

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

_____ **Ігор БОЛБОТ**
(підпис)

«___» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук

_____ **Белла ГОЛУБ**
(підпис)

«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення системи класифікації медичних зображень з використанням глибокого навчання»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н., доцент

_____ (підпис)

Ганна ВАЙГАНГ

**Керівник бакалаврської
кваліфікаційної роботи**

к.т.н., с.н.с.

_____ (підпис)

Світлана ЗАЙЦЕВА

Виконав

_____ (підпис)

Дмитро ШТЕПА

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____

(підпис)

Белла ГОЛУБ

«___» _____ 2025 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Штепі Дмитру Юрійовичу

(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення системи класифікації медичних зображень з використанням глибокого навчання»

затверджена наказом від «19» грудня 2025 р. №3204 «С»

Термін подання завершеної роботи на кафедру «22 травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

Перелік питань, що підлягають дослідженню:

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025 р.

**Керівник бакалаврської
кваліфікаційної роботи**

(підпис)

Світлана ЗАЙЦЕВА

**Завдання взяв до
виконання**

(підпис)

Дмитро ШТЕПА

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	5
ВСТУП.....	7
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1 Опис предметної області.....	10
1.2 Аналіз існуючих рішень.....	12
1.3 Моделювання програмної системи.....	18
1.4 Аналіз вимог розроблювальної системи	22
1.5 Постановка завдання	24
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	26
2.1 Загальна характеристика проєктованої системи	26
2.2 Діаграма класів і кооперації системи	27
2.3 Пакетна та компонентна структура системи	30
2.4 Логічна та фізична модель даних.....	33
2.5 Діаграма розгортання системи	35
2.6 Проєктування алгоритмів роботи системи	37
2.7 Висновки до другого розділу	41
3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	42
3.1 Обґрунтування вибору технологічних засобів	42
3.2 Реалізація інформаційної бази	43
3.3 Реалізація інтерфейсу користувача.....	44
3.4 Реалізація основних програмних функцій	49
3.5 Висновки до третього розділу	51

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ.....	52
4.1 План тестування програмних модулів.....	52
4.2 Тестування інтерфейсу користувача	53
4.3 Тестування алгоритму класифікації	54
4.4 Оцінювання ефективності системи.....	56
4.5 Склад інсталяційного пакета та розгортання.....	57
4.6 Висновки до четвертого розділу	58
ВИСНОВКИ	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	63
ДОДАТОК А	67
ДОДАТОК Б.....	70
ДОДАТОК В	73
ДОДАТОК Г	78

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

- API – Application Programming Interface, програмний інтерфейс прикладного програмування
- CNN – Convolutional Neural Network, згорткова нейронна мережа
- CSV – Comma-Separated Values, формат табличних даних з розділенням комами
- CT – Computed Tomography, комп'ютерна томографія
- DB – Database, база даних
- DBMS – Database Management System, система керування базами даних
- DICOM – Digital Imaging and Communications in Medicine, стандарт передавання, зберігання та обробки медичних зображень
- DL – Deep Learning, глибоке навчання
- FK – Foreign Key, зовнішній ключ
- GPU – Graphics Processing Unit, графічний процесор
- GUI – Graphical User Interface, графічний інтерфейс користувача
- HTTP – HyperText Transfer Protocol, протокол передавання гіпертексту
- HTTPS – HyperText Transfer Protocol Secure, захищений протокол передавання гіпертексту
- ID – Identifier, ідентифікатор
- ML – Machine Learning, машинне навчання
- MRI – Magnetic Resonance Imaging, магнітно-резонансна томографія
- ORM – Object-Relational Mapping, об'єктно-реляційне відображення
- PACS – Picture Archiving and Communication System, система архівування та передавання медичних зображень
- PK – Primary Key, первинний ключ
- QSS – Qt Style Sheets, таблиці стилів для оформлення інтерфейсу Qt

– REST – Representational State Transfer, архітектурний стиль взаємодії вебсервісів

– SQL – Structured Query Language, мова структурованих запитів

– TXT – текстовий формат файлу

– UI – User Interface, інтерфейс користувача

– UX – User Experience, користувацький досвід

– X-ray – рентгенівське зображення

– БД – база даних

– ІС – інформаційна система

– КТ – комп'ютерна томографія

– МРТ – магнітно-резонансна томографія

– ПЗ – програмне забезпечення

– СКБД – система керування базами даних

ВСТУП

Актуальність теми зумовлена необхідністю підвищення точності та оперативності діагностики захворювань у сучасній медичній практиці в умовах зростання обсягів медичних зображень і цифровізації охорони здоров'я. Традиційні методи візуального аналізу медичних зображень, що ґрунтуються виключно на експертному оцінюванні лікаря, є трудомісткими, суб'єктивними та не завжди забезпечують стабільну відтворюваність результатів, особливо за наявності складних патологічних змін або великих масивів даних. У цих умовах застосування інтелектуальних інформаційних систем на основі методів глибокого навчання створює передумови для автоматизованої, об'єктивної та високоточної класифікації медичних зображень, що сприяє підвищенню якості діагностичних рішень і зниженню навантаження на медичний персонал.

Використання згорткових нейронних мереж, сучасних архітектур deep learning та методів оброблення зображень дозволяє виявляти приховані закономірності у візуальних медичних даних, забезпечувати стійкість до шумів і варіативності зображень, а також формувати аналітичні моделі для підтримки клінічних рішень у режимі, близькому до реального часу. Інтеграція таких моделей у програмні системи аналізу медичних зображень є важливим напрямом розвитку медичної інформатики та інтелектуальних діагностичних систем.

Метою дослідження є розроблення інтелектуальної системи класифікації медичних зображень із використанням методів глибокого навчання, яка забезпечує автоматизоване оброблення, аналіз і віднесення зображень до відповідних класів патологій з високою точністю та надійністю.

Для досягнення поставленої мети необхідно реалізувати повний цикл оброблення медичних зображень, що включає формування та підготовку вхідних даних, побудову та навчання нейромережових моделей, організацію зберігання результатів і створення програмного інтерфейсу для взаємодії користувача з системою.

У рамках поставленої мети передбачено розв'язання таких **завдань**:

- проаналізувати предметну область автоматизованої класифікації медичних зображень та сучасні підходи до медичної діагностики;
- дослідити існуючі програмні та алгоритмічні рішення у сфері застосування глибокого навчання для аналізу медичних зображень;
- розробити модель даних для представлення медичних зображень і результатів їх класифікації;
- спроектувати архітектуру інтелектуальної системи класифікації медичних зображень із використанням нейромережевих методів;
- реалізувати програмний прототип системи на основі алгоритмів глибокого навчання;
- провести експериментальне дослідження та оцінити ефективність роботи розробленої системи.

Об'єктом дослідження є процес автоматизованого аналізу та класифікації медичних зображень у діагностичних інформаційних системах.

Предметом дослідження є методи та програмні засоби глибокого навчання для оброблення й класифікації медичних зображень.

Для досягнення поставлених завдань у роботі використано методи системного аналізу, цифрової обробки зображень, машинного та глибокого навчання, статистичної обробки даних, а також методи оцінювання якості класифікаційних моделей. Програмна реалізація системи передбачає використання мов програмування Python із застосуванням бібліотек TensorFlow або PyTorch, OpenCV, NumPy та засобів розроблення інтерфейсу користувача, що забезпечує інтеграцію моделей глибокого навчання з програмною оболонкою системи.

Наукова новизна роботи полягає у розробленні інтелектуальної системи класифікації медичних зображень, що поєднує сучасні архітектури глибокого навчання з програмними засобами автоматизованого аналізу та інтерпретації результатів класифікації.

Практична цінність роботи полягає у можливості використання розробленої системи як допоміжного інструменту для лікарів-діагностів під час аналізу медичних зображень, а також у навчальних і дослідницьких цілях, що сприяє підвищенню точності діагностики, скороченню часу аналізу та підвищенню ефективності медичних інформаційних систем.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область даного дослідження пов'язана з автоматизованою обробкою, аналізом і класифікацією медичних зображень у складі інтелектуальних інформаційних систем медичного призначення. У сучасних умовах розвитку цифрової медицини обсяги діагностичних зображень, що формуються під час клінічних обстежень, постійно зростають, що ускладнює їх оперативний і якісний аналіз із використанням виключно традиційних методів візуальної інтерпретації. Це зумовлює необхідність застосування програмних засобів автоматизованого аналізу на основі методів штучного інтелекту та глибокого навчання.

Медичні зображення формуються різними діагностичними пристроями та зберігаються у стандартизованих форматах, зокрема DICOM, у спеціалізованих системах архівування та передавання зображень (PACS), що функціонують у складі медичних інформаційних систем лікувальних закладів. Такі зображення супроводжуються структурованими метаданими, які містять відомості про параметри дослідження, тип обладнання, часові характеристики та ідентифікаційні дані. У межах предметної області ці дані розглядаються як вхідна інформація для інтелектуальної системи класифікації.

Ключовим елементом предметної області є процес перетворення вхідних медичних зображень у формалізовані результати класифікації з використанням моделей глибокого навчання. Зазначений процес включає етапи імпорту та валідації даних, попереднього оброблення і нормалізації зображень, виконання інференсу нейромережевої моделі, а також формування результатів у вигляді прогнозованих класів і відповідних ймовірнісних оцінок. Окрему роль відіграють механізми керування доступом користувачів, збереження результатів аналізу та ведення журналів подій, що забезпечує відповідність системи вимогам безпеки та трасованості.

На рисунку 1.1 представлено узагальнену функціональну модель предметної області інтелектуальної системи класифікації медичних зображень, яка відображає основні процеси оброблення даних та інформаційні потоки між ними.

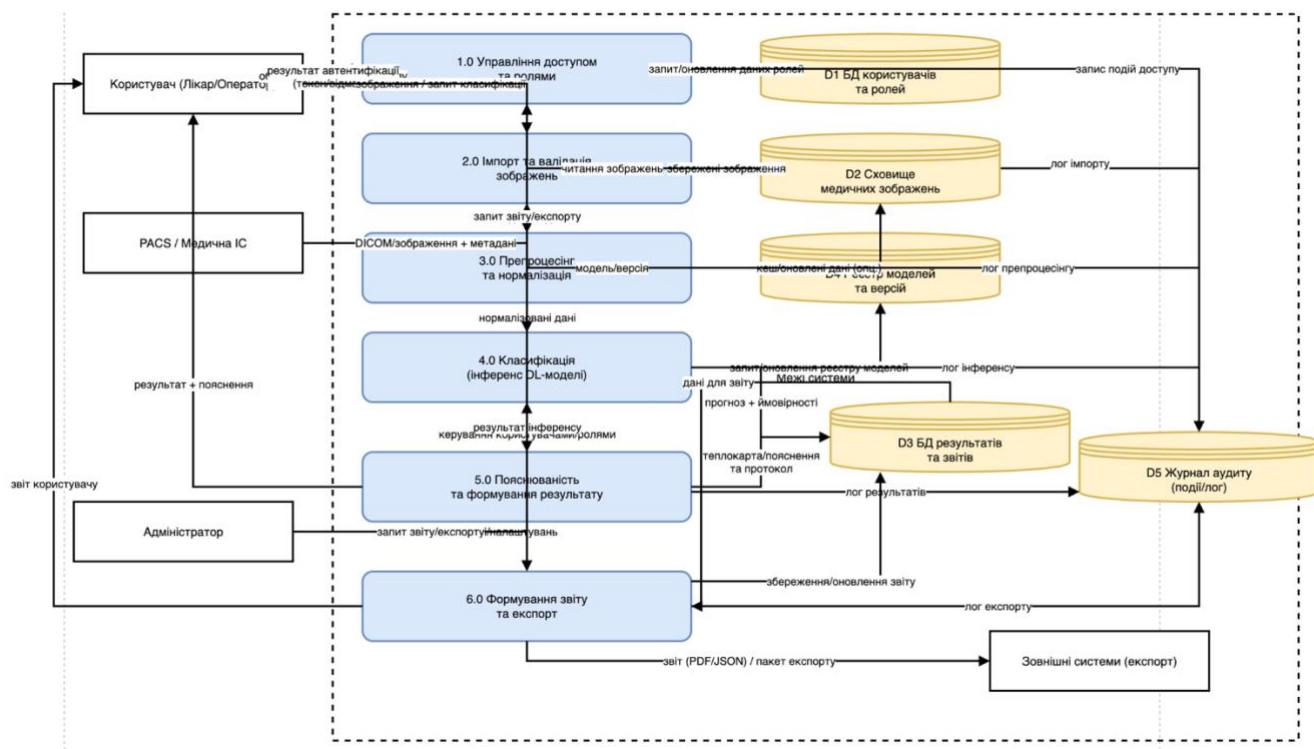


Рисунок 1.1 – Функціональна модель предметної області інтелектуальної системи класифікації медичних зображень

Як показано на рисунку 1.1, предметна область включає такі функціональні процеси: управління доступом та ролями користувачів, імпорт і валідацію медичних зображень із зовнішніх систем, попереднє оброблення і нормалізацію даних, класифікацію з використанням моделей глибокого навчання, формування пояснюваних результатів і підготовку звітної інформації. Взаємодія між процесами реалізується через централізовані сховища даних, що забезпечують збереження зображень, моделей, результатів аналізу та журналів аудиту.

Для систематизації основних інформаційних сутностей предметної області у таблиці 1.1 наведено їх опис та функціональне призначення.

Таблиця 1.1 – Основні інформаційні сутності предметної області

Інформаційна сутність	Характеристика
Медичне зображення	Вхідні візуальні дані діагностичного дослідження у стандартизованому форматі
Метадані зображення	Структурована інформація про параметри та умови проведення дослідження
Модель глибокого навчання	Навчена нейромережева модель для виконання класифікації
Результат інференсу	Клас патології та відповідні ймовірнісні оцінки
Пояснення результату	Інтерпретоване подання рішення моделі для користувача
Звіт	Формалізований результат аналізу для збереження або експорту

На основі проведеного опису предметної області встановлено, що програмне забезпечення системи класифікації медичних зображень є складною програмною системою, яка поєднує процеси збору та оброблення медичних даних, нейромережевий аналіз, механізми безпеки та засоби представлення результатів. Зазначені особливості предметної області визначають вимоги до архітектури, функціональної структури та програмної реалізації системи, що є підґрунтям для подальшого аналізу існуючих рішень і формування вимог до розроблюваного програмного забезпечення.

1.2 Аналіз існуючих рішень

На сьогоднішній день на ринку медичних інформаційних технологій представлено низку програмних рішень, що реалізують автоматизований аналіз і класифікацію медичних зображень із використанням методів глибокого навчання. Дані системи орієнтовані на різні клінічні сценарії, типи зображень і рівні інтеграції з медичними інформаційними середовищами. Аналіз існуючих рішень дозволяє визначити їх функціональні можливості, архітектурні обмеження та напрямки подальшого вдосконалення.

Одним із прикладів промислового впровадження інтелектуального аналізу медичних зображень є платформа Viz.ai, що використовується для

автоматизованого аналізу комп'ютерної томографії з метою раннього виявлення критичних патологій. На рисунку 1.2 наведено інтерфейс мобільного застосунку Viz.ai, який поєднує перегляд КТ-зображень, клінічні показники пацієнта та засоби комунікації між лікарями. Основною особливістю даного рішення є орієнтація на швидке прийняття клінічних рішень, однак система має закриту архітектуру та обмежені можливості адаптації моделей.



Рисунок 1.2 – Інтерфейс мобільного застосунку Viz.ai для аналізу КТ-зображень і клінічної комунікації

Іншим напрямом розвитку є системи цифрової патології, орієнтовані на аналіз гістопатологічних зображень високої роздільної здатності. На рисунку 1.3 представлено платформу PathAI AISight, яка використовується для управління патологічними зображеннями, аналізу тканинних зрізів і впровадження моделей штучного інтелекту у робочі процеси лабораторій. Дане рішення забезпечує високу точність аналізу, проте орієнтоване переважно на вузьку предметну область і не підтримує універсальну класифікацію різних типів медичних зображень.

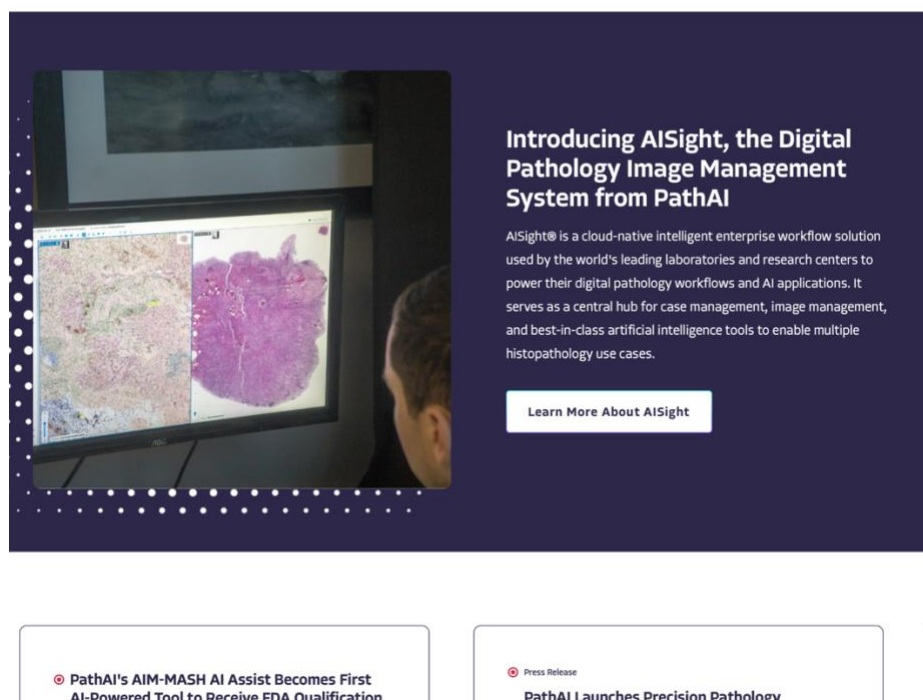


Рисунок 1.3 – Платформа PathAI AISight для аналізу гістопатологічних зображень

Серед рішень для радіологічної діагностики поширення набули хмарні платформи, що забезпечують автоматизований аналіз МРТ та КТ-досліджень. На рисунку 1.4 наведено інтерфейс платформи Arterys, яка використовує глибокі нейронні мережі для сегментації та кількісного аналізу анатомічних структур. Основною перевагою даного рішення є розвинені інструменти візуалізації, однак залежність від хмарної інфраструктури ускладнює використання в автономних медичних системах.



Рисунок 1.4 – Інтерфейс платформи Arterys для автоматизованого аналізу МРТ-зображень

Окрему групу становлять платформи клінічної координації, які поєднують аналіз медичних зображень із підтримкою клінічних робочих процесів. На рисунку 1.5 показано вебінтерфейс платформи Viz.ai, орієнтованої на координацію дій медичного персоналу на основі результатів автоматизованого аналізу зображень. Такі системи ефективні в екстреній медицині, проте не забезпечують гнучкого налаштування алгоритмів класифікації.

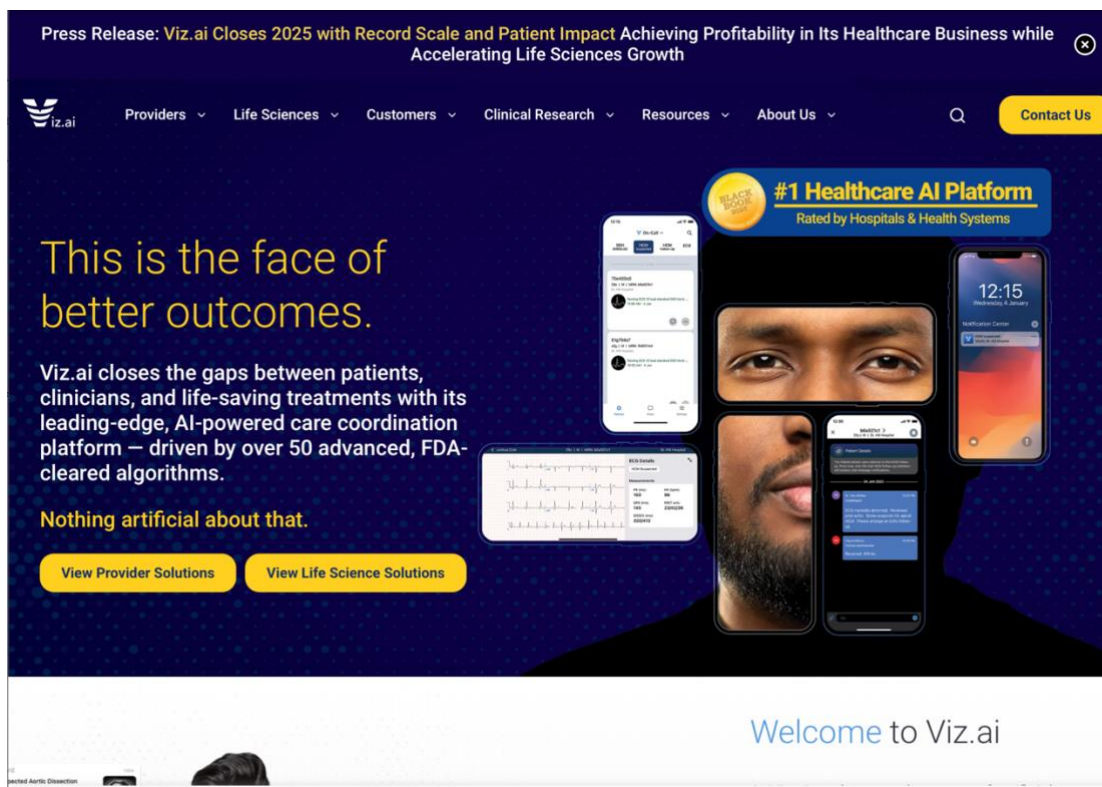


Рисунок 1.5 – Вебінтерфейс платформи Viz.ai для клінічної координації та аналізу медичних зображень

Також застосовуються спеціалізовані системи комп'ютерної підтримки діагностики, орієнтовані на виявлення конкретних патологій. На рисунку 1.6 наведено приклад рішення Aidoc, що виконує автоматичне виявлення патологічних змін на КТ-зображеннях та формує попередні висновки для лікаря. Недоліком таких систем є вузька спеціалізація та обмежені можливості повторного навчання моделей.



Рисунок 1.6 – Інтерфейс системи Aidoc для автоматизованого виявлення патологій на КТ-зображеннях

Для узагальнення результатів аналізу у таблиці 1.2 наведено порівняння основних характеристик існуючих рішень із розроблюваною інтелектуальною системою.

Таблиця 1.2 – Порівняння існуючих рішень із розроблюваною системою

Критерій	Viz.ai	PathAI AISight	Arterys	Aidoc	Розроблювана система
Тип зображень	КТ	Гістологія	КТ, МРТ	КТ	Різні типи
Використання DL	Так	Так	Так	Так	Так
Пояснюваність результатів	Обмежена	Часткова	Обмежена	Обмежена	Передбачена
Гнучкість архітектури	Низька	Середня	Середня	Низька	Висока
Можливість адаптації моделей	Ні	Обмежена	Обмежена	Ні	Так
Орієнтація на навчальні задачі	Ні	Частково	Ні	Ні	Так

На основі проведеного аналізу встановлено, що більшість існуючих рішень є комерційними, закритими та орієнтованими на вузькі клінічні сценарії. Це

обмежує їх універсальність і можливість адаптації до нових типів медичних зображень і задач класифікації. У зв'язку з цим актуальним є розроблення інтелектуальної системи класифікації медичних зображень із використанням глибокого навчання, яка забезпечує модульну архітектуру, підтримку різних типів зображень, пояснюваність результатів і можливість подальшого розвитку алгоритмів.

1.3 Моделювання програмної системи

Моделювання предметної області інтелектуальної системи класифікації медичних зображень виконано з метою формалізації основних функціональних процесів, ролей користувачів та логіки взаємодії між компонентами системи. Використання UML-діаграм та діаграм діяльності дозволяє наочно відобразити сценарії використання системи, послідовність оброблення даних і алгоритмічну структуру процесів аналізу медичних зображень.

На першому етапі моделювання побудовано діаграму прецедентів, яка відображає функціональні можливості системи з точки зору користувачів різних ролей. На рисунку 1.7 наведено UML-діаграму прецедентів інтелектуальної системи класифікації медичних зображень.

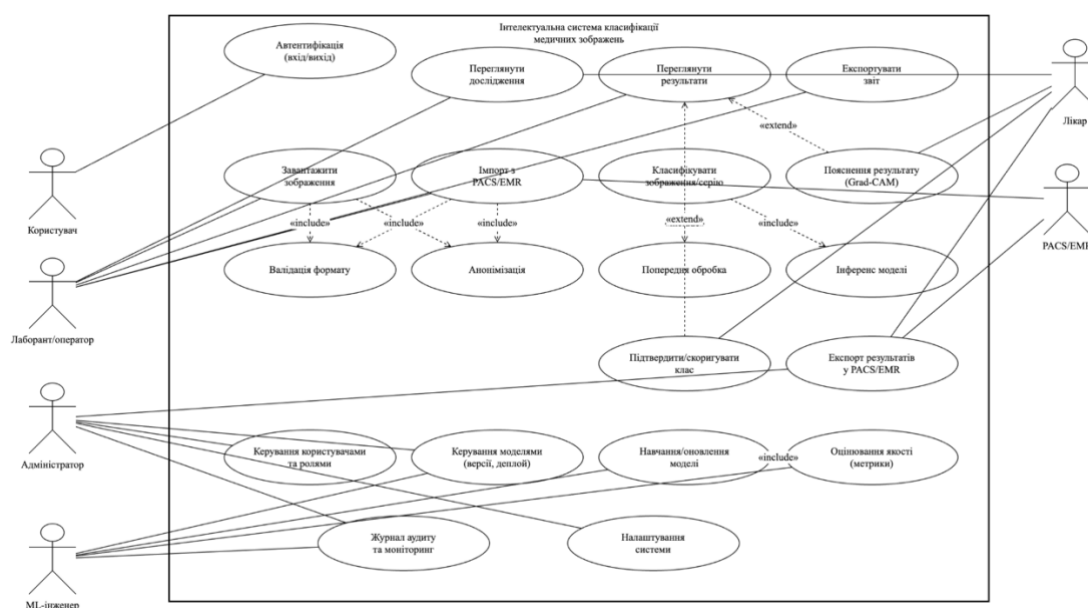


Рисунок 1.7 – Діаграма прецедентів інтелектуальної системи класифікації медичних зображень

Як показано на рисунку 1.7, у системі визначено декілька основних акторів, зокрема лікар, лаборант/оператор, адміністратор та ML-інженер. Ключові прецеденти охоплюють автентифікацію користувачів, завантаження та імпорт зображень із систем PACS/EMR, валідацію та анонімізацію даних, класифікацію зображень із використанням моделей глибокого навчання, формування пояснень результатів (Grad-CAM), перегляд і експорт результатів, а також адміністрування користувачів, моделей і параметрів системи. Така модель дозволяє чітко визначити межі відповідальності кожного актора та функціональну повноту системи.

Для детальнішого опису логіки взаємодії між користувачем і внутрішніми сервісами системи побудовано діаграму послідовності. На рисунку 1.8 представлено UML-діаграму послідовності процесу завантаження медичного зображення, виконання класифікації та формування результатів.

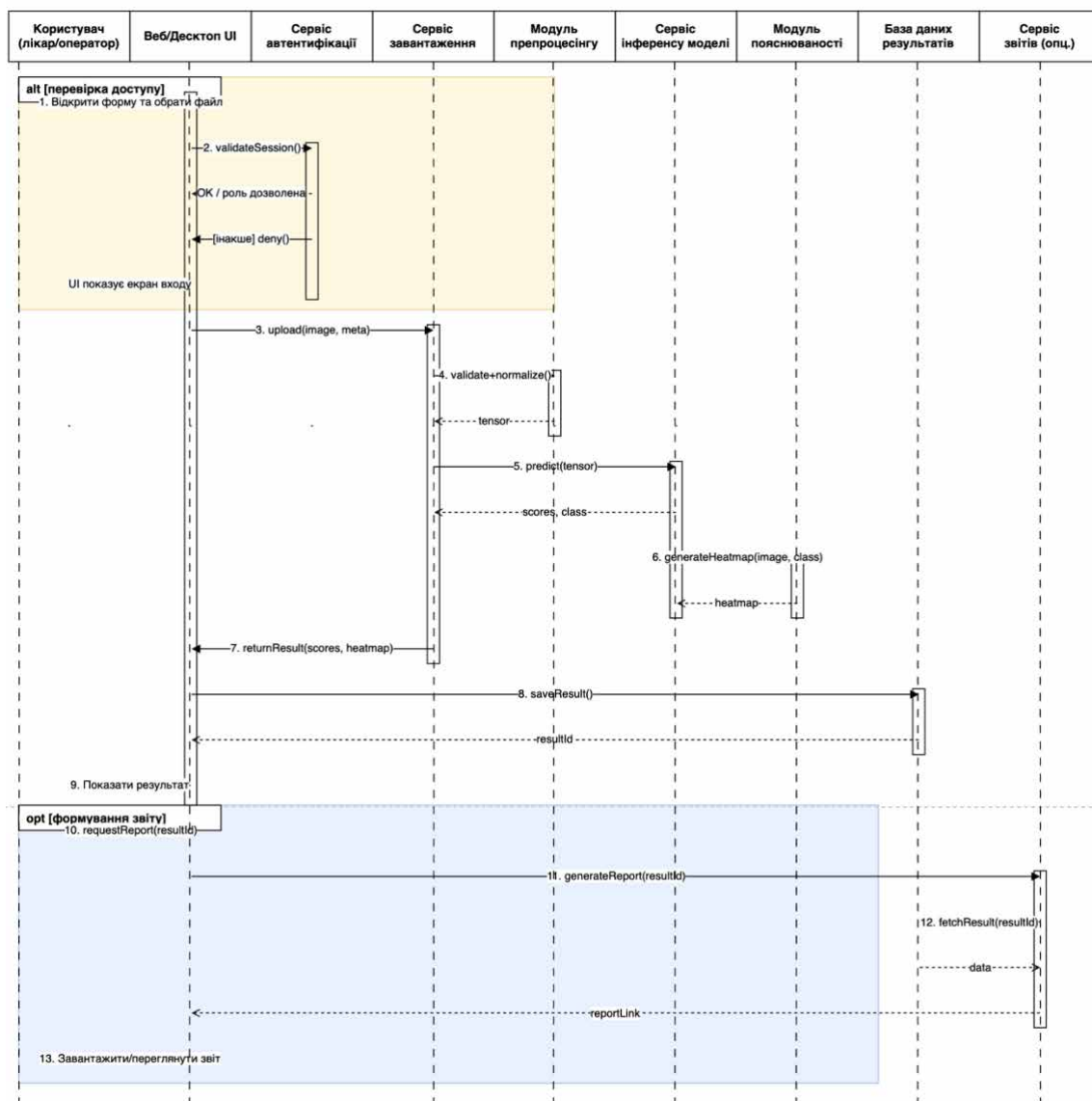


Рисунок 1.8 – Діаграма послідовності процесу класифікації медичного зображення

Згідно з рисунком 1.8, процес починається з перевірки доступу користувача та його ролі, після чого здійснюється завантаження зображення разом із метаданими. Далі система виконує валідацію та нормалізацію даних, передає підготовлений тензор до сервісу інференсу моделі та отримує результати класифікації. У разі потреби формується пояснювальна інформація у вигляді теплової карти, після чого результати зберігаються у базі даних і відображаються користувачу. Окремо передбачено опціональний сценарій формування та експорту звіту.

Алгоритмічну структуру оброблення медичних зображень і прийняття рішень у системі формалізовано за допомогою діаграми діяльності. На рисунку 1.9 наведено діаграму діяльності процесу аналізу та класифікації медичних зображень.

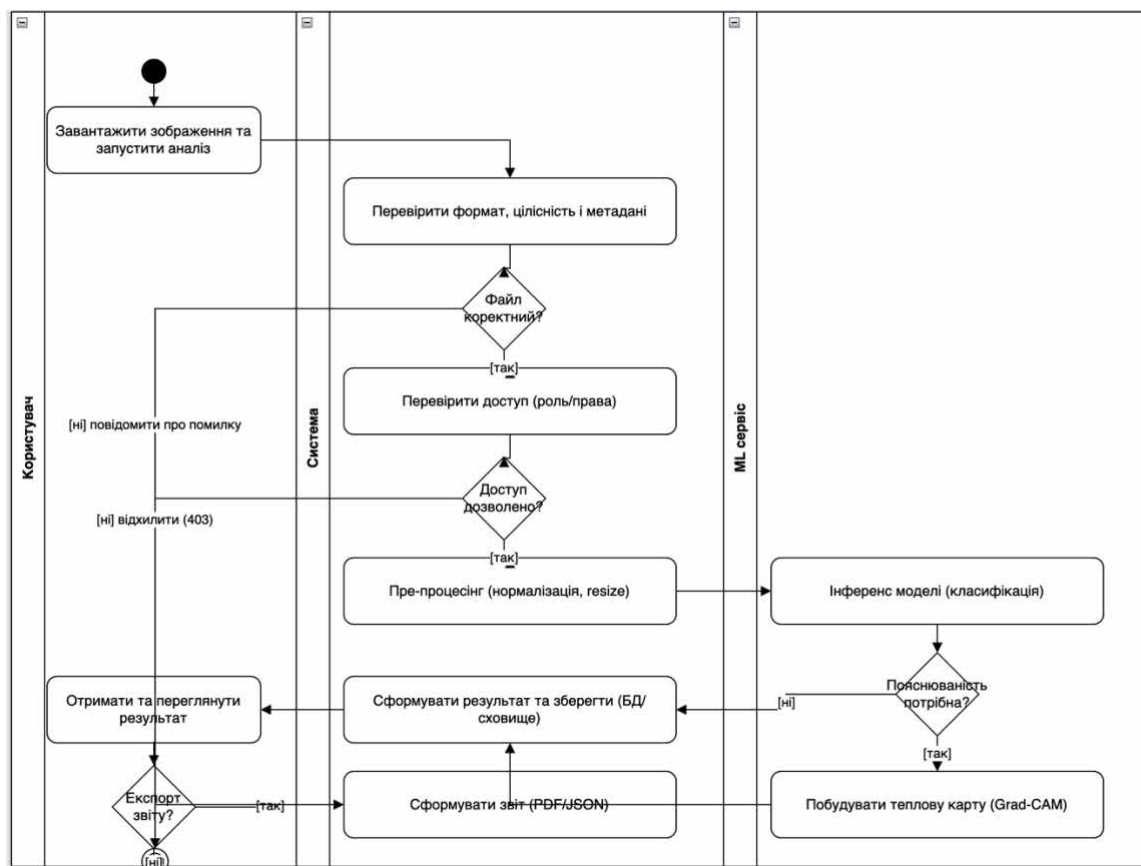


Рисунок 1.9 – Діаграма діяльності процесу аналізу медичних зображень

Як показано на рисунку 1.9, процес аналізу включає послідовні етапи перевірки коректності файлу, контролю доступу користувача, попереднього оброблення зображення, виконання інференсу нейромережевої моделі та формування результатів. У разі виявлення помилок або відсутності прав доступу процес завершується відповідним повідомленням користувача. За потреби пояснення результатів система виконує побудову теплової карти, після чого результати можуть бути збережені та експортовані у вигляді звіту.

У результаті проведеного моделювання предметної області формалізовано функціональну структуру інтелектуальної системи класифікації медичних зображень, визначено ролі користувачів, основні сценарії взаємодії та логіку

оброблення даних. Побудовані моделі є основою для подальшого аналізу вимог, проєктування архітектури системи та реалізації її програмних компонентів.

1.4 Аналіз вимог розроблювальної системи

На основі проведеного аналізу предметної області, існуючих рішень та результатів моделювання процесів функціонування системи сформовано сукупність вимог до інтелектуальної системи класифікації медичних зображень з використанням глибокого навчання. Вимоги визначають функціональні можливості системи, її експлуатаційні характеристики, технічну реалізацію та рівень захисту оброблюваних даних.

Функціональні вимоги визначають перелік основних операцій, які повинна реалізовувати система для забезпечення повного циклу автоматизованої класифікації медичних зображень. Узагальнений перелік функціональних вимог наведено у таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до системи

Код	Вимога
F1	Забезпечення автентифікації та авторизації користувачів з урахуванням ролей
F2	Завантаження медичних зображень у стандартизованих форматах (DICOM, PNG, JPEG)
F3	Імпорт зображень і метаданих із систем PACS/EMR
F4	Виконання валідації та попереднього оброблення зображень
F5	Автоматизована класифікація зображень із використанням DL-моделей
F6	Формування ймовірнісних оцінок результатів класифікації
F7	Побудова пояснюваних результатів (Grad-CAM)
F8	Збереження результатів аналізу у базі даних
F9	Перегляд та експорт результатів у вигляді звітів
F10	Керування моделями, версіями та параметрами системи

Нефункціональні вимоги характеризують якісні властивості системи, що визначають зручність її використання, продуктивність, надійність і масштабованість. Основні нефункціональні вимоги наведено у таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до системи

Категорія	Вимога
Продуктивність	Час виконання інференсу не повинен перевищувати встановлених допустимих меж
Надійність	Забезпечення коректної роботи системи при обробленні великих обсягів даних
Масштабованість	Можливість додавання нових моделей і типів зображень
Зручність використання	Інтуїтивно зрозумілий користувацький інтерфейс
Супроводжуваність	Модульна структура та можливість оновлення компонентів
Відтворюваність	Забезпечення повторюваності результатів класифікації

Технічні вимоги визначають обмеження та умови реалізації програмного забезпечення, а також вимоги до апаратно-програмного середовища. Перелік основних технічних вимог наведено у таблиці 1.5.

Таблиця 1.5 – Технічні вимоги до системи

Параметр	Вимога
Мова програмування	Python для реалізації ML-частини та серверної логіки
Бібліотеки DL	TensorFlow або PyTorch
Обробка зображень	Використання бібліотек OpenCV, NumPy
Архітектура	Модульна, з розділенням сервісів
Інтерфейс	Веб- або десктоп-інтерфейс користувача
Зберігання даних	Реляційна або документна база даних
Інтеграція	Підтримка взаємодії з PACS/EMR

Безпекові вимоги визначають заходи захисту інформації, що обробляється системою, з урахуванням чутливості медичних даних. Основні безпекові вимоги наведено у таблиці 1.6.

Таблиця 1.6 – Безпекові вимоги до системи

Аспект безпеки	Вимога
Контроль доступу	Розмежування прав користувачів за ролями
Конфіденційність	Анонімізація персональних медичних даних
Цілісність	Захист даних від несанкціонованих змін
Аудит	Ведення журналу дій користувачів
Захист каналів	Захищене передавання даних між компонентами
Відповідність стандартам	Дотримання вимог до обробки медичних даних

У результаті аналізу вимог встановлено, що розроблювана програма для забезпечення системи класифікації медичних зображень повинна поєднувати широкий набір функціональних можливостей із високими показниками надійності, безпеки та масштабованості. Сформульовані вимоги є основою для подальшої постановки завдання, проєктування архітектури та реалізації програмного забезпечення системи.

1.5 Постановка завдання

На основі проведеного аналізу предметної області, існуючих програмних рішень, результатів моделювання процесів системи та сформульованих вимог постає завдання розроблення інтелектуальної системи класифікації медичних зображень з використанням методів глибокого навчання. Розроблювана система повинна забезпечувати автоматизований аналіз медичних зображень, підтримку клінічних сценаріїв використання та інтеграцію з медичними інформаційними системами.

Завдання полягає у створенні програмного забезпечення, яке реалізує повний цикл оброблення медичних зображень — від моменту їх завантаження або імпорту з зовнішніх систем до формування, збереження та експорту результатів класифікації. Система повинна забезпечувати контроль доступу користувачів, коректну обробку вхідних даних, виконання інференсу нейромережових моделей та формування пояснюваних результатів для підвищення довіри до прийнятих рішень.

Вхідними даними системи є медичні зображення у стандартизованих форматах, зокрема DICOM, а також пов'язані з ними метадані, що описують параметри дослідження. Додатковими вхідними даними є параметри конфігурації системи, моделі глибокого навчання та налаштування доступу користувачів відповідно до їх ролей.

Вихідними даними системи є результати класифікації медичних зображень у вигляді визначеного класу або набору класів із відповідними ймовірнісними

оцінками, пояснювальні матеріали у вигляді теплових карт або інших інтерпретаційних візуалізацій, а також сформовані звіти для подальшого перегляду або експорту. Результати повинні зберігатися у базі даних та бути доступними для повторного аналізу.

Для реалізації поставленого завдання необхідно спроектувати архітектуру системи, що включає користувацький інтерфейс, серверні модулі оброблення даних, сервіси інференсу нейромережових моделей, механізми збереження результатів та засоби аудиту і моніторингу. Програмна реалізація повинна забезпечувати модульність, масштабованість і можливість подальшого розширення функціональних можливостей системи.

Результатом виконання поставленого завдання має бути працездатний програмний прототип інтелектуальної системи класифікації медичних зображень, який відповідає сформульованим функціональним, нефункціональним, технічним і безпековим вимогам та може бути використаний як основа для впровадження у медичних інформаційних системах або подальших наукових досліджень.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Загальна характеристика проєктованої системи

Програмне забезпечення системи класифікації медичних зображень з використанням глибокого навчання проєктується як навчальний програмний комплекс, призначений для завантаження медичних зображень, їх попередньої обробки, автоматизованої класифікації за допомогою нейронної моделі, збереження результатів аналізу та формування діагностичного висновку. Система не замінює медичного спеціаліста, а використовується як програмний прототип підтримки аналізу зображень і демонстрації можливостей алгоритмів глибокого навчання.

Основними користувачами системи є адміністратор, лаборант, лікар-діагност і пацієнт. Адміністратор відповідає за керування доступом, користувачами та налаштуваннями системи. Лаборант створює дослідження, реєструє дані пацієнта та завантажує медичні зображення. Лікар-діагност переглядає результати класифікації, перевіряє сформовані системою висновки та за потреби коригує їх. Пацієнт у межах системи виступає об'єктом дослідження та отримує підготовлений висновок.

Програмне забезпечення має забезпечувати повний цикл обробки зображення: авторизацію користувача, створення або вибір дослідження, завантаження зображення, перевірку формату й розміру файлу, попередню обробку, запуск інференсу моделі, визначення ймовірностей класів, вибір найбільш імовірного класу, збереження результату та формування звіту. Загальна логіка роботи системи орієнтована на мінімізацію ручних дій користувача та підвищення структурованості збереження результатів аналізу.

У процесі проєктування враховано, що медичні зображення мають зберігатися разом із метаданими дослідження, параметрами попередньої обробки, результатами класифікації, версією використаної моделі та

сформованим висновком. Це дозволяє відтворити хід обробки, перевірити результат і сформулювати звіт для подальшого аналізу.

2.2 Діаграма класів і кооперації системи

Статичну структуру програмного забезпечення подано за допомогою діаграми класів, наведеної на рис. 2.1.

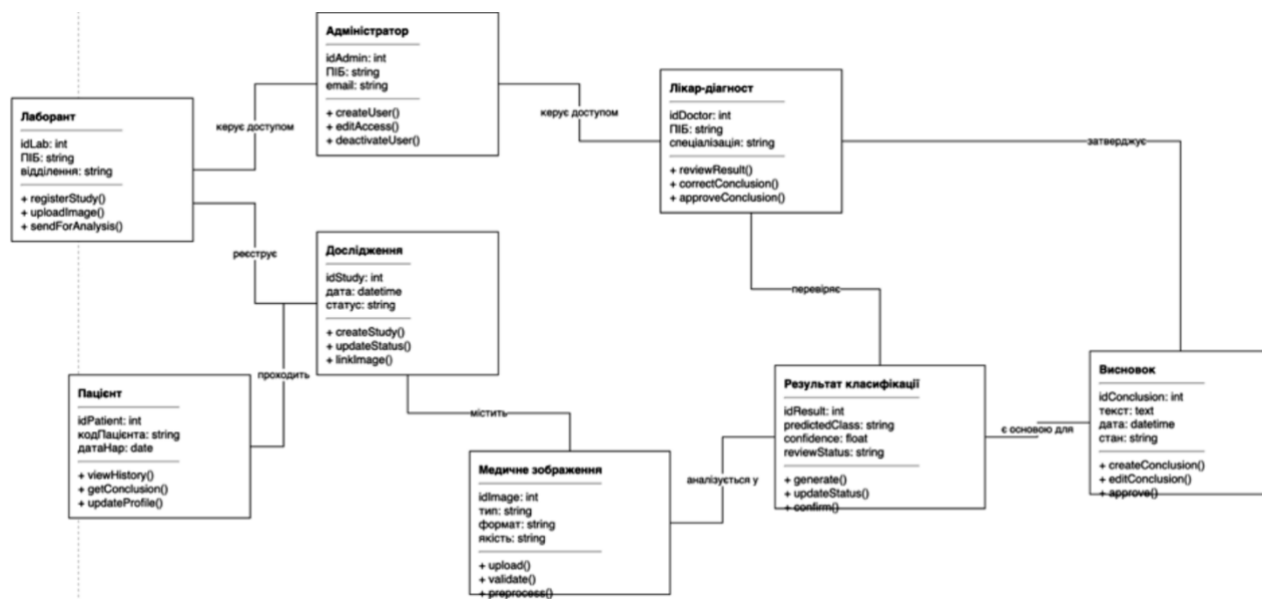


Рисунок 2.1 – Діаграма класів інтелектуальної системи класифікації медичних зображень

Діаграма класів відображає основні сутності системи та зв'язки між ними. Центральними класами є «Пациєнт», «Дослідження», «Медичне зображення», «Результат класифікації» та «Висновок». Клас «Пациєнт» містить ідентифікатор, код пацієнта та дату народження. Він пов'язаний із дослідженням, оскільки одне або декілька досліджень можуть виконуватися для одного пацієнта.

Клас «Дослідження» описує окремий випадок аналізу медичного зображення. Він містить ідентифікатор дослідження, дату, статус і методи створення, оновлення статусу та прив'язки зображення. До одного дослідження може належати одне або кілька медичних зображень. Клас «Медичне зображення» зберігає відомості про файл, тип, формат і якість зображення, а також містить операції завантаження, перевірки та попередньої обробки.

Клас «Результат класифікації» містить прогнозований клас, рівень упевненості моделі та статус перевірки. Саме цей клас відображає результат роботи нейронної моделі. Клас «Висновок» формується на основі результату класифікації та містить текстовий опис, дату створення і стан затвердження. Лікар-діагност має можливість переглянути результат, відкоригувати висновок і затвердити його.

Окрему групу становлять класи користувачів: «Адміністратор», «Лаборант» і «Лікар-діагност». Вони відповідають за різні сценарії роботи. Адміністратор керує доступом, лаборант реєструє дослідження та завантажує зображення, а лікар-діагност перевіряє результати й формує остаточний висновок.

Для деталізації взаємодії об'єктів системи побудовано три діаграми кооперації. Перша діаграма відображає кооперацію між адміністратором, лаборантом і лікарем-діагностом під час надання доступу до системи. Вона наведена на рис. 2.2.

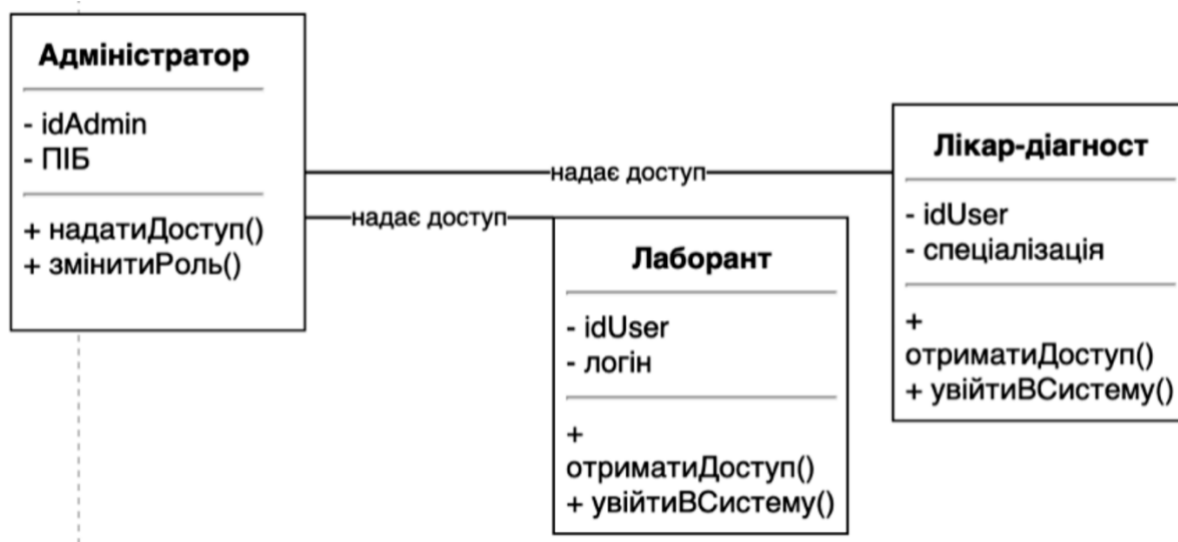


Рисунок 2.2 – Діаграма кооперації керування доступом користувачів

У цьому сценарії адміністратор створює користувачів, призначає ролі та надає доступ до системи. Лаборант і лікар-діагност отримують доступ відповідно до своїх функціональних обов'язків. Такий підхід забезпечує розмежування прав, оскільки лаборант не повинен мати можливість затверджувати медичний

висновок, а лікар-діагност не повинен виконувати адміністративні дії з обліковими записами користувачів.

Друга діаграма кооперації описує створення дослідження та завантаження медичного зображення. Вона наведена на рис. 2.3.



Рисунок 2.3 – Діаграма кооперації створення дослідження та завантаження зображення

У цьому сценарії лаборант створює нове дослідження, вводить дані пацієнта, завантажує файл медичного зображення та передає його на обробку. Пацієнт ідентифікується в системі через код або медичний запис. Дослідження містить завантажене медичне зображення, яке проходить перевірку формату, розміру та якості. Після цього зображення може бути передане до модуля класифікації.

Третя діаграма кооперації відображає взаємодію між медичним зображенням, результатом класифікації, лікарем-діагностом, пацієнтом і висновком. Вона наведена на рис. 2.4.

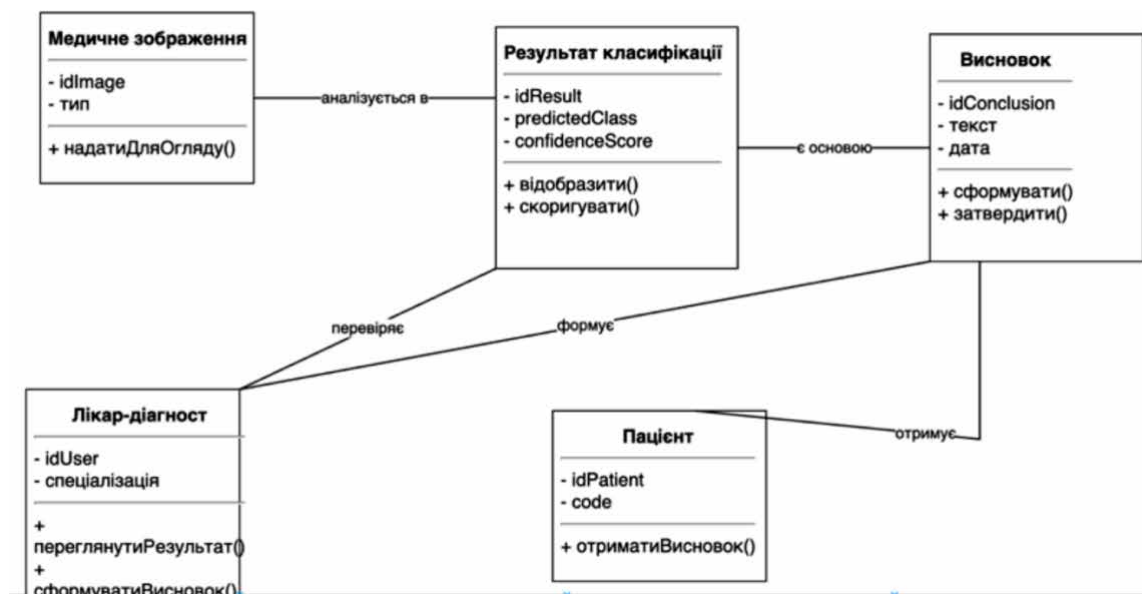


Рисунок 2.4 – Діаграма кооперації перевірки результату та формування висновку

Медичне зображення передається до модуля класифікації, після чого формується результат із прогнозованим класом і рівнем упевненості. Лікар-діагност переглядає результат, аналізує його та формує текстовий висновок. Після затвердження висновок стає доступним для перегляду або експорту. Така кооперація забезпечує участь медичного спеціаліста у фінальному прийнятті рішення та зменшує ризик неконтрольованого використання автоматизованого прогнозу.

2.3 Пакетна та компонентна структура системи

Пакетна структура системи наведена на рис. 2.5.

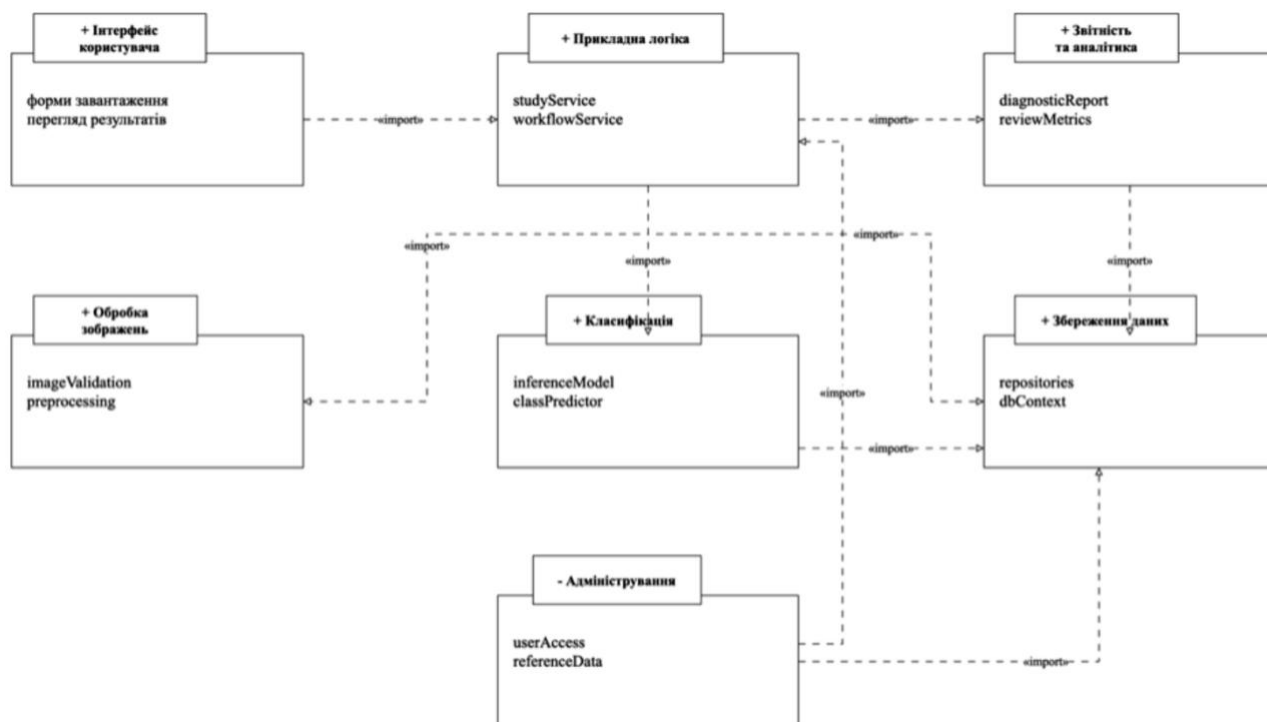


Рисунок 2.5 – Діаграма пакетів інтелектуальної системи класифікації медичних зображень

Пакетна діаграма відображає логічну організацію програмного забезпечення. Система поділена на пакети «Інтерфейс користувача», «Прикладна логіка», «Обробка зображень», «Класифікація», «Збереження даних», «Звітність та аналітика» і «Адміністрування». Така структура дозволяє розділити функції введення даних, обробки зображень, класифікації, збереження результатів і формування звітів.

Пакет «Інтерфейс користувача» відповідає за форми авторизації, завантаження зображень, перегляд результатів і роботу з висновками. Пакет «Прикладна логіка» координує основні сценарії роботи: створення дослідження, обробку запитів, взаємодію з базою даних та виклик модулів класифікації. Пакет «Обробка зображень» містить операції перевірки формату, зміни розміру, нормалізації та підготовки зображення до моделі. Пакет «Класифікація» включає модель інференсу та алгоритм визначення класу. Пакет «Збереження даних» відповідає за репозиторії та контекст бази даних.

Компонентну структуру системи наведено на рис. 2.6.

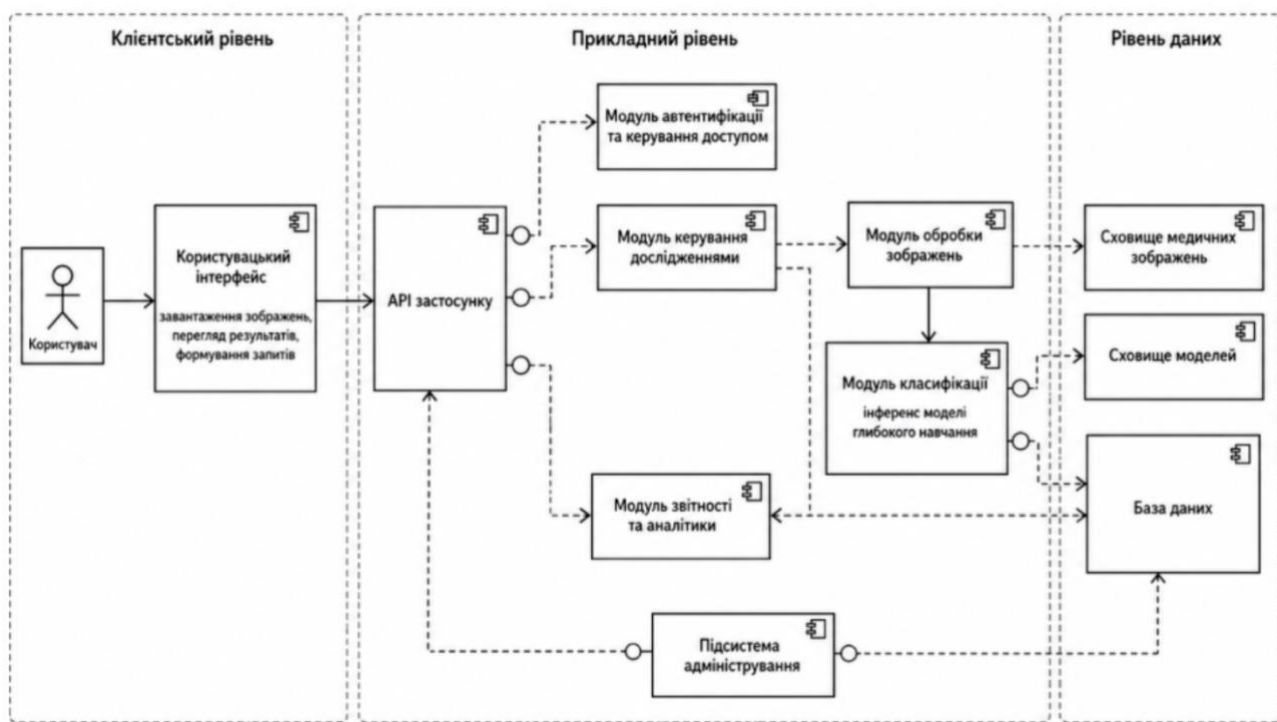


Рисунок 2.6 – Діаграма компонентів системи класифікації медичних зображень

Компонентна діаграма показує поділ системи на клієнтський рівень, прикладний рівень і рівень даних. На клієнтському рівні розташований користувацький інтерфейс, через який користувач завантажує зображення, переглядає результати та формує запити. Прикладний рівень містить API застосунку, модуль автентифікації, модуль керування дослідженнями, модуль обробки зображень, модуль класифікації, модуль звітності та підсистему адміністрування.

Рівень даних складається зі сховища медичних зображень, сховища моделей і бази даних. Сховище зображень використовується для зберігання файлів, сховище моделей — для збереження checkpoint-файлів нейронної мережі, а база даних — для структурованого зберігання інформації про користувачів, пацієнтів, дослідження, результати класифікації та висновки. Завдяки такому поділу система є модульною та може бути розширена без повної зміни архітектури.

2.4 Логічна та фізична модель даних

Для зберігання інформації про користувачів, пацієнтів, дослідження, зображення, моделі, класи діагностики, результати класифікації та звіти побудовано логічну модель даних, наведену на рис. 2.7.

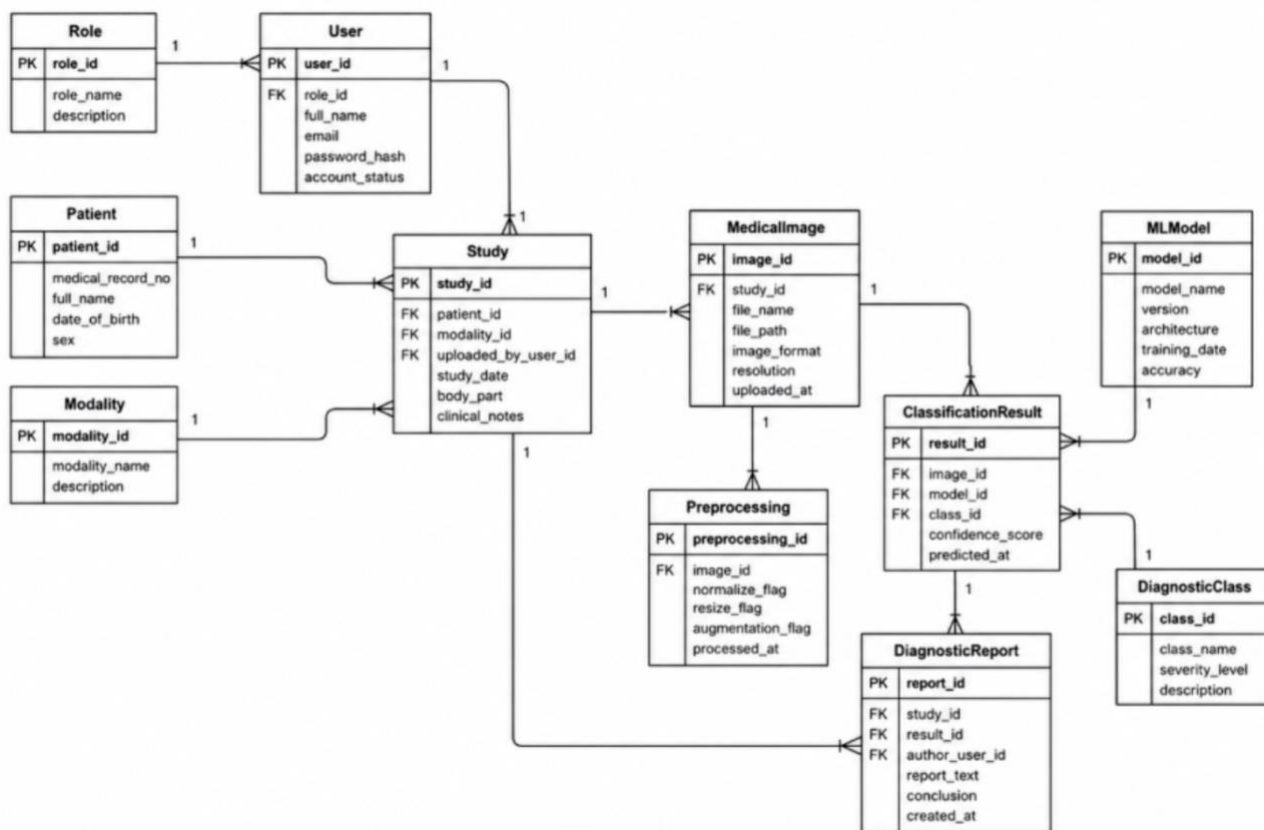


Рисунок 2.7 – Логічна модель даних системи

Логічна модель містить сутності Role, User, Patient, Study, MedicalImage, Preprocessing, MLModel, DiagnosticClass, ClassificationResult і DiagnosticReport. Сутність Role використовується для опису ролей користувачів. Сутність User зберігає облікові записи користувачів і пов'язується з роллю. Сутність Patient описує пацієнта, для якого створюється дослідження. Сутність Study є центральною в логіці обробки, оскільки об'єднує пацієнта, модальність дослідження, користувача, який завантажив дані, дату, частину тіла та клінічні примітки.

Сутність MedicalImage містить дані про файл зображення, формат, роздільну здатність і дату завантаження. Сутність Preprocessing фіксує

параметри попередньої обробки: нормалізацію, зміну розміру та аугментацію. Сутність MLModel містить інформацію про модель глибокого навчання: назву, версію, архітектуру, дату навчання та точність. Сутність DiagnosticClass описує можливі класи класифікації, наприклад норму, пневмонію, пухлинні або запальні зміни. Сутність ClassificationResult зберігає прогнозований клас, використану модель і рівень упевненості. DiagnosticReport містить текст висновку, результат класифікації, автора звіту та дату створення.

Фізичну модель даних наведено на рис. 2.8.

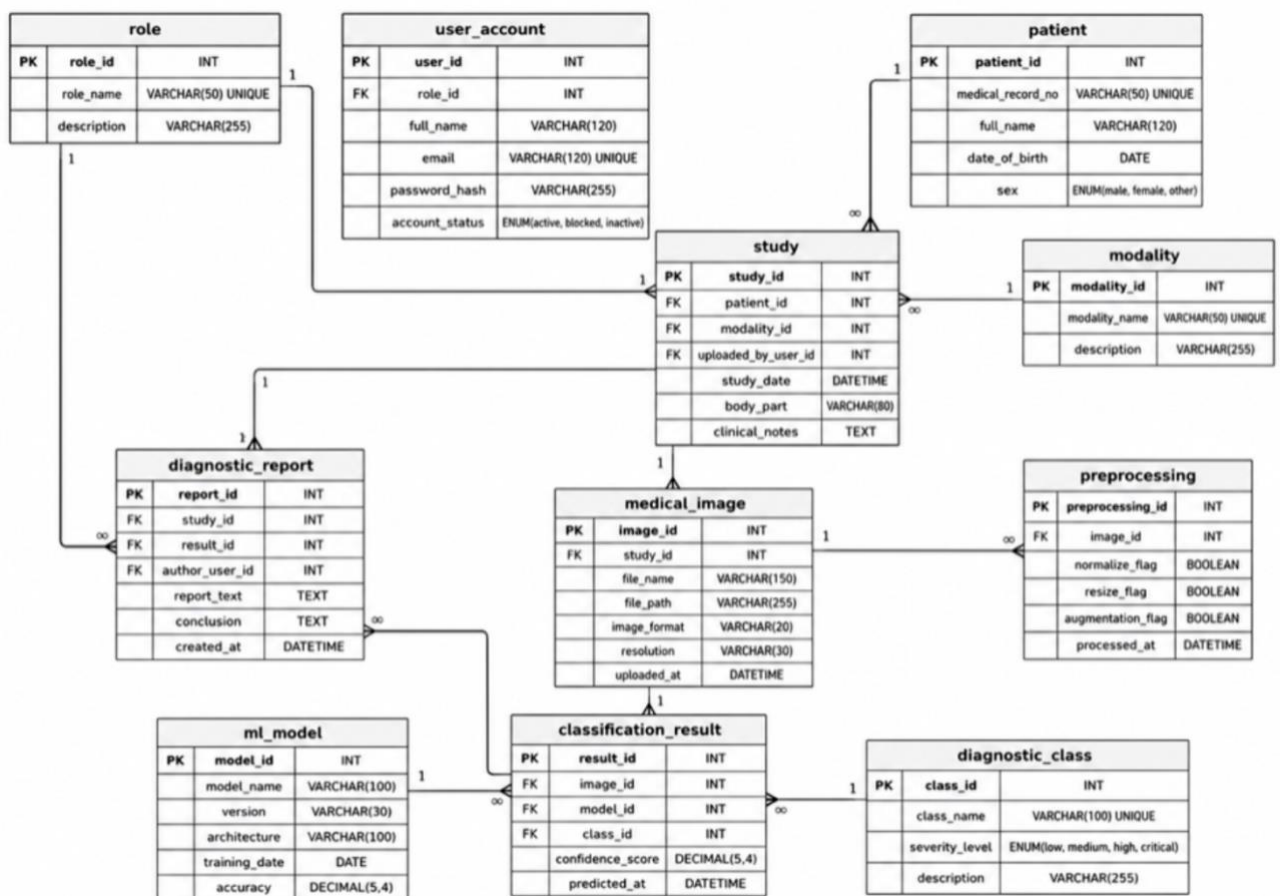


Рисунок 2.8 – Фізична модель даних системи

Фізична модель деталізує структуру таблиць, первинні та зовнішні ключі, типи даних і зв'язки між таблицями. Таблиця user_account містить дані користувачів, role_id, ПІБ, email, password_hash і статус облікового запису. Таблиця patient зберігає медичний номер, ПІБ, дату народження та стать. Таблиця study пов'язує пацієнта, модальність, користувача, дату дослідження, частину тіла та клінічні примітки.

Таблиця `medical_image` зберігає шлях до файлу, формат і роздільну здатність зображення. Таблиця `preprocessing` фіксує параметри попередньої обробки. Таблиця `ml_model` описує модель класифікації, а `diagnostic_class` містить перелік класів. Таблиця `classification_result` зберігає результат інференсу моделі, `confidence_score` і дату прогнозування. Таблиця `diagnostic_report` містить підсумковий текст висновку та посилання на дослідження, результат класифікації й автора звіту.

Основні таблиці інформаційної бази наведено в табл. 2.1.

Таблиця 2.1 – Основні таблиці бази даних

Таблиця	Призначення
<code>role</code>	Зберігання ролей користувачів системи
<code>user_account</code>	Облікові записи користувачів
<code>patient</code>	Дані пацієнтів
<code>modality</code>	Типи медичних досліджень
<code>study</code>	Дані про конкретне дослідження
<code>medical_image</code>	Метадані завантажених медичних зображень
<code>preprocessing</code>	Параметри попередньої обробки зображення
<code>ml_model</code>	Відомості про нейронні моделі
<code>diagnostic_class</code>	Класи діагностичної класифікації
<code>classification_result</code>	Результати класифікації
<code>diagnostic_report</code>	Сформовані діагностичні звіти

Запропонована структура бази даних забезпечує збереження як вхідних даних, так і результатів роботи моделі. Це важливо для подальшого аналізу, повторної перевірки результатів і формування звітної документації.

2.5 Діаграма розгортання системи

Діаграма розгортання наведена на рис. 2.9.

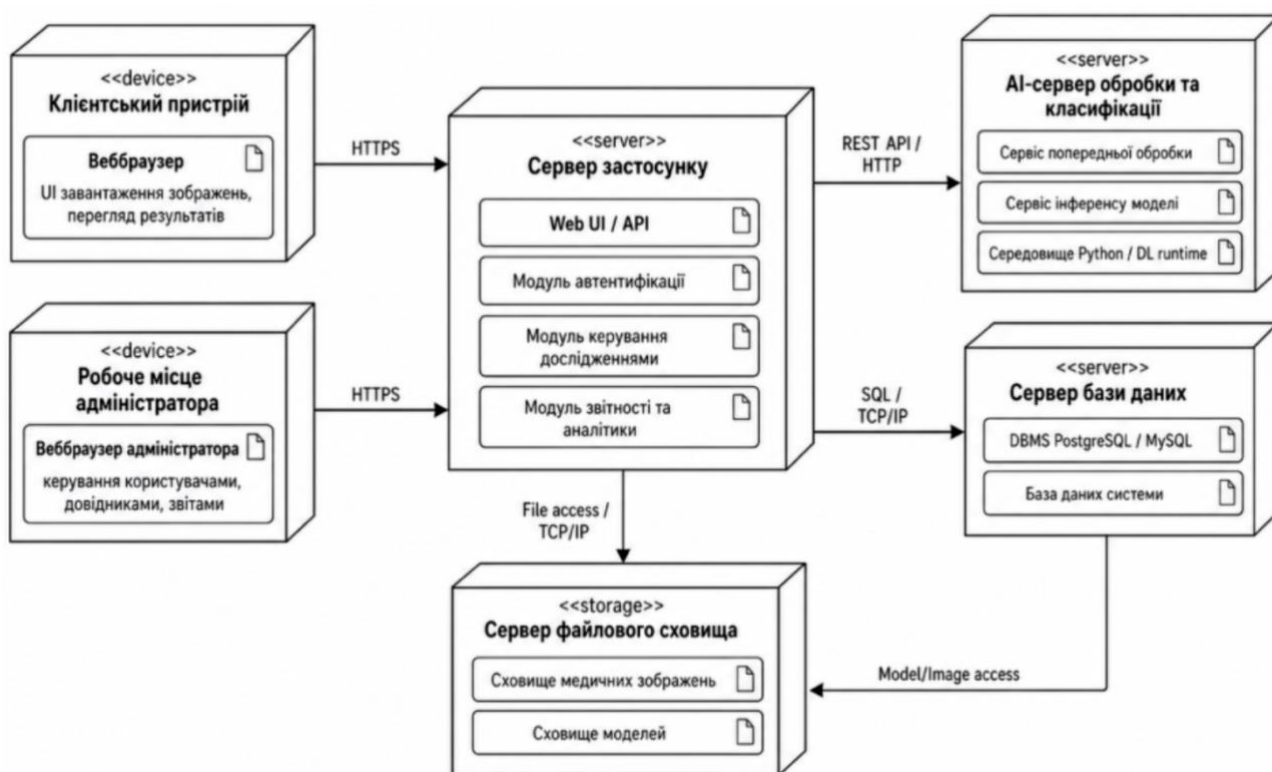


Рисунок 2.9 – Діаграма розгортання системи класифікації медичних зображень

Діаграма розгортання відображає фізичну організацію системи. Клієнтський пристрій користувача та робоче місце адміністратора взаємодіють із сервером застосунку через HTTPS. На сервері застосунку розміщуються Web UI/API, модуль автентифікації, модуль керування дослідженнями та модуль звітності. Окремо виділено AI-сервер обробки та класифікації, який виконує попередню обробку зображень, інференс моделі та роботу середовища Python/DL runtime.

Сервер бази даних містить DBMS PostgreSQL або MySQL і базу даних системи. Сервер файлового сховища використовується для зберігання медичних зображень і моделей. Обмін між сервером застосунку та AI-сервером здійснюється через REST API або HTTP. Доступ до бази даних виконується через SQL/TCP/IP, а доступ до файлів — через файловий інтерфейс або TCP/IP. Така структура дозволяє відокремити основний застосунок, модель класифікації та сховище даних.

2.6 Проєктування алгоритмів роботи системи

Для опису логіки роботи програмних модулів побудовано блок-схеми алгоритмів. Перший алгоритм описує авторизацію користувача і наведений на рис. 2.10.

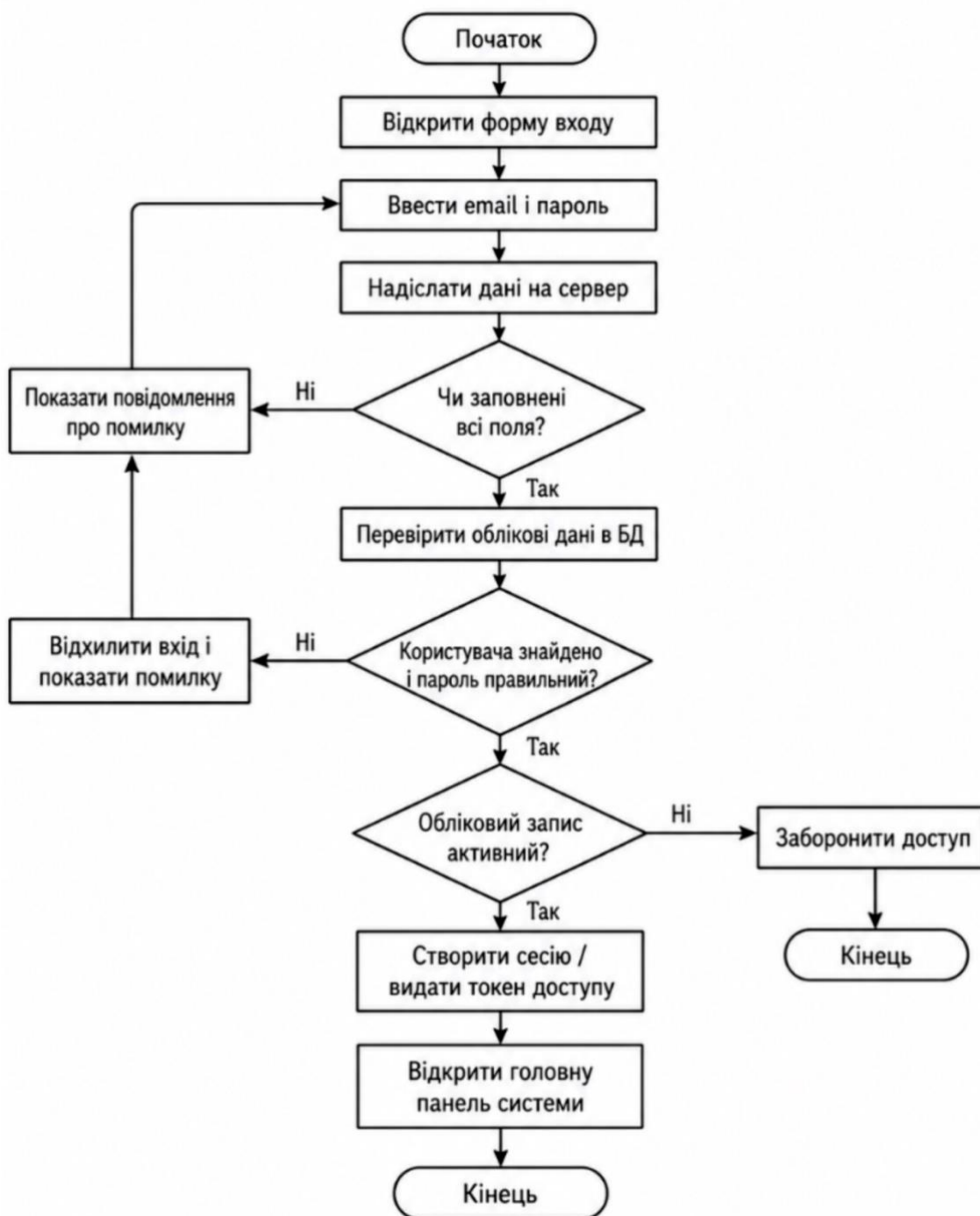


Рисунок 2.10 – Блок-схема алгоритму авторизації користувача

Алгоритм авторизації починається з відкриття форми входу. Користувач вводить логін або email і пароль, після чого дані надсилаються на сервер.

Система перевіряє заповнення полів, наявність користувача в базі даних, правильність пароля та активність облікового запису. Якщо перевірка успішна, створюється сесія або видається токен доступу, після чого відкривається головна панель системи. У разі помилки користувач отримує відповідне повідомлення.

Алгоритм завантаження медичного зображення наведений на рис. 2.11.

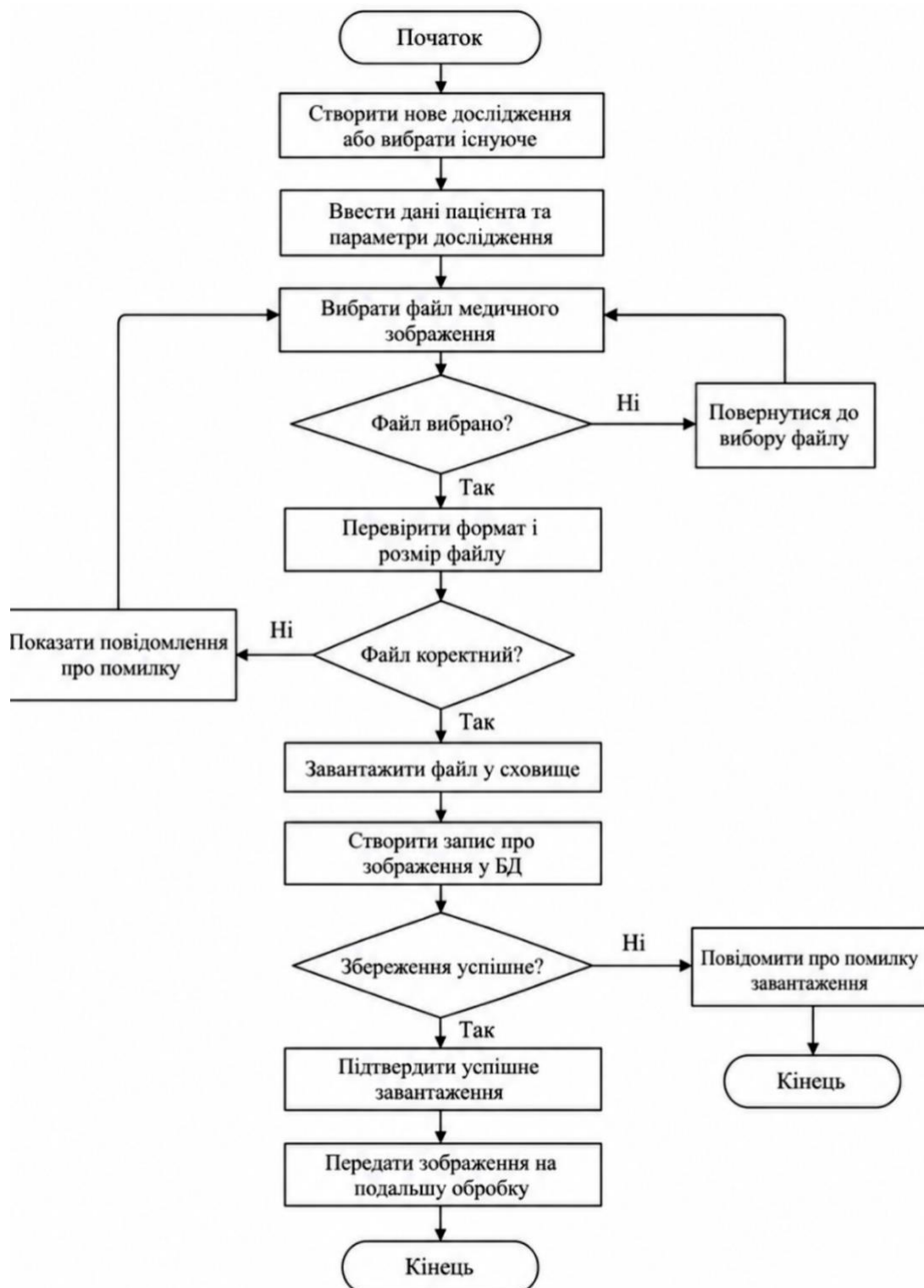


Рисунок 2.11 – Блок-схема алгоритму завантаження медичного зображення

Алгоритм передбачає створення нового дослідження або вибір існуючого, введення даних пацієнта, вибір файлу медичного зображення, перевірку формату й розміру. Якщо файл некоректний, система показує повідомлення про помилку та повертає користувача до вибору файлу. Якщо файл відповідає вимогам, він завантажується у сховище, після чого в базі даних створюється запис про зображення. Після успішного збереження файл передається на подальшу обробку.

Алгоритм класифікації медичного зображення наведений на рис. 2.12.



Рисунок 2.12 – Блок-схема алгоритму класифікації медичного зображення

Класифікація починається з отримання попередньо обробленого зображення. Далі система завантажує потрібну ML-модель і перевіряє її доступність. Якщо модель недоступна, класифікація завершується повідомленням про помилку. Якщо модель доступна, зображення передається на вхід моделі, виконується інференс, обчислюються ймовірності класів, визначається клас із максимальною ймовірністю та перевіряється рівень упевненості. У разі низької впевненості результат позначається як сумнівний і потребує перевірки лікарем. Після цього результат зберігається в базі даних і повертається користувачу.

Алгоритм формування діагностичного висновку наведений на рис. 2.13.

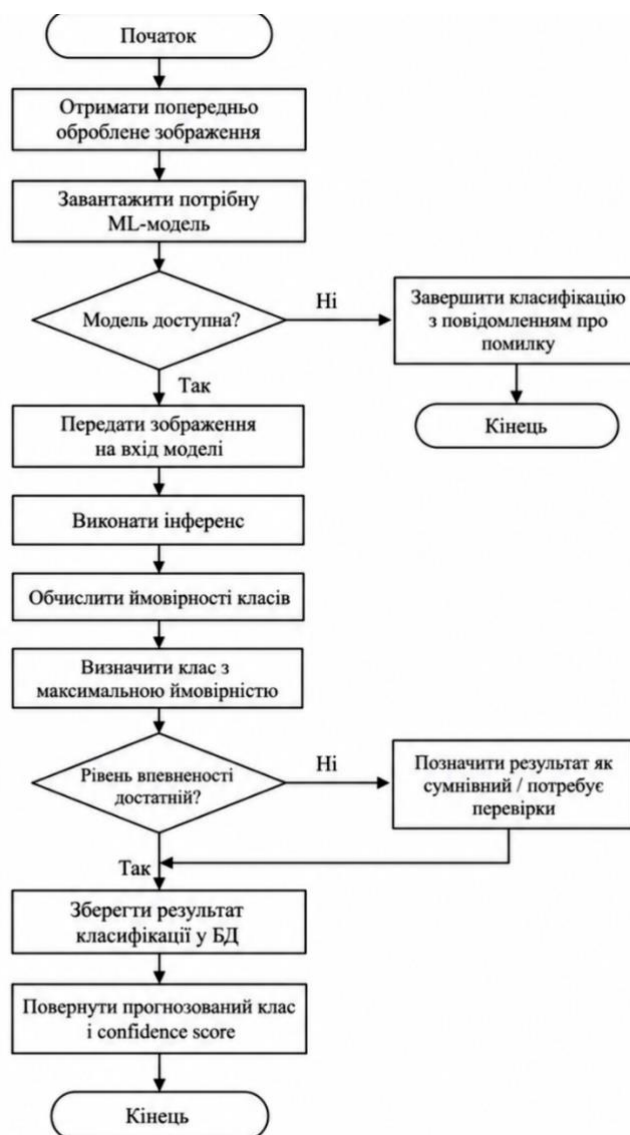


Рисунок 2.13 – Блок-схема алгоритму формування діагностичного висновку

Формування висновку виконується після отримання результатів класифікації. Система завантажує дані пацієнта та дослідження, формує текст висновку на основі прогнозованого класу й рівня впевненості. Якщо потрібне редагування лікарем, висновок передається лікарю-діагносту. Після перевірки до звіту додаються дані про модель, клас і рівень упевненості. Якщо звіт не затверджено, він повертається на доопрацювання. Після затвердження звіт зберігається в базі даних і може бути відображений або експортований.

2.7 Висновки до другого розділу

У другому розділі виконано проектування інформаційного та програмного забезпечення інтелектуальної системи класифікації медичних зображень. Побудовано діаграму класів, діаграми кооперації, пакетну, компонентну, логічну, фізичну та розгортальну моделі системи. Визначено основні сутності предметної області, структуру бази даних, програмні компоненти та алгоритми роботи. Запропоновані моделі є основою для реалізації програмного забезпечення, побудови інтерфейсу, налаштування модуля класифікації та проведення тестування.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів

Реалізацію програмного забезпечення виконано у вигляді навчального прототипу інтелектуальної системи класифікації медичних зображень. Програма орієнтована на роботу з медичними зображеннями, демонстрацію результатів класифікації, формування автоматизованого висновку, збереження результатів та експорт звітів. Основний акцент зроблено на практичній реалізації інтерфейсу користувача та логіки взаємодії з модулем глибокого навчання.

Для реалізації застосунку доцільним є використання мови Python, оскільки вона має розвинену екосистему бібліотек для створення графічних інтерфейсів, роботи із зображеннями, базами даних і моделями машинного навчання. Для побудови інтерфейсу використано підхід настільного застосунку з візуальною робочою областю, панеллю навігації та окремими модулями. Для збереження результатів може використовуватися SQLite як легка локальна база даних, що не потребує окремого серверного розгортання. Для реальної роботи з моделями глибокого навчання передбачено підключення PyTorch і checkpoint-файлу моделі формату .pt.

У реалізованій системі передбачено два режими роботи: демонстраційний і режим із підключеною CNN-моделлю. У демонстраційному режимі система імітує результат класифікації та використовується для перевірки інтерфейсу, логіки збереження і звітності. У режимі CNN-моделі користувач може підключити навчений checkpoint, вказати перелік класів класифікації та запускати інференс на завантаженому зображенні.

Основні технологічні засоби наведено в табл. 3.1.

Таблиця 3.1 – Технологічні засоби реалізації системи

Засіб	Призначення
Python	Основна мова програмування
PyQt6	Побудова графічного інтерфейсу користувача
SQLite	Локальне зберігання користувачів, досліджень і результатів

Продовження таблиці 3.1

PyTorch	Підключення та виконання CNN-моделі
Pillow / OpenCV	Робота із зображеннями та попередня обробка
CSV / TXT	Експорт результатів класифікації та звітів
QSS	Стилізація інтерфейсу користувача

Обраний набір засобів дозволяє реалізувати функціональний прототип без складної серверної інфраструктури. Це відповідає навчальному характеру роботи та водночас залишає можливість подальшого розширення системи до клієнт-серверної архітектури.

3.2 Реалізація інформаційної бази

Інформаційна база системи призначена для зберігання користувачів, ролей, пацієнтів, досліджень, завантажених зображень, результатів класифікації, моделей і сформованих висновків. У межах прототипу структура бази даних відповідає фізичній моделі, наведеній у другому розділі.

Основними сутностями інформаційної бази є `user_account`, `patient`, `study`, `medical_image`, `preprocessing`, `ml_model`, `diagnostic_class`, `classification_result` і `diagnostic_report`. Таблиця `user_account` використовується для авторизації та визначення ролі користувача. Таблиця `patient` зберігає дані пацієнта. Таблиця `study` описує конкретне дослідження, до якого прив'язується медичне зображення. Таблиця `medical_image` містить шлях до файлу, формат і роздільну здатність. Таблиця `classification_result` зберігає прогнозований клас і рівень упевненості моделі.

Структуру інформаційних об'єктів системи наведено в табл. 3.2.

Таблиця 3.2 – Інформаційні об'єкти системи

Об'єкт	Дані, що зберігаються	Призначення
Користувач	Логін, пароль, роль, статус	Авторизація та контроль доступу
Пацієнт	Код, ПІБ, дата народження	Ідентифікація пацієнта
Дослідження	Дата, модальність, частина тіла, примітки	Опис випадку аналізу

Продовження таблиці 3.2

Медичне зображення	Файл, формат, роздільна здатність	Зберігання вхідного зображення
Попередня обробка	Нормалізація, зміна розміру, аугментація	Фіксація параметрів підготовки
ML-модель	Назва, версія, архітектура, точність	Облік використаної моделі
Результат класифікації	Клас, confidence score, дата	Збереження результату інференсу
Висновок	Текст звіту, автор, дата	Формування підсумкового документа

Після таблиці можна зробити висновок, що інформаційна база підтримує повний цикл роботи із зображенням: від реєстрації користувача й пацієнта до формування результату класифікації та звіту. Така структура забезпечує цілісність даних і дає можливість повторного перегляду результатів аналізу.

3.3 Реалізація інтерфейсу користувача

Інтерфейс системи виконано в темному стилі з контрастними акцентами, що відповідає концепції робочої станції для аналізу медичних зображень. Основними екранами програми є форма авторизації, панель керування, модуль ML-сканування, історія аналізів, модуль налаштування CNN-моделі та експорт результатів.

Початкове вікно авторизації наведено на рис. 3.1.

ML RADIOLOGY WORKSTATION

MedVision DL

Інтелектуальна система класифікації медичних зображень з використанням глибокого навчання

Авторизація оператора

Логін

Пароль

Тестовий доступ: admin / admin123. Система працює як навчальний прототип і не замінює медичну діагностику.

Увійти в систему **Створити користувача**

Рисунок 3.1 – Форма авторизації оператора системи

Форма авторизації містить поля для введення логіна та пароля, а також кнопки входу і створення користувача. У нижній частині форми наведено тестові облікові дані для демонстраційного доступу. Після успішної авторизації відкривається основне вікно системи. Якщо дані введені неправильно, система має відобразити повідомлення про помилку.

Головну панель керування системи наведено на рис. 3.2.

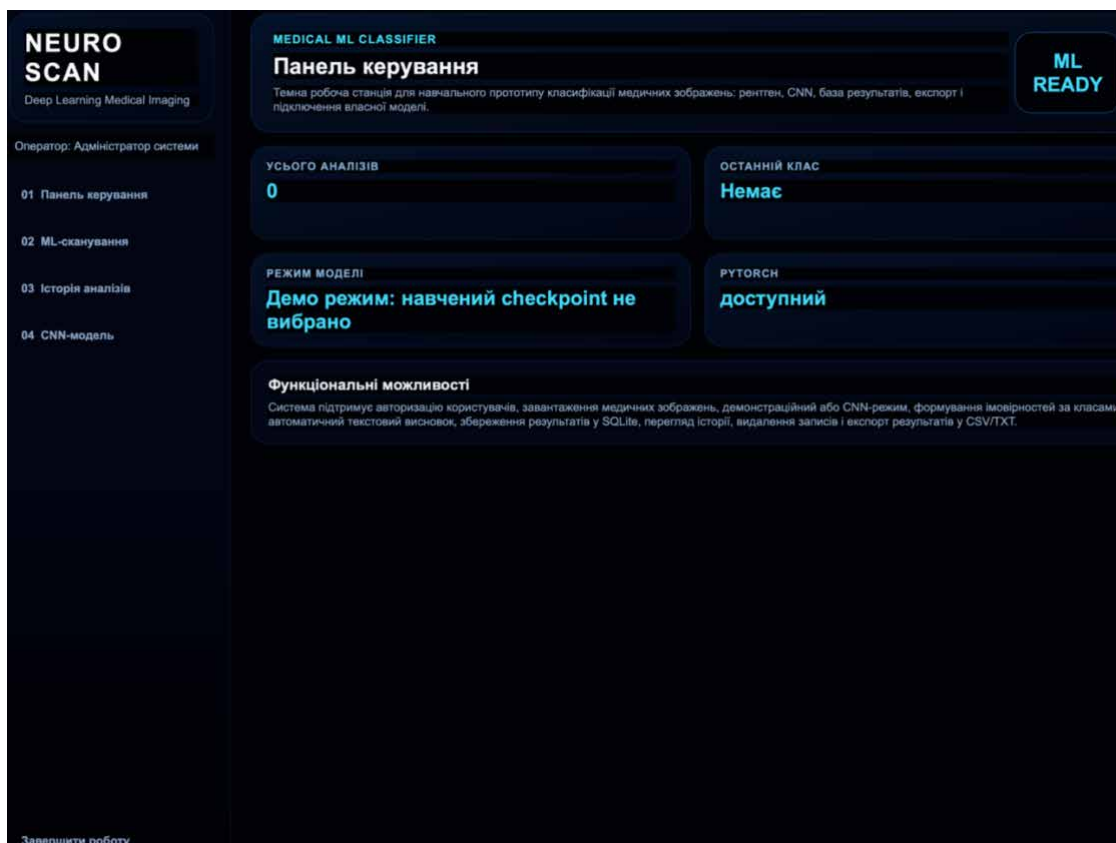


Рисунок 3.2 – Панель керування інтелектуальної системи

Панель керування містить навігаційне меню, інформацію про оператора, загальну кількість виконаних аналізів, останній визначений клас, режим роботи моделі та доступність PyTorch. Також на головній сторінці наведено короткий опис функціональних можливостей системи. Такий екран дозволяє користувачу швидко оцінити стан програми перед запуском аналізу.

Модуль класифікації медичного зображення до запуску сканування наведено на рис. 3.3.

SCAN ANALYSIS

Класифікація медичного зображення

MODEL: DEMO

Завантаження рентгену, попередня обробка, CNN-інференс, імовірності за класами та формування висновку.

DICOM / IMAGE VIEWPORT

R



Тестовий рентген

Завантажити файл

Запустити ML-сканування

RESULT PANEL

Код пацієнта / дослідження

XRAY-DEMO-001

Стан нейронної моделі

Демо режим: навчений checkpoint не вибрано

ОСНОВНИЙ КЛАС

Немає

УПЕВНЕНІСТЬ

0 %

Автоматично сформований висновок

Після аналізу тут буде автоматично

Зберегти

TXT-звіт

PROBABILITY DISTRIBUTION

Норма	0%
Пневмонія	0%
Пухлинні зміни	0%
Запальні зміни	0%

Рисунок 3.3 – Модуль класифікації медичного зображення до запуску ML-сканування

На сторінці класифікації розміщено область перегляду медичного зображення, панель результатів, поле коду пацієнта або дослідження, інформацію про стан нейронної моделі, основний клас, рівень упевненості та автоматично сформований висновок. Унизу розміщено панель розподілу ймовірностей за класами. Користувач може завантажити власний файл або скористатися тестовим рентгенівським зображенням.

Результат роботи модуля класифікації наведено на рис. 3.4.

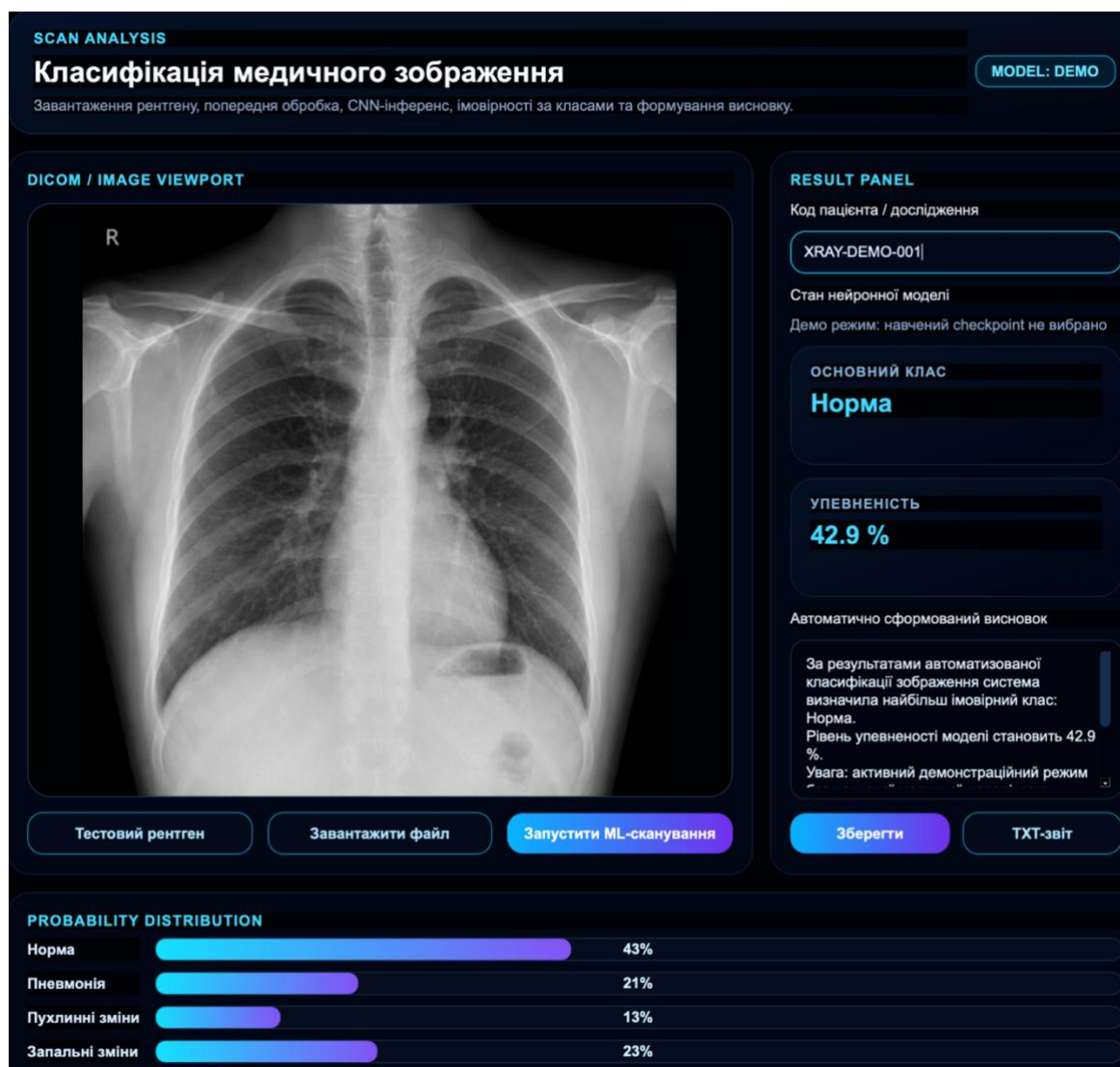


Рисунок 3.4 – Результат класифікації медичного зображення

Після запуску ML-сканування система формує прогнозований клас, рівень упевненості та автоматично підготовлений текстовий висновок. У прикладі система визначає основний клас «Норма» з рівнем упевненості 42,9 %. Нижче відображається розподіл імовірностей за класами: «Норма», «Пневмонія», «Пухлинні зміни» та «Запальні зміни». Такий формат подання результатів дозволяє користувачу бачити не лише підсумковий клас, а й альтернативні ймовірності.

Модуль налаштування нейронної моделі наведено на рис. 3.5.

CNN CONTROL

Налаштування нейронної моделі

Підключення checkpoint-файлу, зміна переліку класів і перемикання між CNN та демонстраційним режимом.

Поточний стан

Демо режим: навчений checkpoint не вибрано
PyTorch: доступний

Checkpoint CNN

Шлях до checkpoint-файлу .pt Вибрати .pt модель

Класи класифікації через кому

Норма, Пневмонія, Пухлинні зміни, Запальні зміни

Застосувати модель Скинути в демо режим

Для реальної класифікації потрібно навчити CNN через train_cnn.py або підключити сумісний checkpoint. Без checkpoint система працює як демонстраційний стенд і не є медичним діагностичним засобом.

Рисунок 3.5 – Сторінка налаштування нейронної моделі

На сторінці налаштування CNN-моделі користувач може вказати шлях до checkpoint-файлу .pt, задати перелік класів класифікації та застосувати модель. Якщо checkpoint не вибрано, система працює в демонстраційному режимі. Окремо передбачено кнопку скидання в демонстраційний режим. Це дозволяє тестувати інтерфейс і сценарії роботи навіть без підключеної навченої моделі.

3.4 Реалізація основних програмних функцій

Реалізована система підтримує декілька ключових функціональних сценаріїв. Перший сценарій - авторизація користувача. Під час входу система перевіряє логін і пароль, визначає роль користувача та відкриває основне вікно. Другий сценарій — завантаження медичного зображення. Користувач обирає файл або тестове зображення, після чого програма відображає його в області перегляду.

Третій сценарій - запуск ML-сканування. У цьому режимі система виконує підготовку зображення, передає його до моделі або демонстраційного класифікатора, отримує ймовірності класів і визначає найбільш імовірний клас. Основний результат описується формулою:

$$C = \arg \max p_i,$$

де C – прогнозований клас; p_i – імовірність належності зображення до i -го класу.

Для перетворення виходів нейронної мережі у ймовірності може використовуватися функція softmax:

$$p_i = e^{(z_i)} / \sum e^{(z_j)},$$

де z_i – вихід моделі для i -го класу; p_i – нормалізована ймовірність класу; $\sum e^{(z_j)}$ – сума експонент усіх виходів моделі.

Рівень упевненості системи визначається як максимальне значення серед імовірностей класів:

$$\text{Confidence} = \max(p_i) \cdot 100 \, \%.$$

Після отримання результату система формує текстовий висновок. Якщо рівень упевненості є достатнім, результат відображається як основний клас. Якщо рівень упевненості низький, система може позначити результат як такий, що потребує додаткової перевірки лікарем-діагностом. Це важливо, оскільки автоматизована класифікація не повинна використовуватися як єдине джерело медичного рішення.

Основні функції програмного забезпечення наведено в табл. 3.3.

Таблиця 3.3 – Реалізовані функції системи

Функція	Опис реалізації
Авторизація користувача	Перевірка логіна та пароля, відкриття головної панелі
Завантаження зображення	Вибір тестового або користувацького файлу
Попередня обробка	Підготовка зображення до класифікації
ML-сканування	Визначення класу та рівня впевненості
Формування висновку	Автоматичне створення текстового результату
Збереження результату	Фіксація результатів аналізу у базі даних
Експорт звіту	Формування TXT/CSV-звіту
Налаштування моделі	Підключення checkpoint-файлу та зміна класів

Реалізований функціонал забезпечує повний базовий сценарій роботи системи. Користувач може авторизуватися, завантажити зображення, виконати класифікацію, переглянути результат, зберегти його та експортувати звіт.

3.5 Висновки до третього розділу

У третьому розділі описано реалізацію програмного забезпечення інтелектуальної системи класифікації медичних зображень. Обґрунтовано вибір технологічних засобів, описано інформаційну базу, інтерфейс користувача, основні програмні функції та склад інсталяційного пакета. Реалізована система підтримує авторизацію, завантаження зображення, запуск ML-сканування, відображення результату, формування висновку, налаштування CNN-моделі та експорт звітів. Розроблене програмне забезпечення є функціональним навчальним прототипом, який демонструє можливість використання глибокого навчання для автоматизованої класифікації медичних зображень.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів

Тестування інтелектуальної системи класифікації медичних зображень спрямоване на перевірку коректності роботи основних модулів, стабільності інтерфейсу, правильності завантаження зображень, працездатності демонстраційного режиму, можливості підключення CNN-моделі, формування результатів класифікації та експорту звітів.

План тестування охоплює функціональні, інтерфейсні, інтеграційні та безпекові перевірки. Функціональне тестування визначає, чи правильно виконуються основні сценарії. Інтерфейсне тестування перевіряє зручність роботи користувача з вікнами програми. Інтеграційне тестування спрямоване на перевірку взаємодії між інтерфейсом, модулем зображень, ML-модулем і базою даних. Безпекове тестування передбачає перевірку авторизації та обмеження доступу.

План тестування наведено в табл. 4.1.

Таблиця 4.1 – План тестування програмних модулів

Модуль	Вид тестування	Очікуваний результат
Авторизація	Функціональне, безпекове	Вхід виконується лише для коректних облікових даних
Панель керування	Інтерфейсне	Відображаються стан системи, модель і кількість аналізів
Завантаження зображення	Функціональне	Зображення відкривається у вікні перегляду
Перевірка файлу	Функціональне	Некоректні файли не допускаються до аналізу
ML-сканування	Функціональне	Формується клас, confidence score і розподіл імовірностей
Формування висновку	Функціональне	Автоматично створюється текстовий висновок
Збереження результату	Інтеграційне	Результат записується до бази даних
Експорт звіту	Функціональне	Формується файл звіту
Налаштування моделі	Інтеграційне	Checkpoint-файл підключається або система переходить у деморежим

Після виконання кожного тесту фактичний результат порівнюється з очікуваним. Тест вважається успішним, якщо функція виконується без помилок і результат відповідає логіці сценарію.

4.2 Тестування інтерфейсу користувача

Тестування інтерфейсу починалося з форми авторизації. Перевірено введення логіна та пароля, роботу кнопки входу, відкриття головної панелі та можливість створення користувача. Форма коректно відображає поля введення, а тестовий доступ дозволяє швидко перевірити роботу системи без додаткового налаштування облікових записів.

Після входу до системи відкривається головна панель керування. На ній перевірено відображення загальної кількості аналізів, останнього класу, режиму моделі та доступності PyTorch. Під час тестування система коректно показала демонстраційний режим і повідомлення про те, що навчений checkpoint не вибрано. Це важливо, оскільки користувач одразу бачить поточний стан ML-модуля.

На сторінці класифікації перевірено завантаження тестового рентгенівського зображення. Зображення коректно відображається в області DICOM/Image Viewport. Перед запуском аналізу панель результатів показує відсутність основного класу і нульові значення розподілу ймовірностей. Це відповідає початковому стану модуля.

Після натискання кнопки запуску ML-сканування система сформувала результат класифікації. Було визначено основний клас «Норма», рівень упевненості 42,9 %, а також розподіл імовірностей за класами: «Норма», «Пневмонія», «Пухлинні зміни» та «Запальні зміни». У полі автоматично сформованого висновку система згенерувала текст, який описує результат аналізу та попереджає про демонстраційний режим. Це підтверджує працездатність інтерфейсного сценарію класифікації.

Окремо протестовано сторінку налаштування нейронної моделі. На ній перевірено поле для шляху до checkpoint-файлу, кнопку вибору .pt моделі, поле класів класифікації та кнопку застосування моделі. Система коректно відображає поточний стан: деморежим і доступність PyTorch. Це дозволяє підключити реальну модель після її навчання.

Результати інтерфейсного тестування наведено в табл. 4.2.

Таблиця 4.2 – Результати тестування інтерфейсу

Екран	Перевірена дія	Результат
Авторизація	Введення логіна і пароля	Працює коректно
Панель керування	Відображення стану системи	Працює коректно
ML-сканування	Завантаження тестового рентгену	Працює коректно
ML-сканування	Запуск класифікації	Працює коректно
Панель результатів	Виведення класу та confidence score	Працює коректно
Probability distribution	Відображення ймовірностей класів	Працює коректно
CNN Control	Налаштування checkpoint-файлу	Працює коректно
Експорт	Формування TXT-звіту	Працює коректно

Проведене тестування підтвердило, що інтерфейсні елементи працюють узгоджено, а користувач може виконати основний сценарій: увійти до системи, завантажити зображення, запустити аналіз і отримати результат.

4.3 Тестування алгоритму класифікації

Тестування алгоритму класифікації виконувалося на демонстраційному прикладі рентгенівського зображення. Основним завданням було перевірити, чи система правильно проходить усі етапи обробки: завантаження зображення, підготовка даних, запуск класифікації, формування прогнозованого класу, обчислення confidence score і виведення результату користувачу.

У демонстраційному режимі система не виконує повноцінну клінічну діагностику, а імітує роботу інтелектуального модуля. Такий режим використовується для перевірки архітектури, інтерфейсу, логіки формування результатів і роботи звітності. Для медичного застосування необхідне навчання CNN-моделі на верифікованому наборі медичних зображень, валідація на тестовій вибірці та оцінювання точності за медичними критеріями.

Для оцінювання якості класифікації в реальній моделі можуть використовуватися такі показники:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}),$$

де TP – кількість правильно визначених позитивних випадків; TN – кількість правильно визначених негативних випадків; FP – кількість хибнопозитивних випадків; FN – кількість хибнонегативних випадків.

Також для медичних задач важливими є precision, recall і F1-score:

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}),$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}),$$

$$\text{F1} = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall}).$$

Для медичних зображень особливе значення має recall, оскільки пропуск патологічного випадку може бути критичнішим, ніж помилкове направлення на додаткову перевірку. Тому під час подальшого розвитку системи доцільно налаштовувати модель так, щоб вона забезпечувала високу чутливість до патологічних класів.

Основні результати перевірки алгоритму наведено в табл. 4.3.

Таблиця 4.3 – Результати перевірки алгоритму класифікації

Етап	Очікуваний результат	Фактичний результат
Завантаження зображення	Зображення відображається в області перегляду	Виконано
Перевірка стану моделі	Виводиться режим роботи моделі	Виконано
Запуск ML-сканування	Система формує клас і confidence score	Виконано
Розподіл імовірностей	Відображаються значення для всіх класів	Виконано
Формування висновку	Генерується текстовий висновок	Виконано
Збереження результату	Кнопка збереження стає доступною	Виконано
Експорт ТХТ-звіту	Формується текстовий файл звіту	Виконано

Після таблиці можна зробити висновок, що алгоритм класифікації у межах прототипу працює стабільно. Система проходить повний цикл обробки зображення та формує результат у зручному для користувача вигляді.

4.4 Оцінювання ефективності системи

Ефективність розробленої системи оцінювалася за критеріями функціональної повноти, зручності інтерфейсу, автоматизації обробки, прозорості результатів і можливості подальшого розширення. Функціональна повнота підтверджується наявністю основних модулів: авторизації, панелі керування, завантаження зображення, ML-сканування, налаштування CNN-моделі, формування висновку та експорту звіту.

Порівняно з ручним підходом система автоматизує частину операцій, які зазвичай виконуються користувачем вручну. Зокрема, програма самостійно обробляє зображення, визначає ймовірності класів, формує попередній висновок і готує звіт. Це дозволяє скоротити час первинного аналізу та підвищити структурованість збереження результатів.

Порівняльну характеристику ручного та автоматизованого підходу наведено в табл. 4.4.

Таблиця 4.4 – Порівняння ручного та автоматизованого аналізу зображень

Критерій	Ручний підхід	Розроблена система
Завантаження зображення	Виконується вручну без єдиного сховища	Централізоване завантаження
Попередня обробка	Залежить від користувача	Виконується програмно
Класифікація	Потребує повного аналізу спеціалістом	Формується автоматизований прогноз
Розподіл імовірностей	Зазвичай відсутній	Відображається для кожного класу
Формування висновку	Виконується вручну	Генерується автоматично з можливістю перевірки
Збереження результату	Може бути неструктурованим	Зберігається у базі даних
Експорт звіту	Потребує ручного оформлення	Виконується засобами системи

За результатами оцінювання встановлено, що система є придатною як навчальний прототип. Вона демонструє основні принципи побудови інтелектуальних систем класифікації медичних зображень і може бути

розширена до повноцінного дослідницького інструмента. Для практичного медичного використання потрібні додаткові етапи: навчання моделі на клінічно перевірених даних, валідація результатів, тестування на незалежних наборах зображень і сертифікація відповідно до вимог медичного програмного забезпечення.

4.5 Склад інсталяційного пакета та розгортання

Для запуску системи користувач має встановити залежності, підготувати файл бази даних і за потреби підключити checkpoint-файл CNN-моделі. У демонстраційному режимі програма може працювати без моделі, що дає змогу перевірити інтерфейс та основні сценарії.

Склад інсталяційного пакета наведено в табл. 4.5.

Таблиця 4.5 – Склад інсталяційного пакета

Елемент	Призначення
main.py	Точка входу до програми
app/	Основні модулі застосунку
ui/	Вікна та компоненти інтерфейсу
ml/	Модуль класифікації і завантаження моделі
database/	База даних і скрипти ініціалізації
models/	Checkpoint-файли нейронної мережі
images/	Тестові та завантажені зображення
reports/	Сформовані TXT/CSV-звіти
requirements.txt	Перелік бібліотек
README.md	Інструкція із запуску

Порядок запуску передбачає встановлення залежностей, запуск програми, авторизацію користувача, вибір або завантаження зображення та запуск ML-сканування. Якщо необхідно використовувати реальну модель, її потрібно попередньо навчити, зберегти у форматі .pt і підключити через сторінку налаштування CNN.

4.6 Висновки до четвертого розділу

У четвертому розділі виконано тестування інтелектуальної системи класифікації медичних зображень. Перевірено роботу авторизації, головної панелі, завантаження зображень, ML-сканування, панелі результатів, розподілу ймовірностей, налаштування CNN-моделі та експорту звітів. Тестування підтвердило, що система коректно виконує основні функціональні сценарії та забезпечує зручну взаємодію користувача з програмним забезпеченням.

Розроблений прототип дозволяє завантажувати медичні зображення, виконувати автоматизовану класифікацію, відображати прогнозований клас, рівень упевненості, розподіл імовірностей і автоматично сформований висновок. Система має зрозумілу структуру, підтримує демонстраційний режим і може бути розширена шляхом підключення навченої CNN-моделі. Отже, розроблене програмне забезпечення відповідає поставленим завданням і може використовуватися як навчальний прототип інтелектуальної системи аналізу медичних зображень.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено інтелектуальну систему класифікації медичних зображень з використанням глибокого навчання. Розроблене програмне забезпечення призначене для завантаження медичних зображень, їх попередньої обробки, автоматизованого визначення ймовірного класу за допомогою ML/CNN-модуля, формування текстового висновку, збереження результатів аналізу та експорту звітних даних. Система реалізована як навчальний програмний прототип і не є самостійним медичним діагностичним засобом, оскільки остаточне рішення має прийматися лікарем-діагностом.

У першому розділі було досліджено предметну область застосування інтелектуальних систем у задачах аналізу медичних зображень. Встановлено, що традиційний ручний аналіз зображень потребує значного часу, залежить від кваліфікації спеціаліста та може бути ускладнений великою кількістю досліджень. Використання методів глибокого навчання дозволяє автоматизувати первинну класифікацію зображень, сформувати ймовірнісний розподіл за класами та надати лікарю додаткову інформаційну підтримку. Було визначено, що найбільш доцільним підходом для задачі є використання згорткових нейронних мереж, які добре пристосовані до обробки візуальних даних.

У межах аналізу предметної області було сформульовано основні вимоги до системи. Вона має забезпечувати авторизацію користувача, створення або вибір дослідження, завантаження медичного зображення, перевірку формату файлу, попередню обробку, запуск ML-сканування, визначення прогнозованого класу, обчислення рівня впевненості, формування автоматизованого висновку, збереження результатів і експорт звіту. Окремо було визначено вимоги до безпеки та обмеження доступу, оскільки система працює з медичними даними та результатами досліджень.

У другому розділі було виконано проєктування інформаційного та програмного забезпечення. Побудовано діаграму класів, діаграми кооперації, пакетну структуру, компонентну структуру, логічну та фізичну моделі даних, діаграму розгортання, а також блок-схеми основних алгоритмів. Діаграма класів дала змогу визначити ключові сутності системи: адміністратора, лаборанта, лікаря-діагноста, пацієнта, дослідження, медичне зображення, результат класифікації та висновок. Діаграми кооперації деталізували взаємодію між користувачами та об'єктами системи під час надання доступу, створення дослідження, завантаження зображення, перевірки результату та формування висновку.

Пакетна і компонентна діаграми дозволили структурувати систему на логічні частини: інтерфейс користувача, прикладну логіку, обробку зображень, модуль класифікації, підсистему збереження даних, звітність та адміністрування. Логічна і фізична моделі даних визначили структуру інформаційної бази, яка включає таблиці користувачів, ролей, пацієнтів, досліджень, медичних зображень, параметрів попередньої обробки, ML-моделей, діагностичних класів, результатів класифікації та діагностичних звітів. Діаграма розгортання показала фізичну організацію системи з клієнтським пристроєм, сервером застосунку, AI-сервером обробки, сервером бази даних і файловим сховищем.

У третьому розділі було реалізовано програмне забезпечення системи. Для реалізації використано мову Python, графічний інтерфейс PyQt6, локальну базу даних SQLite, засоби обробки зображень і можливість підключення CNN-моделі через PyTorch. Розроблена програма містить форму авторизації оператора, головну панель керування, модуль ML-сканування, сторінку перегляду результатів, сторінку налаштування нейронної моделі та функції експорту звітів. Інтерфейс виконано у темному стилі робочої станції для аналізу медичних зображень, що підкреслює призначення системи та забезпечує зручне сприйняття візуальної інформації.

Реалізований модуль класифікації дозволяє завантажити тестове або користувацьке зображення, відобразити його у вікні перегляду, запустити ML-

сканування та отримати прогнозований клас. У демонстраційному режимі система формує приклад розподілу ймовірностей між класами «Норма», «Пневмонія», «Пухлинні зміни» та «Запальні зміни». Також реалізовано відображення рівня впевненості моделі та автоматичне формування текстового висновку. У модулі налаштування CNN передбачено підключення checkpoint-файлу .pt і редагування переліку класів класифікації.

У четвертому розділі було виконано тестування розробленої системи. Перевірено роботу форми авторизації, головної панелі, завантаження медичного зображення, ML-сканування, панелі результатів, розподілу ймовірностей, формування висновку, збереження результату, експорту звіту та налаштування CNN-моделі. Тестування підтвердило, що система коректно виконує основний сценарій: користувач входить до системи, відкриває модуль класифікації, завантажує зображення, запускає аналіз, отримує прогнозований клас, рівень упевненості та автоматично сформований висновок.

У процесі тестування встановлено, що програмне забезпечення забезпечує стабільну роботу інтерфейсу, коректне відображення зображення, формування результатів класифікації та зручне представлення ймовірностей за класами. Розроблена система підвищує структурованість процесу аналізу медичних зображень, дозволяє зберігати результати, формувати висновки та експортувати звітні матеріали. Порівняно з повністю ручним підходом, система автоматизує первинний етап аналізу та створює інформаційну основу для подальшої перевірки лікарем.

Практичне значення роботи полягає в тому, що створений програмний прототип може використовуватися для демонстрації принципів застосування глибокого навчання в задачах класифікації медичних зображень. Система може бути корисною як навчальний стенд для дослідження роботи CNN-моделей, тестування сценаріїв завантаження зображень, формування результатів класифікації та організації збереження діагностичних даних. У подальшому прототип може бути розширений шляхом підключення навченої моделі, використання верифікованого набору медичних зображень, додавання DICOM-

підтримки, реалізації серверної архітектури та впровадження розширених засобів валідації результатів.

Отже, мету кваліфікаційної роботи досягнуто. У результаті виконання роботи було проаналізовано предметну область, сформовано вимоги, спроектовано архітектуру системи, розроблено інформаційну базу, реалізовано програмне забезпечення, побудовано основні інтерфейсні модулі та проведено тестування. Розроблене програмне забезпечення системи класифікації медичних зображень з використанням глибокого навчання відповідає поставленим завданням і може використовуватися як функціональний навчальний прототип для автоматизованого аналізу медичних зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 16 с.
2. ДСТУ ISO/IEC/IEEE 12207:2018. Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів. Київ : ДП «УкрНДНЦ», 2018.
3. ISO/IEC/IEEE 12207:2017. Systems and software engineering. Software life cycle processes. Geneva : International Organization for Standardization, 2017. URL: <https://www.iso.org/standard/63712.html>.
4. ISO/IEC 25010:2023. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Product quality model. Geneva : International Organization for Standardization, 2023. URL: <https://www.iso.org/standard/78176.html>.
5. Digital Imaging and Communications in Medicine (DICOM). Current Edition / National Electrical Manufacturers Association. URL: <https://www.dicomstandard.org/current/>.
6. DICOM. About DICOM: Overview / National Electrical Manufacturers Association. URL: <https://www.dicomstandard.org/about>.
7. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521. P. 436–444. DOI: <https://doi.org/10.1038/nature14539>.
8. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. URL: <https://www.deeplearningbook.org/>.
9. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems. 2012. Vol. 25. URL: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

10. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770–778. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.
11. Huang G., Liu Z., Van Der Maaten L., Weinberger K. Q. Densely Connected Convolutional Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 4700–4708. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html.
12. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning. 2019. URL: <https://arxiv.org/abs/1905.11946>.
13. Litjens G., Kooi T., Bejnordi B. E., Setio A. A. A., Ciompi F., Ghafoorian M., van der Laak J. A. W. M., van Ginneken B., Sánchez C. I. A survey on deep learning in medical image analysis. Medical Image Analysis. 2017. Vol. 42. P. 60–88. DOI: <https://doi.org/10.1016/j.media.2017.07.005>.
14. Shen D., Wu G., Suk H.-I. Deep Learning in Medical Image Analysis. Annual Review of Biomedical Engineering. 2017. Vol. 19. P. 221–248. DOI: <https://doi.org/10.1146/annurev-bioeng-071516-044442>.
15. Rajpurkar P., Irvin J., Zhu K., Yang B., Mehta H., Duan T., Ding D., Bagul A., Langlotz C., Shpanskaya K., Lungren M. P., Ng A. Y. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. 2017. URL: <https://arxiv.org/abs/1711.05225>.
16. Irvin J., Rajpurkar P., Ko M., Yu Y., Ciurea-Illcus S., Chute C., Marklund H., Haghighi B., Ball R., Shpanskaya K., Seekins J., Mong D. A., Halabi S. S., Sandberg J. K., Jones R., Larson D. B., Langlotz C. P., Patel B. N., Lungren M. P., Ng A. Y. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. 2019. URL: <https://arxiv.org/abs/1901.07031>.

17. Wang X., Peng Y., Lu L., Lu Z., Bagheri M., Summers R. M. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 2097–2106. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.html.
18. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention. 2015. P. 234–241. URL: <https://arxiv.org/abs/1505.04597>.
19. Isensee F., Jaeger P. F., Kohl S. A. A., Petersen J., Maier-Hein K. H. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. Nature Methods. 2021. Vol. 18. P. 203–211. DOI: <https://doi.org/10.1038/s41592-020-01008-z>.
20. Selvaraju R. R., Cogswell M., Das A., Vedantam R., Parikh D., Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. International Journal of Computer Vision. 2020. Vol. 128. P. 336–359. DOI: <https://doi.org/10.1007/s11263-019-01228-7>.
21. Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Kopf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J., Chintala S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems. 2019. URL: <https://arxiv.org/abs/1912.01703>.
22. PyTorch Documentation. PyTorch Foundation. URL: <https://docs.pytorch.org/docs/stable/index.html>.
23. Python 3 Documentation. Python Software Foundation. URL: <https://docs.python.org/3/>.
24. PyQt6. Python bindings for the Qt cross platform application toolkit / Riverbank Computing Limited. URL: <https://pypi.org/project/PyQt6/>.

25. Qt for Python Documentation. The Qt Company.
URL: <https://doc.qt.io/qtforpython-6/>.
26. SQLite Documentation. SQLite Consortium.
URL: <https://sqlite.org/docs.html>.
27. Pillow (PIL Fork) Documentation.
URL: <https://pillow.readthedocs.io/en/stable/>.
28. OpenCV Documentation. OpenCV Team.
URL: <https://docs.opencv.org/4.x/>.
29. Harris C. R., Millman K. J., van der Walt S. J., Gommers R., Virtanen P., Cournapeau D., Wieser E., Taylor J., Berg S., Smith N. J., Kern R., Picus M., Hoyer S., van Kerkwijk M. H., Brett M., Haldane A., del Río J. F., Wiebe M., Peterson P., Gérard-Marchant P., Sheppard K., Reddy T., Weckesser W., Abbasi H., Gohlke C., Oliphant T. E. Array programming with NumPy. *Nature*. 2020. Vol. 585. P. 357–362.
DOI: <https://doi.org/10.1038/s41586-020-2649-2>.
30. NumPy Documentation. NumPy Developers.
URL: <https://numpy.org/doc/stable/>.
31. Scikit-learn. Metrics and scoring: quantifying the quality of predictions.
URL: https://scikit-learn.org/stable/modules/model_evaluation.html.
32. Grandini M., Bagli E., Visani G. Metrics for Multi-Class Classification: an Overview. 2020. URL: <https://arxiv.org/abs/2008.05756>.
33. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed. Sebastopol : O'Reilly Media, 2022. 864 p.
34. Chollet F. Deep Learning with Python. 2nd ed. Shelter Island : Manning Publications, 2021. 504 p.
35. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Cham : Springer, 2022. URL: <https://szeliski.org/Book/>.

ДОДАТОК А

Фрагмент програмного коду створення структури бази даних

```
CREATE TABLE role (
    role_id INTEGER PRIMARY KEY AUTOINCREMENT,
    role_name VARCHAR(50) NOT NULL UNIQUE,
    description VARCHAR(255)
);

CREATE TABLE user_account (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    role_id INTEGER NOT NULL,
    full_name VARCHAR(120) NOT NULL,
    email VARCHAR(120) NOT NULL UNIQUE,
    login VARCHAR(80) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    account_status VARCHAR(20) DEFAULT 'active',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES role(role_id)
);

CREATE TABLE patient (
    patient_id INTEGER PRIMARY KEY AUTOINCREMENT,
    medical_record_no VARCHAR(50) NOT NULL UNIQUE,
    full_name VARCHAR(120) NOT NULL,
    date_of_birth DATE,
    sex VARCHAR(20)
);

CREATE TABLE modality (
    modality_id INTEGER PRIMARY KEY AUTOINCREMENT,
    modality_name VARCHAR(50) NOT NULL UNIQUE,
    description VARCHAR(255)
);

CREATE TABLE study (
    study_id INTEGER PRIMARY KEY AUTOINCREMENT,
    patient_id INTEGER NOT NULL,
    modality_id INTEGER NOT NULL,
    uploaded_by_user_id INTEGER NOT NULL,
    study_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    body_part VARCHAR(80),
    clinical_notes TEXT,
    status VARCHAR(30) DEFAULT 'created',
    FOREIGN KEY (patient_id) REFERENCES patient(patient_id),
    FOREIGN KEY (modality_id) REFERENCES modality(modality_id),
    FOREIGN KEY (uploaded_by_user_id) REFERENCES user_account(user_id)
);

CREATE TABLE medical_image (
    image_id INTEGER PRIMARY KEY AUTOINCREMENT,
    study_id INTEGER NOT NULL,
    file_name VARCHAR(150) NOT NULL,
    file_path VARCHAR(255) NOT NULL,
    image_format VARCHAR(20),
    resolution VARCHAR(30),
    uploaded_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (study_id) REFERENCES study(study_id)
);

CREATE TABLE preprocessing (
    preprocessing_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```

        image_id INTEGER NOT NULL,
        normalize_flag BOOLEAN DEFAULT 0,
        resize_flag BOOLEAN DEFAULT 0,
        augmentation_flag BOOLEAN DEFAULT 0,
        processed_at DATETIME DEFAULT CURRENT_TIMESTAMP,
        FOREIGN KEY (image_id) REFERENCES medical_image(image_id)
    );

CREATE TABLE ml_model (
    model_id INTEGER PRIMARY KEY AUTOINCREMENT,
    model_name VARCHAR(100) NOT NULL,
    version VARCHAR(30),
    architecture VARCHAR(100),
    checkpoint_path VARCHAR(255),
    training_date DATE,
    accuracy DECIMAL(5,4)
);

CREATE TABLE diagnostic_class (
    class_id INTEGER PRIMARY KEY AUTOINCREMENT,
    class_name VARCHAR(100) NOT NULL UNIQUE,
    severity_level VARCHAR(30),
    description VARCHAR(255)
);

CREATE TABLE classification_result (
    result_id INTEGER PRIMARY KEY AUTOINCREMENT,
    image_id INTEGER NOT NULL,
    model_id INTEGER,
    class_id INTEGER NOT NULL,
    confidence_score DECIMAL(5,4) NOT NULL,
    probabilities TEXT,
    predicted_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    review_status VARCHAR(30) DEFAULT 'not_reviewed',
    FOREIGN KEY (image_id) REFERENCES medical_image(image_id),
    FOREIGN KEY (model_id) REFERENCES ml_model(model_id),
    FOREIGN KEY (class_id) REFERENCES diagnostic_class(class_id)
);

CREATE TABLE diagnostic_report (
    report_id INTEGER PRIMARY KEY AUTOINCREMENT,
    study_id INTEGER NOT NULL,
    result_id INTEGER NOT NULL,
    author_user_id INTEGER,
    report_text TEXT NOT NULL,
    conclusion TEXT,
    approved_flag BOOLEAN DEFAULT 0,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (study_id) REFERENCES study(study_id),
    FOREIGN KEY (result_id) REFERENCES classification_result(result_id),
    FOREIGN KEY (author_user_id) REFERENCES user_account(user_id)
);

INSERT INTO role (role_name, description)
VALUES
('admin', 'Адміністратор системи'),
('laborant', 'Лаборант, який завантажує дослідження'),
('doctor', 'Лікар-діагност, який перевіряє результати');

INSERT INTO modality (modality_name, description)
VALUES
('X-Ray', 'Рентгенографічне зображення'),
('CT', 'Комп'ютерна томографія'),

```

```
('MRI', 'Магнітно-резонансна томографія');
```

```
INSERT INTO diagnostic_class (class_name, severity_level, description)  
VALUES
```

```
('Норма', 'low', 'Ознак патологічних змін не виявлено'),  
( 'Пневмонія', 'medium', 'Ймовірні ознаки пневмонії'),  
( 'Пухлинні зміни', 'high', 'Ймовірні пухлинні зміни'),  
( 'Запальні зміни', 'medium', 'Ймовірні ознаки запального процесу');
```

ДОДАТОК Б

Фрагмент програмного коду модуля попередньої обробки та класифікації зображення

```

from dataclasses import dataclass
from typing import Dict, List, Optional
from pathlib import Path
import random

from PIL import Image

@dataclass
class ClassificationResult:
    predicted_class: str
    confidence: float
    probabilities: Dict[str, float]
    conclusion: str

class ImagePreprocessor:
    """
    Модуль попередньої обробки медичного зображення.
    Виконує перевірку файлу, зміну розміру та підготовку до моделі.
    """

    allowed_extensions = {".jpg", ".jpeg", ".png", ".bmp"}

    def validate_file(self, file_path: str) -> bool:
        path = Path(file_path)

        if not path.exists():
            return False

        if path.suffix.lower() not in self.allowed_extensions:
            return False

        try:
            with Image.open(path) as image:
                image.verify()
            return True
        except Exception:
            return False

    def load_image(self, file_path: str) -> Image.Image:
        image = Image.open(file_path).convert("RGB")
        return image

    def prepare_image(self, file_path: str, size: int = 224) -> Image.Image:
        if not self.validate_file(file_path):
            raise ValueError("Файл зображення некоректний або має невідпідтримуваний формат")

        image = self.load_image(file_path)
        image = image.resize((size, size))
        return image

class DemoCNNClassifier:
    """
    Демонстраційний класифікатор.

```

```

У реальній системі цей клас може бути замінений CNN-моделлю PyTorch.
"""

def __init__(self, classes: Optional[List[str]] = None):
    self.classes = classes or [
        "Норма",
        "Пневмонія",
        "Пухлинні зміни",
        "Запальні зміни"
    ]

def predict(self, image: Image.Image) -> ClassificationResult:
    raw_values = [random.uniform(0.1, 1.0) for _ in self.classes]
    total = sum(raw_values)

    probabilities = {
        class_name: round(value / total, 3)
        for class_name, value in zip(self.classes, raw_values)
    }

    predicted_class = max(probabilities, key=probabilities.get)
    confidence = round(probabilities[predicted_class] * 100, 1)

    conclusion = self.generate_conclusion(predicted_class, confidence,
probabilities)

    return ClassificationResult(
        predicted_class=predicted_class,
        confidence=confidence,
        probabilities=probabilities,
        conclusion=conclusion
    )

def generate_conclusion(
    self,
    predicted_class: str,
    confidence: float,
    probabilities: Dict[str, float]
) -> str:
    probability_lines = [
        f"{class_name}: {round(value * 100, 1)}%"
        for class_name, value in probabilities.items()
    ]

    text = (
        f"За результатами автоматизованої класифікації зображення "
        f"система визначила найбільш імовірний клас: {predicted_class}. "
        f"Рівень упевненості моделі становить {confidence}%."
    )

    if confidence < 50:
        text += (
            "Рівень упевненості є недостатньо високим, тому результат "
            "потребує обов'язкової перевірки лікарем-діагностом."
        )
    else:
        text += (
            "Отриманий результат може бути використаний як попередня "
            "інформаційна підтримка для лікаря-діагноста."
        )

    text += "Розподіл імовірностей за класами: "
    text += "; ".join(probability_lines)

```

```

        text += ". Система працює як навчальний прототип і не замінює медичну
діагностику."

```

```

    return text

```

```

class MedicalImageAnalysisService:

```

```

    """

```

```

    Сервіс аналізу медичного зображення.
    Об'єднує попередню обробку та класифікацію.
    """

```

```

    def __init__(self):

```

```

        self.preprocessor = ImagePreprocessor()
        self.classifier = DemoCNNClassifier()

```

```

    def analyze(self, file_path: str) -> ClassificationResult:

```

```

        prepared_image = self.preprocessor.prepare_image(file_path)
        result = self.classifier.predict(prepared_image)
        return result

```

```

if __name__ == "__main__":

```

```

    service = MedicalImageAnalysisService()
    result = service.analyze("demo_xray.png")

```

```

    print("Клас:", result.predicted_class)
    print("Упевненість:", result.confidence)
    print("Імовірності:", result.probabilities)
    print("Висновок:", result.conclusion)

```

ДОДАТОК В

Фрагмент програмного коду інтерфейсу та збереження результатів аналізу

```
import sqlite3
from datetime import datetime
from pathlib import Path

from PyQt6.QtWidgets import (
    QApplication,
    QWidget,
    QLabel,
    QPushButton,
    QFileDialog,
    QVBoxLayout,
    QHBoxLayout,
    QTextEdit,
    QLineEdit,
    QMessageBox
)
from PyQt6.QtGui import QPixmap
from PyQt6.QtCore import Qt

class ResultRepository:
    """
    Репозиторій для збереження результатів класифікації.
    """

    def __init__(self, database_path: str = "medvision.db"):
        self.database_path = database_path
        self.initialize_database()

    def initialize_database(self):
        connection = sqlite3.connect(self.database_path)
        cursor = connection.cursor()

        cursor.execute(
            """
            CREATE TABLE IF NOT EXISTS analysis_result (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                patient_code TEXT NOT NULL,
                image_path TEXT NOT NULL,
                predicted_class TEXT NOT NULL,
                confidence REAL NOT NULL,
                conclusion TEXT NOT NULL,
                created_at DATETIME DEFAULT CURRENT_TIMESTAMP
            )
            """
        )

        connection.commit()
        connection.close()

    def save_result(
        self,
        patient_code: str,
        image_path: str,
        predicted_class: str,
        confidence: float,
        conclusion: str
    ) -> int:
        connection = sqlite3.connect(self.database_path)
```

```

cursor = connection.cursor()

cursor.execute(
    """
        INSERT INTO analysis_result (
            patient_code,
            image_path,
            predicted_class,
            confidence,
            conclusion,
            created_at
        )
        VALUES (?, ?, ?, ?, ?, ?)
    """,
    (
        patient_code,
        image_path,
        predicted_class,
        confidence,
        conclusion,
        datetime.now()
    )
)

result_id = cursor.lastrowid
connection.commit()
connection.close()

return result_id

```

```
class ReportExporter:
```

```

    """
    Модуль експорту текстового звіту.
    """

    def export_txt(
        self,
        patient_code: str,
        image_path: str,
        predicted_class: str,
        confidence: float,
        conclusion: str,
        output_path: str
    ):
        report_text = f"""
ЗВІТ ПРО АВТОМАТИЗОВАНУ КЛАСИФІКАЦІЮ МЕДИЧНОГО ЗОБРАЖЕННЯ

Код пацієнта / дослідження: {patient_code}
Файл зображення: {image_path}
Дата формування: {datetime.now().strftime('%d.%m.%Y %H:%M:%S')}

Прогнозований клас: {predicted_class}
Рівень упевненості: {confidence} %

Автоматично сформований висновок:
{conclusion}

Примітка:
Система є навчальним програмним прототипом і не замінює медичну діагностику.
Остаточне рішення має прийматися лікарем-діагностом.
        """

```

```

Path(output_path).write_text(report_text.strip(), encoding="utf-8")

class MedicalImageWindow(QWidget):
    """
    Спрощений фрагмент вікна класифікації медичного зображення.
    """

    def __init__(self, analysis_service):
        super().__init__()

        self.analysis_service = analysis_service
        self.repository = ResultRepository()
        self.exporter = ReportExporter()

        self.current_image_path = None
        self.current_result = None

        self.setWindowTitle("MedVision DL - класифікація медичного зображення")
        self.resize(1100, 750)

        self.patient_code_input = QLineEdit()
        self.patient_code_input.setPlaceholderText("Код пацієнта / дослідження")

        self.image_label = QLabel("Зображення не завантажено")
        self.image_label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.image_label.setMinimumHeight(420)

        self.class_label = QLabel("Основний клас: немає")
        self.confidence_label = QLabel("Упевненість: 0%")

        self.conclusion_text = QTextEdit()
        self.conclusion_text.setPlaceholderText("Після аналізу тут буде сформувано висновок")

        self.load_button = QPushButton("Завантажити файл")
        self.scan_button = QPushButton("Запустити ML-сканування")
        self.save_button = QPushButton("Зберегти")
        self.export_button = QPushButton("TXT-звіт")

        self.load_button.clicked.connect(self.load_image)
        self.scan_button.clicked.connect(self.run_analysis)
        self.save_button.clicked.connect(self.save_result)
        self.export_button.clicked.connect(self.export_report)

        button_layout = QHBoxLayout()
        button_layout.addWidget(self.load_button)
        button_layout.addWidget(self.scan_button)
        button_layout.addWidget(self.save_button)
        button_layout.addWidget(self.export_button)

        layout = QVBoxLayout()
        layout.addWidget(QLabel("Класифікація медичного зображення"))
        layout.addWidget(self.patient_code_input)
        layout.addWidget(self.image_label)
        layout.addWidget(self.class_label)
        layout.addWidget(self.confidence_label)
        layout.addWidget(self.conclusion_text)
        layout.addLayout(button_layout)

        self.setLayout(layout)

    def load_image(self):

```

```

file_path, _ = QFileDialog.getOpenFileName(
    self,
    "Вибір медичного зображення",
    "",
    "Images (*.png *.jpg *.jpeg *.bmp)"
)

if not file_path:
    return

self.current_image_path = file_path

pixmap = QPixmap(file_path)
scaled_pixmap = pixmap.scaled(
    700,
    420,
    Qt.AspectRatioMode.KeepAspectRatio,
    Qt.TransformationMode.SmoothTransformation
)

self.image_label.setPixmap(scaled_pixmap)

def run_analysis(self):
    if not self.current_image_path:
        QMessageBox.warning(self, "Помилка", "Спочатку завантажте зображення")
        return

    try:
        result = self.analysis_service.analyze(self.current_image_path)
        self.current_result = result

        self.class_label.setText(f"Основний клас: {result.predicted_class}")
        self.confidence_label.setText(f"Упевненість: {result.confidence}%")
        self.conclusion_text.setPlainText(result.conclusion)

    except Exception as error:
        QMessageBox.critical(self, "Помилка аналізу", str(error))

def save_result(self):
    if not self.current_result:
        QMessageBox.warning(self, "Помилка", "Результат аналізу відсутній")
        return

    patient_code = self.patient_code_input.text().strip()

    if not patient_code:
        QMessageBox.warning(self, "Помилка", "Введіть код пацієнта або дослідження")
        return

    result_id = self.repository.save_result(
        patient_code=patient_code,
        image_path=self.current_image_path,
        predicted_class=self.current_result.predicted_class,
        confidence=self.current_result.confidence,
        conclusion=self.conclusion_text.toPlainText()
    )

    QMessageBox.information(
        self,
        "Збережено",
        f"Результат аналізу збережено. ID запису: {result_id}"
    )

```

```

    )

    def export_report(self):
        if not self.current_result:
            QMessageBox.warning(self, "Помилка", "Немає результату для
експорту")
            return

        output_path, _ = QFileDialog.getSaveFileName(
            self,
            "Зберегти TXT-звіт",
            "diagnostic_report.txt",
            "Text files (*.txt)"
        )

        if not output_path:
            return

        self.exporter.export_txt(
            patient_code=self.patient_code_input.text().strip(),
            image_path=self.current_image_path,
            predicted_class=self.current_result.predicted_class,
            confidence=self.current_result.confidence,
            conclusion=self.conclusion_text.toPlainText(),
            output_path=output_path
        )

        QMessageBox.information(self, "Експорт", "TXT-звіт успішно сформовано")

```

ДОДАТОК Г

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Штепа Дмитро Юрійович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Програмне забезпечення системи класифікації медичних зображень з використанням глибокого навчання» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - ☐ інструменти генеративного ШІ не використовувалися.
 - ☐ інструменти ШІ (вказати назву: **ChatGPT**, Gemini, Claude, DeepL, _____ інші (вказати назву)) були використані для:
 - ☐ перекладу іншомовних джерел;
 - ☐ стилістичного редагування та перевірки граматики;
 - ☐ допомоги у написанні/відлагодженні програмного коду;
 - ☐ інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«__» _____ 2026 р. _____ Дмитро ШТЕПА
(підпис)