiomegaly",
          "confidence": 55.34,
          "description": "Збільшення серця. Може свідчити про
серцеву недостатність."
        }
      ],
      "previewPath": "/predictions/2e575d6b-6a0c-48c0-bdaf-
bb8cec4100c7.png",
      "patientFullName": Jon Dou,
      "patientBirthDate": 15.08.2005,
      "patientGender": Male}
```

Отримана відповідь використовується серверною частиною для збереження результатів аналізу в базі даних та подальшого відображення інформації користувачу через веб-інтерфейс системи.

Для збереження результатів аналізу в базі даних використовується поле DetectionsJson, яке містить серіалізоване JSON-представлення списку виявлених патологій. Під час роботи клієнтської частини ці дані десеріалізуються у колекцію об'єктів detections та використовуються для відображення результатів аналізу й формування медичної довідки.

2.5 Проєктування сценаріїв використання системи

Для визначення способів взаємодії користувачів із системою MedVision було виконано моделювання сценаріїв використання. Даний етап проєктування дозволяє визначити основні функціональні можливості програмного продукту та описати послідовність дій користувачів під час роботи із системою.

У розробленому програмному продукті реалізовано рольову модель доступу. Система підтримує дві категорії користувачів: лікарів та студентів. Під час реєстрації користувач обирає відповідну роль та проходить процедуру верифікації за номером посвідчення або студентського квитка.

Основними сценаріями використання системи є реєстрація, авторизація, верифікація користувача, завантаження рентгенівського знімка, виконання автоматизованого аналізу, перегляд історії досліджень, пошук та фільтрація результатів, перегляд детальної інформації про аналіз, а також видалення окремих записів з історії досліджень.

На рисунку 2.4 представлено діаграму варіантів використання системи MedVision.

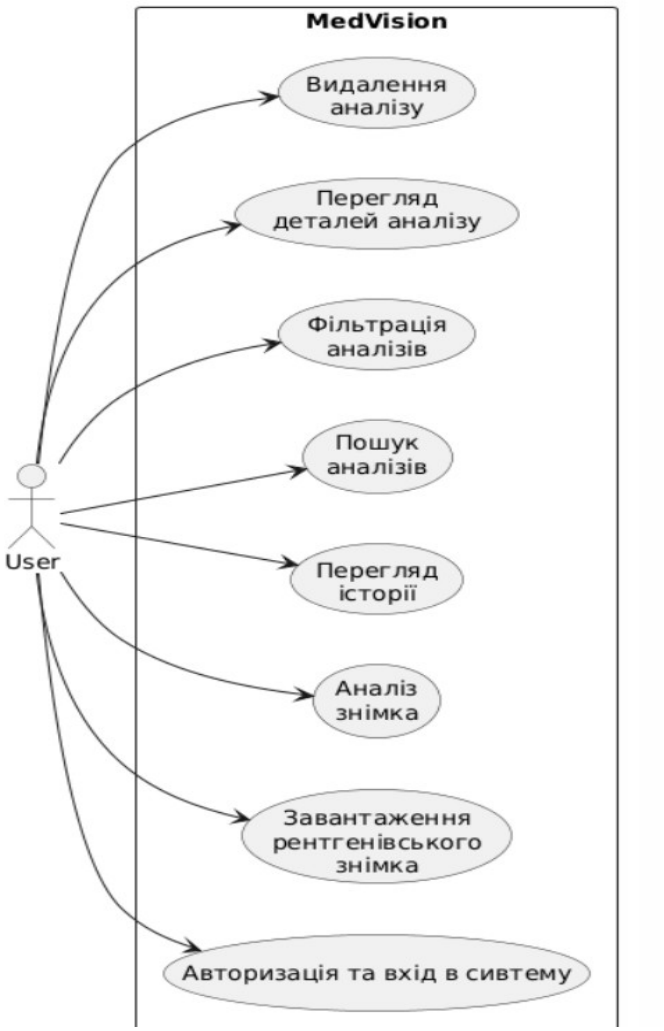

Рисунок 2.4 — Use Case Diagram для системи MedVision

Для більш детального опису процесу взаємодії між окремими компонентами системи було побудовано діаграму послідовностей. На відміну від діаграми варіантів використання, яка відображає функціональні можливості програмного продукту, діаграма послідовностей демонструє порядок обміну повідомленнями між користувачем, клієнтською частиною, серверним застосунком ASP.NET Core Web API, модулем штучного інтелекту FastAPI та базою даних Microsoft SQL Server.

На рисунку 2.5 представлено діаграму послідовностей процесу аналізу рентгенівського знімка в системі MedVision.

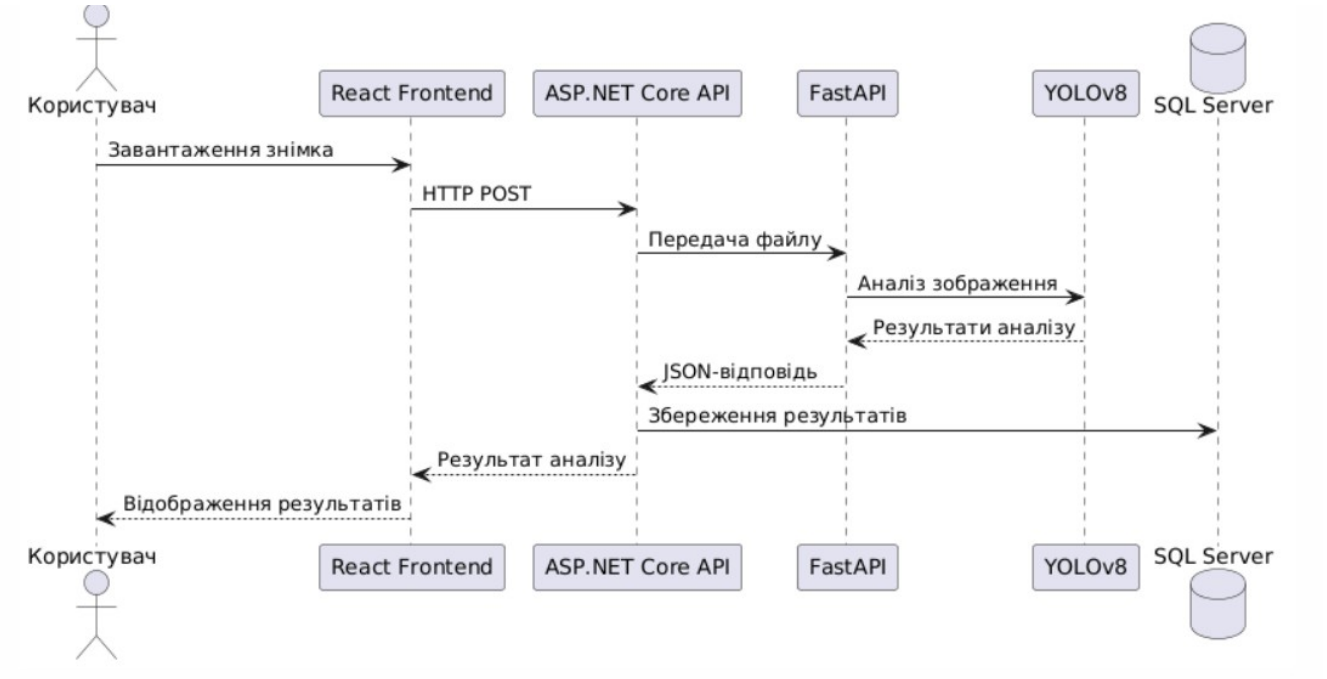

Рисунок 2.5 — Sequence Diagram для системи MedVision(Аналіз знімка)

Відповідно до наведеної діаграми, процес аналізу починається із завантаження користувачем рентгенівського знімка через веб-інтерфейс системи. Клієнтська частина передає файл до серверного застосунку ASP.NET Core Web API, який виконує первинну обробку запиту та надсилає зображення до сервісу аналізу медичних зображень, реалізованого на базі FastAPI.

Після отримання файлу сервіс FastAPI передає його до моделі комп'ютерного зору YOLOv8 для виконання аналізу. Модель здійснює пошук патологічних змін на рентгенівському знімку, визначає їх локалізацію та формує результати аналізу. Отримані дані повертаються до серверної частини у форматі JSON.

Серверний застосунок зберігає результати аналізу у базі даних Microsoft SQL Server та передає підготовлену відповідь клієнтській частині. Після цього результати дослідження відображаються користувачу у веб-інтерфейсі разом із візуалізацією локалізації патологій та медичною довідковою інформацією.

Таким чином, побудовані UML-діаграми дозволяють наочно представити як функціональні можливості системи MedVision, так і внутрішню логіку взаємодії

між її основними компонентами під час виконання аналізу рентгенівських знімків легень.

Для забезпечення контрольованого доступу до функціональних можливостей системи MedVision реалізовано механізми реєстрації, авторизації та верифікації користувачів. Серверна частина системи підтримує рольову модель доступу та дозволяє працювати з двома категоріями користувачів: лікарями та студентами.

Під час реєстрації користувач вказує адресу електронної пошти, пароль, роль та номер посвідчення. Для лікарів використовується номер професійного посвідчення, а для студентів — номер студентського квитка. Варто наголосити, що категорія студентів у системі охоплює виключно майбутніх медичних фахівців. Верифікація належності здобувача освіти саме до вищого медичного навчального закладу здійснюється автоматично на основі аналізу серії студентського квитка. Серія документа містить закодований буквено-цифровий префікс, який жорстко закріплений за конкретним закладом освіти або галуззю знань у Єдиній державній електронній базі з питань освіти (ЄДЕБО). Програмний модуль автентифікації бекенду зіставляє введену серію із вбудованим довідником акредитованих медичних університетів. У разі невідповідності серії встановленим критеріям профільного закладу, реєстрація користувача у статусі студента-медика відхиляється, що запобігає нецільовому використанню обчислювальних ресурсів моделей штучного інтелекту

Після надходження запиту сервер перевіряє відсутність користувача з аналогічною адресою електронної пошти та виконує процедуру верифікації за допомогою внутрішнього реєстру VerificationRegistry. У разі успішної перевірки користувачу присвоюється статус верифікованого користувача.

Інформація про роль користувача зберігається у полі Role, номер посвідчення — у полі VerificationNumber, а результат проходження перевірки — у полі IsVerified. Паролі користувачів зберігаються у вигляді криптографічних хешів із використанням бібліотеки BCrypt.

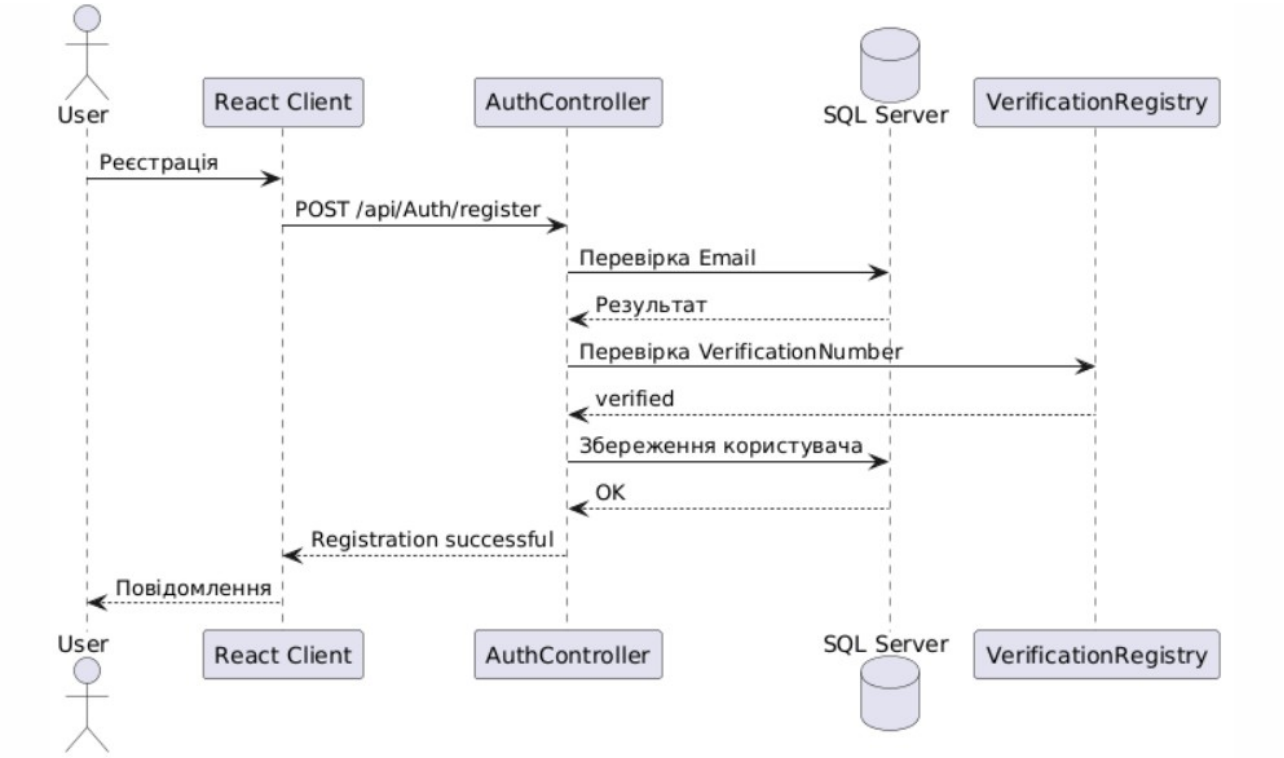

Рисунок 2.6 — Sequence Diagram для системи MedVision (Авторизація)

Після успішної реєстрації користувач може виконати авторизацію та отримати доступ до функціональних можливостей системи, описаних вище, відповідно до своєї ролі.

2.6 Розробка користувацького інтерфейсу

Користувацький інтерфейс є важливою складовою системи MedVision, оскільки саме через нього забезпечується взаємодія користувача з функціональними можливостями програмного продукту. Під час розробки інтерфейсу основна увага приділялася простоті використання, зрозумілості навігації та зручності представлення результатів аналізу рентгенівських знімків.

Для реалізації клієнтської частини системи було використано бібліотеку React, яка забезпечує створення сучасних односторінкових веб-застосунків та дозволяє організувати динамічне оновлення інтерфейсу без перезавантаження

сторінки. Для стилізації елементів інтерфейсу використовувалися засоби Bootstrap та CSS.

Під час проєктування інтерфейсу було реалізовано сторінку авторизації користувачів, головне вікно аналізу рентгенівських знімків, модуль перегляду історії досліджень та вікно детального перегляду результатів аналізу.

Для підвищення зручності роботи користувача реалізовано механізми пошуку та фільтрації історії аналізів, перегляд детальної інформації про результати досліджень, а також можливість видалення окремих записів з історії аналізів. Кожен результат дослідження містить інформацію про пацієнта, дату проведення аналізу, виявлені патології та сформовані візуалізації результатів роботи моделі штучного інтелекту.

З користувацьким інтерфейсом можна ознайомитися у додатку А.

3. ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА СИСТЕМИ

Після завершення проєктування та розробки програмної системи важливим етапом життєвого циклу програмного забезпечення є її тестування. Основною метою тестування є перевірка відповідності реалізованого програмного продукту встановленим функціональним та нефункціональним вимогам, виявлення можливих помилок і забезпечення стабільної роботи системи в різних сценаріях використання.

Для розробленої системи MedVision було проведено комплексне тестування, яке охоплювало перевірку серверної частини застосунку, функціональних можливостей користувацького інтерфейсу та роботи модуля штучного інтелекту для аналізу рентгенівських знімків легень. Особлива увага приділялася коректності взаємодії між компонентами системи, обробці даних користувачів, формуванню результатів аналізу та збереженню історії досліджень.

У даному розділі наведено результати автоматизованого та функціонального тестування системи, розглянуто особливості її впровадження, а також визначено основні напрями подальшого розвитку та підтримки програмного продукту.

3.1 Автоматизоване тестування серверної частини системи

Основною метою тестування є виявлення помилок, перевірка правильності реалізації функціональних можливостей та забезпечення стабільної роботи програмного продукту. Своєчасне проведення тестування дозволяє підвищити якість програмного забезпечення, зменшити кількість дефектів та забезпечити відповідність системи встановленим вимогам

Одним із сучасних підходів до перевірки програмного забезпечення є автоматизоване тестування. На відміну від ручного тестування, автоматизовані тести виконуються за допомогою спеціалізованих програмних засобів без безпосередньої участі користувача. Такий підхід дозволяє значно скоротити час

перевірки функціональності системи, підвищити повторюваність тестів та зменшити ймовірність виникнення людських помилок під час перевірки програмного забезпечення [9].

Для автоматизованого тестування серверної частини системи MedVision було використано фреймворк xUnit, який є одним із найбільш поширених засобів модульного тестування для платформи .NET. Даний фреймворк дозволяє створювати автоматизовані тести для перевірки окремих компонентів програмної системи та контролювати коректність їх роботи після внесення змін до програмного коду.

Під час тестування використовувалася технологія Entity Framework Core InMemory Database, яка забезпечує створення тимчасової бази даних у пам'яті без необхідності підключення до реального сервера Microsoft SQL Server. Це дозволяє ізолювати тестове середовище від робочої бази даних та забезпечує швидке виконання тестів.

Також для створення тестового середовища використовувалася бібліотека Moq, призначена для імітації роботи зовнішніх залежностей програмних компонентів. Використання даного інструмента дозволяє виконувати перевірку окремих модулів системи незалежно від роботи інших компонентів.

У межах дипломного проєкту було реалізовано набір автоматизованих тестів для перевірки основних функціональних можливостей серверної частини системи. Тестування охоплювало механізми реєстрації та авторизації користувачів, отримання історії виконаних аналізів, перегляд інформації про конкретне дослідження та очищення історії аналізів.

Результати виконаних тестових сценаріїв наведено у таблиці 3.1.

Таблиця 3.1 — Виконання тестових сценаріїв

№	Тестовий сценарій	Очікуваний результат	Результат
1	Реєстрація нового лікаря	Створення облікового запису та успішна верифікація	Успішно
2	Реєстрація нового студента	Створення облікового запису та успішна верифікація	Успішно

Продовження таблиці 3.1

1	2	3	4
3	Реєстрація користувача з існуючою електронною адресою	Відмова у створені запису	Успішно
4	Реєстрація лікаря з некоректним номером посвідчення	Відмова в авторизації	Успішно
5	Реєстрація студента з некоректним номером студентського квитка	Відмова в авторизації	Успішно
6	Авторизація з коректними даними	Успішний вхід до системи	Успішно
7	Авторизація з неправильним паролем	Відмова в авторизації	Успішно
8	Отримання історії аналізів користувача	Повернення списку досліджень	Успішно
9	Отримання аналізу за ідентифікатором	Повернення інформації про дослідження	Успішно
10	Видалення аналізу власником	Успішне видалення запису	Успішно
11	Очищення історії аналізів	Видалення всіх записів	Успішно
12	Отримання неіснуючого аналізу	Повернення повідомлення про помилку	Успішно

Після реалізації тестів було виконано їх запуск у середовищі Visual Studio за допомогою вбудованого інструмента Test Explorer. Усі розроблені тести були успішно виконані, що підтверджує коректність роботи основних механізмів серверної частини системи. Результати виконання тестів наведено на рисунку 3.1.

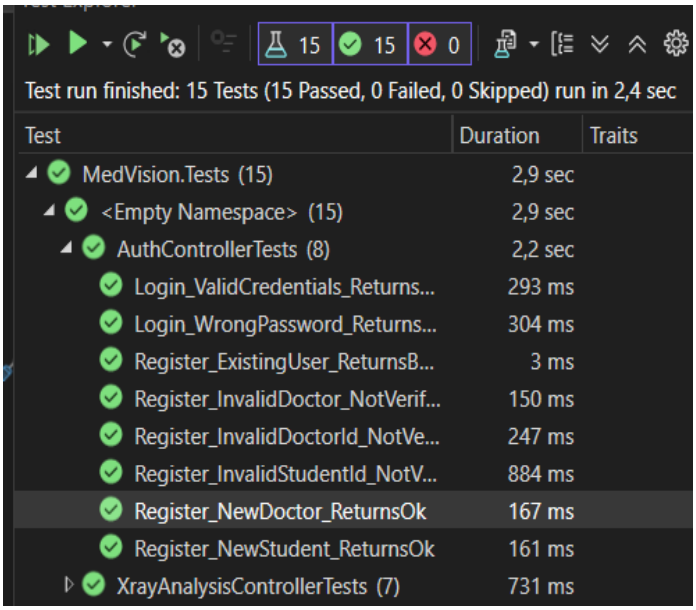

Рисунок 3.1 — Результати виконання тестів основної логіки системи у Visual Studio

Автоматизоване тестування дозволило перевірити правильність реалізації бізнес-логіки застосунку, коректність взаємодії з базою даних та обробки запитів користувачів. Отримані результати підтверджують стабільність роботи серверної частини системи та її готовність до подальшого використання у складі програмного комплексу MedVision.

3.2 Функціональне тестування системи

Функціональне тестування є одним із найбільш поширених видів тестування програмного забезпечення. Його основною метою є перевірка відповідності реалізованого функціоналу встановленим вимогам та підтвердження правильності роботи програмної системи з точки зору кінцевого користувача. Під час функціонального тестування перевіряються всі основні функції програмного продукту, коректність обробки вхідних даних та відповідність отриманих результатів очікуваній поведінці системи [10].

Тож, після завершення автоматизованого тестування серверної частини було проведено функціональне тестування системи MedVision. Основною метою даного етапу була перевірка працездатності реалізованих функціональних можливостей та оцінка правильності роботи системи в умовах, наближених до реальної експлуатації.

Для проведення функціонального тестування та верифікації розробленого програмного інтерфейсу використовувався програмний засіб Postman, який дозволяє формувати HTTP-запити до серверної частини системи та аналізувати отримані відповіді. У сучасній практиці інженерії програмного забезпечення Postman визначається як комплексна платформна екосистема для проєктування, тестування, документування та моніторингу інтерфейсів прикладного програмування (API) [11]. Завдяки підтримці широкого спектра протоколів та форматів даних (зокрема JSON та XML), даний інструмент забезпечує можливість

повноцінної симуляції дій клієнтської сторони без необхідності розгортання графічного інтерфейсу користувача.

У процесі функціонального тестування було перевірено роботу механізмів реєстрації та авторизації користувачів, верифікації лікарів і студентів, отримання персональної історії аналізів, перегляду інформації про конкретне дослідження, видалення окремих записів історії та очищення історії аналізів. Додатково було перевірено коректність роботи рольової моделі користувачів та механізму контролю доступу до даних.

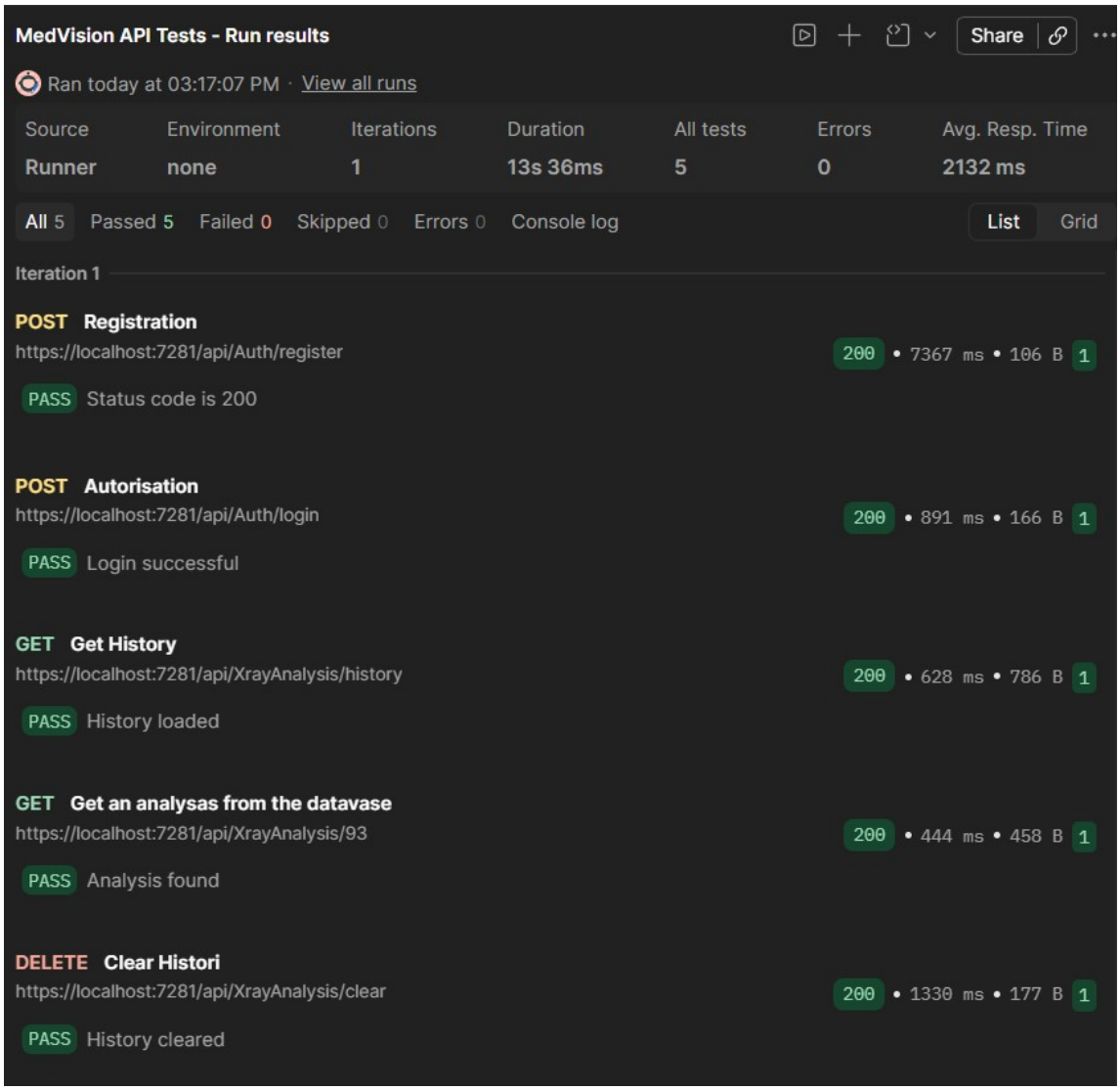

Рисунок 3.2 — Результати запуску колекції тестів у середовищі Postman

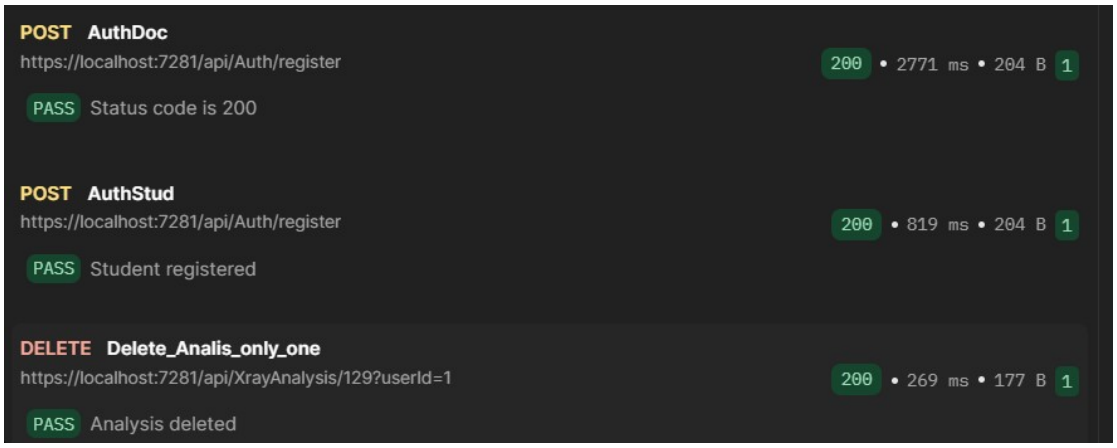

Рисунок 3.3 — Результати запуску колекції тестів у середовищі Postman(продовження)

Результати функціонального тестування показали, що всі перевірені програмні компоненти працюють відповідно до встановлених вимог. Усі тестові сценарії були виконані успішно, а серверна частина системи коректно обробляла HTTP-запити та повертала очікувані результати.

На рисунку 3.2 та рисунку 3.3 наведено результати виконання колекції функціональних тестів у середовищі Postman. Під час тестування перевірялися REST API методи реєстрації користувачів, авторизації, отримання історії аналізів, перегляду окремого дослідження та очищення історії аналізів. Усі запити повернули коректні HTTP-відповіді зі статусом 200 (OK), що підтверджує правильність роботи серверної частини системи.

Отримані результати свідчать про відповідність функціональних можливостей системи MedVision встановленим вимогам та підтверджують готовність програмного продукту до практичного використання.

3.3 Тестування модуля аналізу медичних зображень

Одним із ключових компонентів системи MedVision є модуль аналізу медичних зображень, реалізований на основі моделі штучного інтелекту YOLOv8. Оскільки від коректності його роботи залежить якість отриманих результатів,

важливим етапом розробки стало проведення тестування модуля аналізу рентгенівських знімків органів грудної клітки.

Тестування моделей машинного навчання має певні особливості порівняно з традиційним тестуванням програмного забезпечення. Якщо для звичайних програмних компонентів результат роботи за однакових вхідних даних є детермінованим, то ефективність моделей штучного інтелекту оцінюється за допомогою аналізу якості прогнозування на наборах даних, які не використовувалися під час навчання моделі. Основною метою такого тестування є перевірка здатності моделі правильно класифікувати об'єкти та виявляти закономірності на нових даних [13].

У рамках дипломного проєкту тестування проводилося на рентгенівських знімках органів грудної клітки, що містили як нормальні випадки, так і приклади з різними патологічними змінами. Для кожного дослідження система виконувала аналіз зображення, формувала висновок щодо наявності або відсутності патології, визначила рівень достовірності прогнозу та генерувала теплову карту (Heatmap), яка дозволяє локалізувати області зображення, що найбільше вплинули на результат класифікації.

Під час тестування особлива увага приділялася таким характеристикам роботи моделі:

- правильність визначення наявності або відсутності патологій;
- коректність класифікації патологічних змін;
- достовірність отриманих значень ймовірності прогнозу;
- формування теплової карти локалізації патологічних областей;
- швидкість виконання аналізу медичних зображень.

Для оцінки роботи системи було проведено аналіз результатів обробки декількох рентгенівських знімків із різними діагностичними випадками.

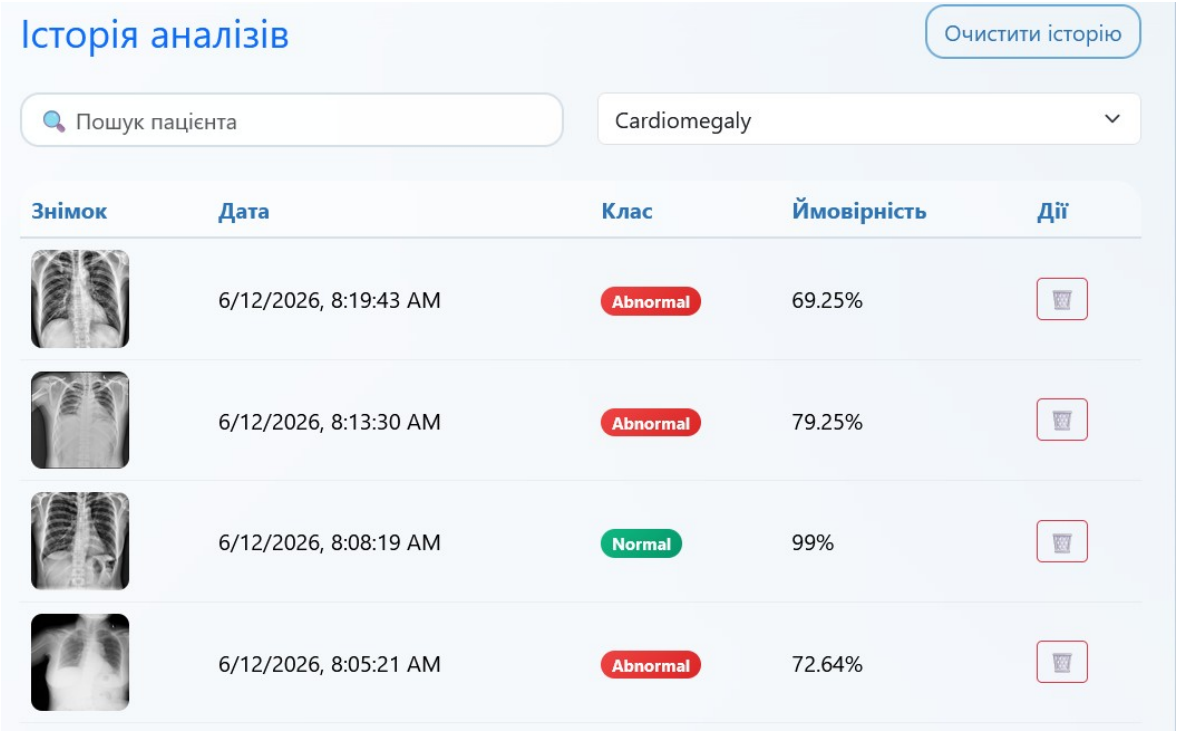

Рисунок 3.4 — Результати обробки рентгенівських знімків

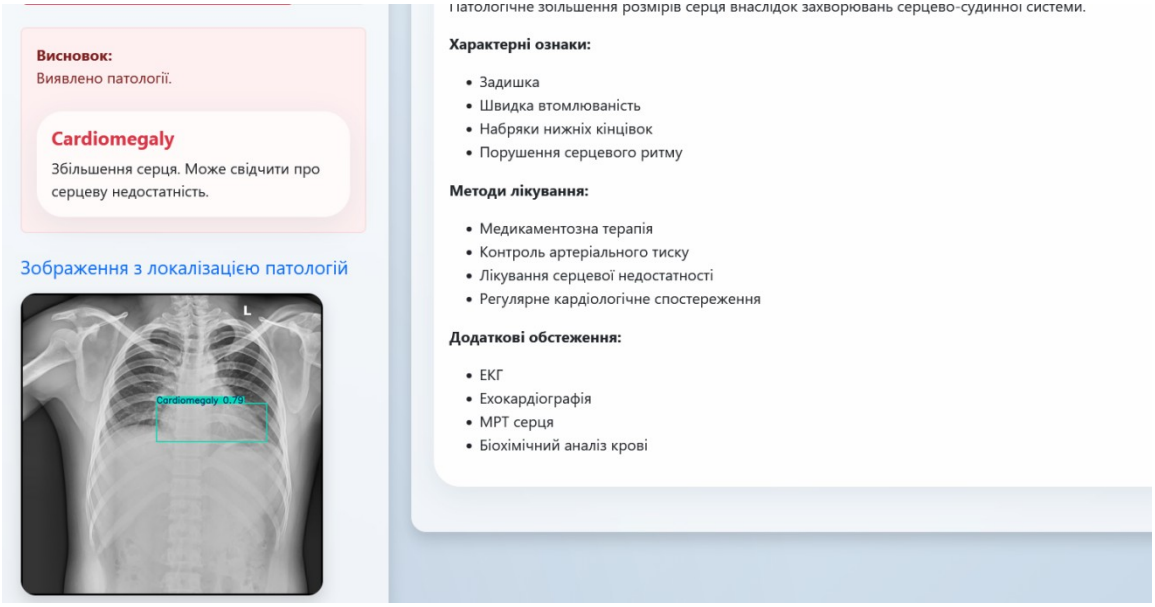

Рисунок 3.5 — Результати аналізу конкретного знімка легень

Результати тестування показали, що модель штучного інтелекту успішно визначає наявність патологічних змін на рентгенівських знімках органів грудної клітки та формує коректні результати класифікації. Для досліджених прикладів

прогноз моделі відповідав фактичному стану зображень, а значення ймовірності прогнозу перебували на достатньому рівні для використання системи як допоміжного інструменту підтримки прийняття рішень.

Крім класифікації знімків, система успішно формувала теплові карти локалізації патологічних областей та зображення з нанесеними межами виявлених патологій, LG підвищує інтерпретованість результатів роботи моделі для кінцевого користувача.

3.4 Впровадження системи MedVision

Після завершення розробки та тестування системи MedVision було виконано її розгортання та підготовку до практичного використання. Метою етапу впровадження є забезпечення працездатності всіх компонентів програмного комплексу в єдиному середовищі та перевірка можливості виконання повного циклу обробки медичних зображень.

Розроблена система реалізована за клієнт-серверною архітектурою та складається з чотирьох основних компонентів:

- клієнтської частини, реалізованої за допомогою бібліотеки React;
- серверної частини, створеної на платформі ASP.NET Core Web API;
- бази даних Microsoft SQL Server;
- модуля штучного інтелекту, реалізованого мовою Python із використанням моделі YOLOv8.

Взаємодія між компонентами системи здійснюється через HTTP-запити. Користувач працює з веб-інтерфейсом, який взаємодіє із серверною частиною. Сервер обробляє отримані запити, виконує операції з базою даних та передає рентгенівські знімки до сервісу штучного інтелекту для виконання аналізу.

Для розгортання системи використовувалося таке програмне забезпечення:

- операційна система Windows 11;
- Microsoft Visual Studio 2022;

- Visual Studio Code;
- Microsoft SQL Server;
- Node.js;
- Python версії 3.10;
- бібліотека Ultralytics YOLOv8;
- веб-сервер Kestrel, вбудований у платформу ASP.NET Core.

Перед запуском системи необхідно виконати налаштування бази даних та застосувати міграції Entity Framework Core. Після цього запускається серверна частина застосунку, модуль штучного інтелекту та клієнтський веб-застосунок.

Порядок запуску системи складається з таких етапів:

1. запуск сервера бази даних Microsoft SQL Server;
2. запуск серверної частини ASP.NET Core Web API;
3. запуск сервісу штучного інтелекту на основі Python та YOLOv8;
4. запуск клієнтського застосунку React;
5. відкриття веб-інтерфейсу системи у браузері.

Після виконання зазначених дій користувач отримує можливість пройти процедуру реєстрації, авторизації та верифікації, завантажити рентгенівський знімок, виконати його аналіз, переглянути історію досліджень та отримати результати роботи модуля штучного інтелекту.

Впровадження системи підтвердило коректність взаємодії між усіма компонентами програмного комплексу та можливість використання MedVision як інструмента підтримки прийняття рішень під час аналізу рентгенівських знімків органів грудної клітки.

3.5 Супровід та перспективи розвитку системи

Супровід та подальший розвиток ПЗ є важливим етапом його життєвого циклу. Основною метою супроводу є забезпечення стабільної роботи програмного продукту, усунення виявлених помилок, адаптація до змін середовища

експлуатації та розширення функціональних можливостей системи відповідно до потреб користувачів.

Розроблена система MedVision має модульну архітектуру, що спрощує процес її модернізації та підтримки. Використання сучасних технологій, зокрема React, ASP.NET Core, Microsoft SQL Server та Python, забезпечує можливість незалежного оновлення окремих компонентів програмного комплексу без необхідності суттєвої зміни всієї системи.

Під час супроводу системи можуть виконуватися такі роботи:

- оновлення програмних компонентів та бібліотек;
- виправлення виявлених помилок;
- оптимізація продуктивності серверної частини;
- підвищення точності моделей штучного інтелекту;
- розширення функціональних можливостей користувацького інтерфейсу;
- удосконалення механізмів безпеки та захисту даних.

На поточному етапі система підтримує реєстрацію, авторизацію та верифікацію користувачів, аналіз рентгенівських знімків легень, формування теплових карт локалізації патологій, перегляд історії досліджень, пошук і фільтрацію результатів аналізів, а також збереження інформації про пацієнтів. Проте архітектура системи дозволяє реалізувати низку додаткових функціональних можливостей у майбутніх версіях програмного продукту.

Перспективними напрямами розвитку системи є:

- інтеграція з відкритими державними реєстрами для автоматичної перевірки професійного статусу лікарів та студентів;
- розширення рольової моделі шляхом додавання адміністраторів, лікарів-рентгенологів та пацієнтів;
- підтримка додаткових типів медичних досліджень, зокрема комп'ютерної томографії та магнітно-резонансної томографії;

- розширення переліку патологій, які можуть бути виявлені моделлю штучного інтелекту;
- інтеграція із медичними інформаційними системами та електронними медичними картками пацієнтів;
- реалізація механізмів експорту результатів досліджень у форматах PDF та DICOM;
- розгортання системи у хмарному середовищі для забезпечення віддаленого доступу та підвищення масштабованості.

Окремим перспективним напрямом розвитку є вдосконалення механізмів верифікації користувачів. У майбутньому можлива інтеграція з інформаційними ресурсами Міністерства охорони здоров’я України та Єдиної державної електронної бази з питань освіти для автоматичного підтвердження статусу медичних працівників і студентів. Такий підхід дозволить підвищити рівень достовірності даних користувачів та забезпечити додатковий контроль доступу до функціональних можливостей системи.

Таким чином, розроблена система MedVision має достатній рівень функціональності для практичного використання та водночас володіє значним потенціалом для подальшого розвитку, удосконалення та інтеграції з сучасними медичними інформаційними технологіями.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У даному розділі кваліфікаційної роботи розглядаються теоретичні та практичні аспекти забезпечення нормативних умов праці, а також комплекс інженерно-технічних рішень, спрямованих на запобігання впливу небезпечних і шкідливих виробничих чинників під час розробки та експлуатації медичної вебплатформи MedVision. Проведено детальний аналіз взаємодії користувача з елементами обчислювальної техніки в межах системного підходу, визначено ключові ергономічні вимоги до організації робочого простору та обґрунтовано заходи безпеки розробленого програмного комплексу. Окрему увагу приділено питанням пожежної профілактики та технічним засобам захисту апаратного забезпечення від аварійних режимів роботи електромереж.

4.1. Безпека життєдіяльності

Дослідження умов життєдіяльності людини у процесі створення та впровадження сучасних програмних продуктів є невід'ємною частиною інженерного проєктування. Забезпечення безпеки життєдіяльності розробників та кінцевих користувачів медичної вебплатформи MedVision ґрунтується на аналізі потенційних загроз техногенного та психофізіологічного походження. У межах цього підходу розглядаються закономірності адаптації організму людини до умов тривалої роботи за комп'ютерною технікою, аналізуються чинники ризику виробничого середовища та розробляються методи їхньої мінімізації.

4.1.1. Характеристика життєдіяльності людини у системі «людина – комп’ютер – середовище існування»

Під час розробки, розгортання та експлуатації медичної вебплатформи MedVision ключовим аспектом є дослідження взаємодії суб’єкта (програміста,

ML-інженера, лікаря-рентгенолога) з технічними засобами в межах робочої зони. Дана взаємодія розглядається як функціонування трикомпонентної системи «людина – комп’ютер – середовище існування» (Л–К–СІ), де технічним ядром виступає персональний комп’ютер (робоча станція) та серверна інфраструктура [14]

Прямі та зворотні зв’язки у досліджуваній системі мають таку специфіку:

1. Канал «Людина – Комп’ютер»: Здійснюється через введення керуючих команд за допомогою клавіатури, миші та графічних маніпуляторів. Навантаження припадає на нервово-м’язову систему кистей рук, суглоби та вимагає високої координації рухів і точної дрібної моторики [15]. При тривалій роботі виникає ризик розвитку тунельного синдрому (синдрому зап'ястного каналу).
2. Канал «Комп’ютер – Людина»: Візуальне відображення інформації (інтерфейсу React, медичних знімків, результатів сегментації нейромережі) на рідкокристалічному моніторі. Створює інтенсивне навантаження на зоровий аналізатор та центральну нервову систему [15]. Тривале статичне сприйняття графічної інформації високої чіткості викликає синдром комп'ютерного зору (сухість, почервоніння, спазм акомодації).
3. Канал «Середовище існування – Людина»: Стан мікроклімату (температура, вологість), рівень шуму від кулерів серверів та систем охолодження, штучна й природна освітленість, наявність електромагнітних випромінювань безпосередньо впливають на загальне самопочуття, швидкість реакції та працездатність оператора [14].

У процесі експлуатації системи Л–К–СІ виділяються такі потенційні небезпечні та шкідливі чинники, які класифікуються на дві основні групи:

Психофізіологічні чинники:

- Значне розумове напруження, пов'язане з аналізом складних алгоритмів та архітектури нейромереж;
- Нервово-емоційне напруження під час діагностики та верифікації

критичних медичних знімків;

- Тривала статична гіподинамія через тривале перебування користувача в сидячому положенні;
- Вимушена робоча поза, яка призводить до нерівномірного навантаження на хребет і м'язи спини;
- Монотонність праці під час рутинного тестування програмного коду або розмітки великих медичних датасетів для навчання моделей [15].

Фізичні чинники:

- Невідповідність параметрів мікроклімату нормативним показникам (перегрів приміщення технікою);
- Недостатнє, надмірне або нерівномірне штучне освітлення робочих поверхонь;
- Підвищений рівень шуму, що генерується обчислювальною технікою, кондиціонерами та серверами;
- Електромагнітне випромінювання промислової частоти (50 Гц) від ліній живлення та блоків потужності [14].

З метою мінімізації негативного впливу наведених чинників у проєкті впроваджено комплексний захист: раціональний режим праці та відпочинку (регламентовані перерви по 10 хвилин через кожну годину інтенсивної роботи з монітором) [16], використання ергономічного меблевого обладнання та постійна підтримка нормативних параметрів виробничого середовища відповідно до вимог чинного законодавства [17].

4.2. Основи охорони праці

Охорона праці є системою правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності. Під час експлуатації розробленого програмного забезпечення

та супутньої серверної інфраструктури правове й технічне регулювання безпеки виконується відповідно до чинних державних нормативно-правових актів України (НПАОП, ДБН, ДСН). Практична реалізація цих вимог передбачає створення оптимальних гігієнічних умов на робочих місцях операторів та інженерів, а також проєктування надійних систем захисту від ураження електричним струмом і попередження пожежних аварійних ситуацій.

4.2.1. Загальні вимоги безпеки з охорони праці та ергономічні вимоги до організації робочого місця оператора

Організація робочого місця інженера-програміста та медичного персоналу, які взаємодіють із платформою MedVision, виконується відповідно до вимог НПАОП 0.00-7.15-18 «Вимоги безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [16].

Основні гігієнічні та ергономічні рішення, що впроваджуються на робочому місці, включають такі деталізовані параметри:

1. Паспортні параметри виробничого приміщення: На одне робоче місце оператора передбачено вільну площу не менше 6,0 м2 та загальний об’єм приміщення не менше 20,0 м3 [16]. Поверхня підлоги в приміщенні обчислювального центру виконана з антистатичних матеріалів (спеціальний лінолеум із заземленим мідним контуром), що повністю унеможливлює накопичення статичного заряду, захищаючи як персонал від неприємних мікроударів, так і внутрішні плати ПК від пробоїв [15].

2. Геометричні параметри робочого столу та стільця: Висота робочої поверхні столу встановлена у фіксованому межах 725 мм [15]. Ширина столу становить не менше 1200 мм, а глибина — 800 мм, що дозволяє вільно розмістити периферійні пристрої. Конструкція робочого стільця (крісла) забезпечує підтримку раціональної анатомічної робочої поза за рахунок системи плавного регулювання висоти сидіння (в межах 400–550 мм), кута нахилу спинки (від 90°

до 110° відносно горизонталі) та висоти підлокітників. Спинка стільця має ергономічний вигин у поперековій зоні для зниження компресійного навантаження на хребет.

3. Технічні вимоги до монітора (відеотерміналу): Екран відеотерміналу розміщується на оптимальній відстані 600–700 мм від очей користувача, що мінімізує напругу м'язів кришталика. Кут зору до центру екрана становить 15–20° нижче горизонтальної лінії очей [16]. Частота оновлення екрана встановлена на рівні не менше 75–120 Гц для повного виключення флікер-ефекту (видимого чи прихованого мерехтіння матриці), що є критично важливим для точного аналізу дрібних деталей медичних знімків лікарем-рентгенологом.

4. Розрахунок та вимоги до виробничого освітлення: Комбіноване освітлення (природне та штучне) забезпечує рівень загальної освітленості на робочому столі не менше 300–500 лк відповідно до ДБН В.2.5-28:2018 «Природне і штучне освітлення» [18]. Джерела штучного світла реалізовані у вигляді світлодіодних панелей із розсіювачами, які розташовуються рядами паралельно до лінії погляду оператора. Це унеможливлює утворення стробоскопічного ефекту. Віконні отвори приміщення обов'язково обладнані щільними регульованими жалюзі для запобігання появі прямих або відбитих бліків на екрані монітора.

Для контролю штучного освітлення використовується інженерна формула розрахунку світлового потоку методом коефіцієнта використання:

$$F = \frac{E \cdot S \cdot k \cdot z}{N \cdot \eta} \tag{4.1}$$

де E — нормована освітленість (300 лк); S — площа приміщення, м2; k — коефіцієнт запасу (1,4 для світлодіодів); z — коефіцієнт нерівномірності (1,1); N — кількість світильників; η — коефіцієнт використання світлового потоку.

5. Організація та параметри мікроклімату: Нормативні кліматичні умови забезпечуються за рахунок припливно-витяжної систем вентиляції та локальних систем кондиціонування повітря. Відповідно до Санітарних норм ДСН 3.3.6.042-

99 для категорії робіт Іа (легка фізична праця, що виконується сидячи) встановлено такі жорсткі обмеження для теплого періоду року [17]:

- Температура повітря в робочій зоні: 22–25 °С;
- Відносна вологість повітря: 40–60%;
- Швидкість руху повітряних потоків: не більше 0,1 м/с.

Необхідний об'єм подачі свіжого повітря на одного працівника за відсутності природного провітрювання становить не менше 60 м3/год.

4.2.2. Захист електрообладнання від короткого замикання та перенавантаження

Приміщення обчислювального центру, офіси та робочі кабінети IT-розробників належать до категорій приміщень без підвищеної небезпеки ураження електричним струмом (сухі, з нормальною температурою, з ізольованою підлогою та без пилу). Проте висока щільність розміщення серверного обладнання, комутаторів, блоків живлення та потужних робочих станцій значно підвищує ризики виникнення аварійних режимів (коротких замикань та тривалих струмових перенавантажень) у трифазних та однофазних електричних мережах напругою до 1000 В [14].

Електробезпека, пожежна профілактика та комплексний захист апаратної частини системи MedVision забезпечуються реалізацією таких інженерно-технічних рішень:

1. Захисне занулення (система TN-S): Вся трипровідна (L, N, PE) та п'ятипровідна мережа будівлі реалізована із чітким розділенням робочого нуля (N) та захисного нуля (PE). Усі металеві неструмопровідні частини обладнання (корпуси системних блоків, металеві серверні шафи, стійки, зовнішні оболонки кондиціонерів, лотки кабельних трас), які можуть опинитися під небезпечною напругою внаслідок пошкодження або деструкції ізоляції, надійно приєднані до нульового захисного провідника (PE) [14]. При пробої фази на корпус система

забезпечує миттєве виникнення струму однофазного короткого замикання, що веде до швидкого відключення аварійної ділянки.

2. Автоматичні апарати захисту від аварійних режимів: Лінії живлення комп'ютерної техніки та розеткові групи захищені сучасними автоматичними вимикачами з комбінованими розчеплювачами згідно з вимогами НАПБ А.01.001-2014 [19]. Захист складається з двох контурів:

- Тепловий розчеплювач (біметалева пластина) здійснює захист від тривалого перенавантаження мережі, яке виникає при одночасному підключенні надмірної кількості техніки;
- Електромагнітний розчеплювач (соленоїд) здійснює миттєву відсічку струмів короткого замикання. Для офісних та лабораторних ліній живлення використовуються автомати з характеристикою типу «В» або «С».

3. Пристрої захисного вимкнення (ПЗВ) та диференційний захист: Для безпосереднього захисту людей від ураження струмом при прямому випадковому дотику до відкритих струмопровідних частин або при витоках струму через пошкоджену ізоляцію, розеткові групи обладнані ПЗВ або диференційними автоматами. Струм спрацювання (уставка) дифзахисту становить не більше ≤30 мА, а повний час автоматичного відключення мережі не перевищує 0,1 с [14]. Це значення є безпечним для життя людини, оскільки фібриляція серця за такий час не встигає виникнути.

4. Захист від імпульсних перенапруг та архітектура безперебійного живлення: Для запобігання втрати критичних медичних даних пацієнтів, збереження логів та попередження пошкодження дорогих моделей штучного інтелекту (YOLOv8) при різких стрибках напруги, вся серверна інфраструктура підключена за схемою подвійного перетворення через джерела безперебійного живлення (UPS) он-лайн (On-Line) типу [15]. Вони забезпечують нульовий час перемикання на акумулятори. Вхідні щити обладнані пристроями захисту від імпульсних перенапруг (ПЗІП) класу II та III для нівелювання наслідків грозових розрядів та комутаційних сплесків в електромережі міста.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено інформаційну систему MedVision, призначену для автоматизованого аналізу рентгенівських знімків органів грудної клітки із використанням технологій штучного інтелекту. Основною метою роботи було створення програмного продукту, який забезпечує завантаження медичних зображень, їх автоматичну обробку за допомогою моделі машинного навчання, збереження результатів досліджень та надання користувачеві зручного інтерфейсу для роботи з отриманими даними.

У процесі виконання роботи було проведено аналіз предметної області та сучасних підходів до використання технологій штучного інтелекту в медицині. На основі отриманих результатів було сформульовано функціональні та нефункціональні вимоги до програмної системи, визначено архітектуру програмного комплексу та обрано технології для його реалізації.

Також, у ході проєктування було розроблено структурну схему системи, діаграми варіантів використання та послідовностей, а також спроєктовано базу даних для зберігання інформації про користувачів та результати виконаних досліджень. Реалізована структура бази даних забезпечує підтримку механізмів автентифікації користувачів, збереження історії аналізів та зберігання інформації про пацієнтів.

Серверну частину системи реалізовано на платформі ASP.NET Core Web API із використанням технології Entity Framework Core та системи керування базами даних Microsoft SQL Server. Для клієнтської частини було використано бібліотеку React, що дозволило створити сучасний веб-інтерфейс із підтримкою інтерактивної взаємодії користувача із системою. Модуль аналізу медичних зображень реалізовано мовою Python із використанням моделі штучного інтелекту YOLOv8.

Ще було реалізовано механізми реєстрації, авторизації та верифікації користувачів із підтримкою рольової моделі доступу. Система підтримує роботу

двох категорій користувачів — лікарів та студентів. Для підтвердження статусу користувача реалізовано механізм перевірки номерів посвідчень та студентських квитків із використанням внутрішнього реєстру верифікації. Реалізовано функціональні можливості збереження інформації про пацієнтів, перегляду історії досліджень, пошуку та фільтрації результатів аналізу, перегляду детальної інформації про виконані дослідження та видалення окремих записів історії.

Для перевірки працездатності системи було проведено автоматизоване та функціональне тестування. Автоматизовані тести реалізовано за допомогою фреймворку xUnit із використанням технології Entity Framework Core InMemory Database та бібліотеки Moq. Результати тестування підтвердили коректність роботи механізмів реєстрації, авторизації, верифікації користувачів, роботи з історією аналізів та контролю доступу до даних. Додатково було виконано функціональне тестування API за допомогою інструмента Postman.

Належну увагу було приділено тестуванню модуля штучного інтелекту. Проведене тестування показало можливість успішного виявлення патологічних змін на рентгенівських знімках органів грудної клітки, формування текстового висновку, теплових карт локалізації патологічних областей та зображень із позначенням виявлених змін. Отримані результати підтвердили можливість використання системи як допоміжного інструменту підтримки прийняття рішень під час аналізу медичних зображень.

Розроблена система MedVision успішно пройшла етапи проєктування, реалізації, тестування та впровадження. Отримані результати свідчать про досягнення поставленої мети дипломного проєкту та підтверджують можливість практичного використання розробленого програмного продукту. Подальший розвиток системи може бути пов'язаний із розширенням функціональних можливостей, інтеграцією з медичними інформаційними системами, використанням більш точних моделей штучного інтелекту та впровадженням механізмів автоматичної верифікації користувачів через державні інформаційні ресурси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рентгенологічні дослідження. [Електронний ресурс] URL: https://viasan.ua/ua/rentgen

2. Сучасні досягнення ШІ для радіологічної діагностики. [Електронний ресурс] URL: https://radiolance.ua/suchasni-dosiagnennia-shi-dlia-radiologichnoyi-diagnostyky/

3. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. [Електронний ресурс] URL: https://stanfordmlgroup.github.io/projects/chexnet/

4. NIH Chest X-rays. [Електронний ресурс] URL: https://www.kaggle.com/datasets/nih-chest-xrays/data

5. Що таке комп'ютерний зір: пояснення, принцип роботи та застосування. [Електронний ресурс] URL: https://university.sigma.software/what-is-computer-vision/

6. CNN та її застосування. [Електронний ресурс] URL: https://evergreens.com.ua/ua/articles/cnn.html

7. Шульгін О. Я., Штовба С. Д. Огляд можливостей AI фреймворку YOLOv8 для вирішення задач розпізнавання об'єктів на зображеннях. Збірник наукових праць. 2024. С. 1–2. [Електронний ресурс] URL: [https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/download/19819/16612&ved=2ahUKEwif5puioPiUAxVvR_EDHSNCC80QFnoECBgQAQ&usg=AOvVaw1wL833MyK6tkennxg41PoY] (дата звернення: 08.06.2026).

8. VinBigData Chest X-ray Abnormalities Detection. [Електронний ресурс] URL: https://www.kaggle.com/competitions/vinbigdata-chest-xray-abnormalities-detection

9. Ручне і автоматизоване тестування. [Електронний ресурс] URL: https://www.academy4heroes.com/novyny/ruchne-i-avtomatyzovane-testuvannya/

10. Функціональне тестування. [Електронний ресурс] URL: https://qalight.ua/baza-znaniy/funktsionalne-testuvannya/

11. Використання Postman в тестуванні. [Електронний ресурс] URL: https://training.qatestlab.com/blog/technical-articles/use-postman-in-testing/

12. Postman API Platform: Create, test, and share APIs. Postman Learning Center. [Електронний ресурс] URL: https://learning.postman.com/docs/getting-started/overview/

13. Стратегії тестування моделей машинного навчання [Електронний ресурс] URL: https://ua.linkedin.com/pulse/machine-learning-models-testing-strategies-testrigor-0g6xe?tl=uk

14. Безпека життєдіяльності та охорона праці : підруч. / В. В. Сокуренко, О. М. Бандурка та ін. Харків : ХНУВС, 2021. 308 с.

15. Жидецький В. Ц. Охорона праці користувачів комп’ютерів : підручник. Львів : Афіша, 2020. 176 с.

16. НПАОП 0.00-7.15-18. Вимоги безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Затверджено Наказом Міністерства соціальної політики України 14.02.2018 № 207. Київ, 2018. 14 с.

17. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень : Державні санітарні норми України. Міністерство охорони здоров'я України. Київ, 1999. 18 с.

18. ДБН В.2.5-28:2018. Природне і штучне освітлення. Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України. Київ : ДП «Укрархбудінформ», 2018. 104 с.

19. НАПБ А.01.001-2014. Правила пожежної безпеки в Україні : Затверджено Наказом Міністерства внутрішніх справ України 30.12.2014 № 1417. Київ, 2015. 98 с.

20. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М. – Тернопіль:

Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 45 с. Відповідальний за випуск: д.ф.-м.н., проф. Петрик М.Р. URL: https://elartu.tntu.edu.ua/handle/lib/50317

ДОДАТКИ

ДОДАТОК А

Інтерфейс системи MedVision

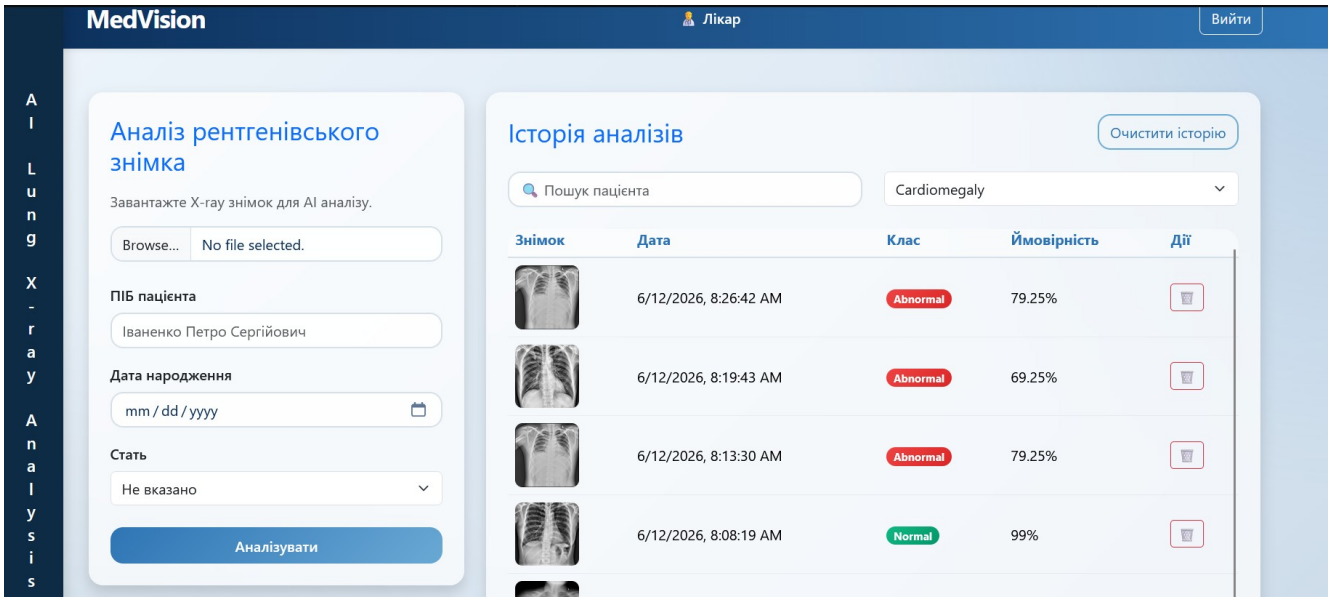

Рисунок А.1 — Головна сторінка веб-застосунку MedVision

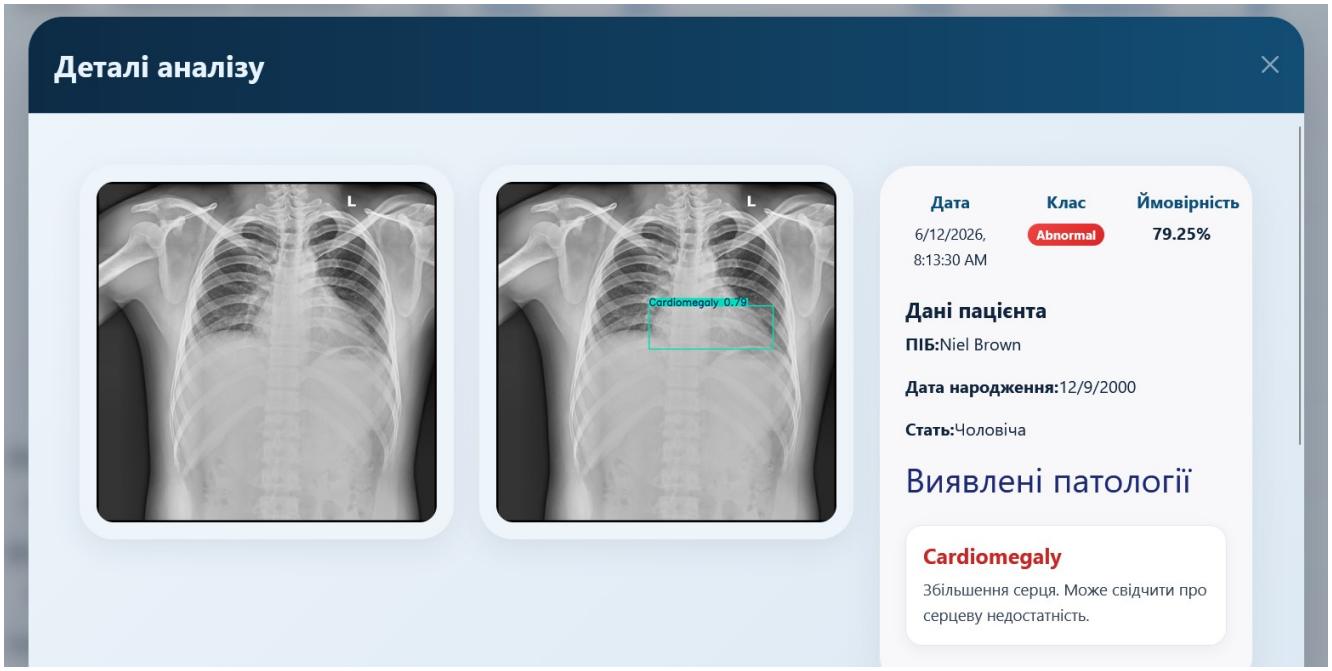

Рисунок А.2 — Сторінка деталей аналізу рентгенівського знімка з історії аналізів

Рисунок А.3 — Підготування даних для аналізу знімка в системі

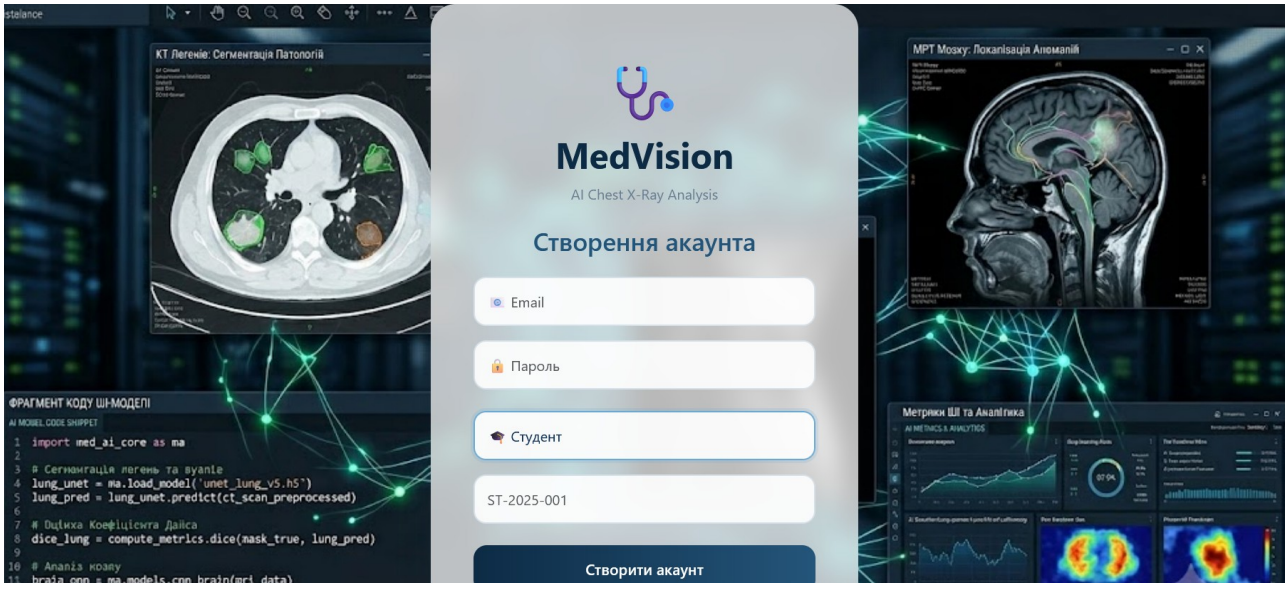

Рисунок А.4 — Реєстрація користувача в системі MedVision

ДОДАТОК Б

Код модуля штучного інтелекту

```
from fastapi import FastAPI, UploadFile, File
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from fastapi.responses import FileResponse
from ultralytics import YOLO

import os
import uuid
import shutil
import cv2
import pydicom
import numpy as np

app = FastAPI(title="MedVision AI Service")



# ==========================
# CORS
# ==========================

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



# ==========================
# LOAD YOLO MODEL
# ==========================
```

```
model = YOLO("models/medvision_yolo.pt")


CONFIDENCE_THRESHOLD = 0.5
DICOM_CONFIDENCE_THRESHOLD = 0.3

PATHOLOGY_INFO = {
    "Cardiomegaly": "Збільшення серця. Може свідчити про серцеву недостатність.",
    "Aortic enlargement": "Розширення аорти. Потребує додаткового обстеження.",
    "Pleural effusion": "Наявність рідини у плевральній порожнині.",
    "Nodule/Mass": "Можливе новоутворення в легенях.",
    "Pneumothorax": "Наявність повітря у плевральній порожнині.",
    "Lung Opacity": "Виявлено затемнення легеневої тканини.",
    "Consolidation": "Можлива пневмонія або запальний процес.",
    "Infiltration": "Інфільтративні зміни у легенях.",
    "Pulmonary fibrosis": "Фіброзні зміни легеневої тканини.",
    "Pleural thickening": "Потовщення плеври.",
    "Calcification": "Кальциновані ділянки у тканинах.",
    "Atelectasis": "Часткове спадіння легені.",
    "ILD": "Інтерстиціальне захворювання легень.",
    "Other lesion": "Інші патологічні зміни.",
}


# ========================
# CREATE OUTPUT FOLDER
# ========================

OUTPUT_FOLDER = "predictions"

os.makedirs(OUTPUT_FOLDER, exist_ok=True)
```

```
app.mount(
    "/predictions",
    StaticFiles(directory=OUTPUT_FOLDER),
    name="predictions",
)


# =========================
# HELPERS
# =========================


def convert_dicom_to_png(file_path: str) -> str:

    dicom = pydicom.dcmread(file_path)

    image = dicom.pixel_array.astype(np.float32)

    # =========================
    # NORMALIZATION
    # =========================

    image = image - np.min(image)

    image = image / np.max(image)

    image = (image * 255).astype(np.uint8)

    # =========================
    # CLAHE CONTRAST ENHANCEMENT
    # =========================

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

    image = clahe.apply(image)
```

```
    # ========================
    # GAUSSIAN DENOISING
    # ========================

    image = cv2.GaussianBlur(image, (3, 3), 0)

    converted_path = os.path.splitext(file_path)[0] + ".png"

    cv2.imwrite(converted_path, image)

    return converted_path


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


@app.get("/")
def health_check():
    return {"message": "MedVision YOLO AI Service is running"}


@app.post("/preview")
async def preview(file: UploadFile = File(...)):

    extension = os.path.splitext(file.filename)[1].lower()

    unique_filename = f"preview_{uuid.uuid4()}{extension}"

    temp_path = os.path.join(OUTPUT_FOLDER, unique_filename)

    with open(temp_path, "wb") as buffer:
        shutil.copyfileobj(file.file, buffer)
```

```
        if extension in [".dcm", ".dicom"]:

            png_path = convert_dicom_to_png(temp_path)

            return FileResponse(png_path, media_type="image/png")

        return FileResponse(temp_path)


    @app.post("/predict")
    async def predict(file: UploadFile = File(...)):
        extension = os.path.splitext(file.filename)[1].lower()

        unique_filename = f"{uuid.uuid4()}{extension}"

        temp_path = os.path.join(OUTPUT_FOLDER, unique_filename)

        with open(temp_path, "wb") as buffer:
            shutil.copyfileobj(file.file, buffer)

        is_dicom = extension in [".dcm", ".dicom"]

        preview_path = None

        if is_dicom:
            temp_path = convert_dicom_to_png(temp_path)

            preview_path =
f"/predictions/{os.path.basename(temp_path)}"

        current_threshold = DICOM_CONFIDENCE_THRESHOLD if is_dicom
else CONFIDENCE_THRESHOLD

        results = model(temp_path, conf=current_threshold)

        result = results[0]
```

```
        plotted_image = result.plot()

        predicted_filename = f"pred_{uuid.uuid4()}.jpg"

        predicted_path = os.path.join(OUTPUT_FOLDER,
predicted_filename)

        cv2.imwrite(predicted_path, plotted_image)

        detections = []

        boxes = result.boxes

        if boxes is not None:
            for box in boxes:

                class_id = int(box.cls[0])

                confidence = float(box.conf[0])

                if confidence < current_threshold:
                    continue

                class_name = model.names[class_id]

                detections.append(
                    {
                "className": class_name,
                        "confidence": round(confidence * 100, 2),
                        "description": PATHOLOGY_INFO.get(
                            class_name, "Опис патології відсутній."
                        ),
                    }
                )
```

```
        return {
            "resultClass": "Abnormal" if len(detections) > 0 else
"Normal",
            "probability": max(
                [d["confidence"] for d in detections],
                default=99.0,
            ),
            "conclusion": (
                "Виявлено патології." if len(detections) > 0 else
"Патологій не виявлено."
            ),
            "detections": detections,
            "heatmapPath": f"/predictions/{predicted_filename}",
            "previewPath": preview_path,
        }
```

ДОДАТОК В

Опис REST API системи

```
using MedVision.Api.Data;
using MedVision.Api.DTOs;
using MedVision.Api.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MedVision.Api.DTOs;

namespace MedVision.Api.Controllers;

[Route("api/[controller]")]
[ApiController]
public class XrayAnalysisController : ControllerBase
{
    private readonly AppDbContext _context;
    private readonly IWebHostEnvironment _environment;
    private readonly IHttpClientFactory _httpClientFactory;

    public XrayAnalysisController(
        AppDbContext context,
        IWebHostEnvironment environment,
        IHttpClientFactory httpClientFactory)
    {
        _context = context;
        _environment = environment;
        _httpClientFactory = httpClientFactory;
    }

    [HttpPost("upload")]
    public async Task<ActionResult<XrayAnalysisResponseDto>>
UploadXray(IFormFile file,

    [FromForm] int userId,
```

```
            [FromForm] string? patientFullName,
            [FromForm] DateTime? patientBirthDate,
            [FromForm] string? patientGender)
            {
                if (file == null || file.Length == 0)
                {
                    return BadRequest("Файл не було завантажено.");
                }

                var allowedExtensions = new[]
{ ".jpg",".jpeg",".png",".dcm",".dicom"};


                var extension =
Path.GetExtension(file.FileName).ToLower();

                if (!allowedExtensions.Contains(extension))
                {
                    return BadRequest("Дозволені лише файли JPG, JPEG,
PNG або DICOM.");
                }

                var uploadsFolder =
Path.Combine(_environment.WebRootPath, "uploads");

                if (!Directory.Exists(uploadsFolder))
                {
                    Directory.CreateDirectory(uploadsFolder);
                }

                var uniqueFileName = $"{Guid.NewGuid()}{extension}";
                var filePath = Path.Combine(uploadsFolder,
uniqueFileName);

                await using (var stream = new FileStream(filePath,
FileMode.Create))
```

```
            {
                await file.CopyToAsync(stream);
            }

            var client = _httpClientFactory.CreateClient();

            await using var imageStream =
System.IO.File.OpenRead(filePath);

            using var content = new MultipartFormDataContent();
            using var fileContent = new StreamContent(imageStream);

            fileContent.Headers.ContentType =
                new
System.Net.Http.Headers.MediaTypeHeaderValue(file.ContentType);

            content.Add(fileContent, "file", uniqueFileName);

            var aiResponse = await client.PostAsync(
                "http://127.0.0.1:8000/predict",
                content);

            if (!aiResponse.IsSuccessStatusCode)
            {
                return StatusCode(500, "Помилка під час звернення
до AI-сервісу.");
            }

            var prediction = await
aiResponse.Content.ReadFromJsonAsync<AiPredictionResponse>();

            if (prediction == null)
            {
                return StatusCode(500, "AI-сервіс не повернув
результат.");
            }
```

```
var probability = prediction.Probability;
var resultClass = prediction.ResultClass;
var conclusion = prediction.Conclusion;

var detectionsJson =
System.Text.Json.JsonSerializer.Serialize(
prediction.Detections );

var analysis = new XrayAnalysis
{
    ImagePath = $"/uploads/{uniqueFileName}",
    ResultClass = resultClass,
    Probability = probability,
    Conclusion = conclusion,
    CreatedAt = DateTime.UtcNow,
    HeatmapPath = prediction.HeatmapPath,
    DetectionsJson = detectionsJson,
    PreviewPath = prediction.PreviewPath,
    UserId = userId,
    PatientFullName = patientFullName,

    PatientBirthDate = patientBirthDate,

    PatientGender = patientGender,
};

_context.XrayAnalyses.Add(analysis);
await _context.SaveChangesAsync();

var response = new XrayAnalysisResponseDto
{
    Id = analysis.Id,

    ImagePath = analysis.ImagePath,
```

```
                ResultClass = analysis.ResultClass,

                Probability = analysis.Probability,

                Conclusion = analysis.Conclusion,

                CreatedAt = analysis.CreatedAt,

                HeatmapPath = analysis.HeatmapPath,

                DetectionsJson = analysis.DetectionsJson,
                PreviewPath = analysis.PreviewPath,
                Detections = prediction.Detections

            };

            return Ok(response);
        }

        [HttpGet("history/{userId}")]
        public async
Task<ActionResult<IEnumerable<XrayAnalysisResponseDto>>> GetHistory(
          int userId)
        {
            var analyses = await _context.XrayAnalyses
                .Where(x => x.UserId == userId)
                .OrderByDescending(x => x.CreatedAt)
                .Select(x => new XrayAnalysisResponseDto
                {
                    Id = x.Id,
                    ImagePath = x.ImagePath,
                    ResultClass = x.ResultClass,
                    Probability = x.Probability,
                    Conclusion = x.Conclusion,
                    CreatedAt = x.CreatedAt,
                    HeatmapPath = x.HeatmapPath,
```

```
                    DetectionsJson = x.DetectionsJson,
                    PreviewPath = x.PreviewPath,

                    PatientFullName = x.PatientFullName,
                    PatientBirthDate = x.PatientBirthDate,
                    PatientGender = x.PatientGender
                })
                .ToListAsync();

            return Ok(analyses);
        }

        [HttpDelete("clear")]
        public async Task<IActionResult> ClearHistory()
        {
            var analyses = await
_context.XrayAnalyses.ToListAsync();

            _context.XrayAnalyses.RemoveRange(analyses);

            await _context.SaveChangesAsync();

            return Ok(new
            {
                message = "Історію очищено"
            });
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<XrayAnalysisResponseDto>>
GetById(int id)
        {
            var analysis = await
_context.XrayAnalyses.FindAsync(id);

            if (analysis == null)
```

```
        {
            return NotFound("Аналіз не знайдено.");
        }

        var response = new XrayAnalysisResponseDto
        {
            Id = analysis.Id,
            ImagePath = analysis.ImagePath,
            ResultClass = analysis.ResultClass,
            Probability = analysis.Probability,
            Conclusion = analysis.Conclusion,
            CreatedAt = analysis.CreatedAt,
            HeatmapPath = analysis.HeatmapPath,
            DetectionsJson = analysis.DetectionsJson,

            PreviewPath = analysis.PreviewPath,

            PatientFullName = analysis.PatientFullName,
            PatientBirthDate = analysis.PatientBirthDate,
            PatientGender = analysis.PatientGender,
        };

        return Ok(response);
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteAnalysis(
     int id,
     [FromQuery] int userId)
    {
        var analysis = await _context.XrayAnalyses
            .FindAsync(id);

        if (analysis == null)
        {
            return NotFound();
```

```
        }

        if (analysis.UserId != userId)
        {
            return Forbid();
        }

        _context.XrayAnalyses.Remove(analysis);

        await _context.SaveChangesAsync();

        return Ok(new
        {
            message = "Аналіз видалено"
        });
    }
}
```

ДОДАТОК Г

Посилання на репозиторій GitHub

Рисунок Г.1 — QR-код для репозиторію на GitHub для системи MedVision