

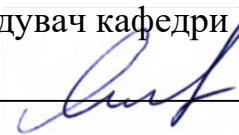
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: Дослідження методів пояснюваності нейронних мереж у медичній
сфері

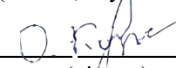
Виконав: студент групи 6141 м



Вербицький О.С.
(підпис)

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,
(посада, науковий ступінь вчене звання)



Гайдаєнко О. В.
(підпис)

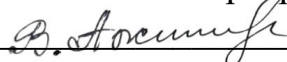
м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
 Навчально-науковий інститут комп'ютерних наук та управління проєктами
 Кафедра інформаційних управляючих систем та технологій
 Спеціальність 122 "Комп'ютерні науки"
 Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми



к.т.н, доц. Ажищев В.Ф.

" ____ " _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту _____ Вербицькому Олександр Сергійовичу _____
 (прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів пояснюваності нейронних мереж у медичній сфері

Керівник роботи _____ Гайдаєнко Оксана Володимирівна _____

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів): Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Дослідження Методів; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Техніко-Економічне Обґрунтування Розробки; Розділ з охорони праці; Розділ з охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, інструкція користувача, програма та методика випробувань програмного забезпечення, текст програми).

5. Перелік презентаційних матеріалів: Титульний слайд, Мета та завдання Кваліфікаційної роботи, Аналіз існуючих рішень, Аналіз вимог, Архітектура програмного забезпечення, Діаграма варіантів використання, Діаграма послідовностей, Діаграма розміщення програмного забезпечення, Діаграма класів, Діаграма потоку даних, Робочий проєкт, Результати розробки, Апробація дослідження, Висновки, Перспективи розвитку проєкту, Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гайдаєнко О. В., доцент кафедри. ІУСТ, к.т.н.	05.09.2024	22.12.2024
2	Гайдаєнко О. В., доцент кафедри. ІУСТ, к.т.н.	23.12.2024	15.06.2025
3	Гайдаєнко О. В., доцент кафедри. ІУСТ, к.т.н.	15.06.2025	25.09.2025
4	Гайдаєнко О. В., доцент кафедри. ІУСТ, к.т.н.	25.09.2025	14.11.2025
5	Гайдаєнко О. В., доцент кафедри. ІУСТ, к.т.н.	25.09.2025	14.11.2025
6	Гайдаєнко О. В., доцент кафедри. ІУСТ, к.т.н.	25.09.2025	14.11.2025

7. Дата видачі завдання 05 вересня 2024 р.**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	виконано
2	Розробка концепції	15.03.2025	виконано
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	виконано
4	Реалізація проєкту та розробка робочої документації	25.09.2025	виконано
5	Обчислення економічної ефективності	14.11.2025	виконано
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	виконано
7	Оформлення пояснювальної записки	14.11.2025	виконано
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	виконано
9	Подання магістерської роботи рецензенту	20.11.2025	виконано
10	Подання магістерської роботи на державний захист	15.12.2025	виконано

Студент

(підпис)

Керівник роботи

(підпис)

Вербицький О.С.

(ПІБ)

Гайдаєнко О.В.

(ПІБ)

АНОТАЦІЯ

Дана курсова робота присвячена аналізу методів пояснюваності нейронних мереж у медичній сфері. У сучасній медицині штучний інтелект та нейронні мережі знаходять дедалі ширше застосування, зокрема у діагностиці, прогнозуванні та персоналізованому лікуванні. Однак складність та "чорний ящик" таких моделей викликають питання довіри та необхідності інтерпретації результатів, що стає особливо важливим у контексті прийняття критичних рішень для життя і здоров'я пацієнтів.

Мета роботи полягає в підвищенні оцінки ефективності існуючих методів пояснюваності нейронних мереж у медичній сфері для підвищення довіри, прозорості та прийнятності рішень ШІ серед медичних фахівців. Окрему увагу приділено викликам, пов'язаним із коректністю даних, уникненням помилок у діагностиці та підвищенням прозорості моделей.

Результати дослідження можуть сприяти подальшому розвитку методологій пояснюваності, забезпечуючи більшу довіру до штучного інтелекту в медицині, що є важливим кроком на шляху до покращення якості медичних послуг.

Кваліфікаційна робота викладена на 80 сторінках машинописного тексту та містить: 18 рисунків, 4 додатків, список використаних джерел з 15 найменувань.

Ключові слова: пояснюваність нейронних мереж; штучний інтелект у медицині; медична діагностика; прозорість моделей ШІ; довіра до штучного інтелекту.

ABSTRACT

This term paper is dedicated to the analysis of explainability methods for neural networks in the medical field. In modern medicine, artificial intelligence and neural networks are increasingly used in diagnostics, prognosis, and personalized treatment. However, the complexity and "black-box" nature of such models raise questions of trust and the necessity for interpretation of results, which becomes critically important when making life-impacting decisions for patients.

The objective of this study is to analyze current approaches to explaining the operation of neural networks and evaluate their effectiveness in the medical context. The paper examines various explainability methods, their advantages, limitations, and potential for integration into medical systems. Special attention is given to challenges related to data accuracy, error avoidance in diagnostics, and enhancing model transparency.

The research findings can contribute to the further development of explainability methodologies, fostering greater trust in artificial intelligence in medicine, which is a vital step toward improving the quality of medical services.

The qualification work is laid out on 80 pages of typewritten text and contains: 80 figures, 4 appendices, a list of used sources from 15 titles.

Key words: explainability of neural networks; artificial intelligence in medicine; medical diagnostics; transparency of ai models; trust in artificial intelligence;

ЗМІСТ

	Стор.
ВСТУП	9
РОЗДІЛ 1. ДОСЛІДЖЕННЯ МЕТОДІВ ПОЯСНЮВАНOSTІ НЕЙРОННИХ МЕРЕЖ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Аналіз предметної області	11
1.2 Аналіз існуючих рішень та їх обмеження, обґрунтування розробки	13
1.3 Аналіз ефективності сучасних рішень у медичній сфері	18
1.4 Формулювання проблеми	20
1.5 Концепція системи	21
1.6 Постановка задачі	23
Висновки до розділу 1	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОЯСНЮВАНOSTІ НЕЙРОННИХ МЕРЕЖ У МЕДИЧНІЙ СФЕРІ	25
2.1 Архітектурне проєктування системи	25
2.1.1 Вибір та обґрунтування архітектурного стилю	25
2.1.2 Компонентна архітектура системи	26
2.1.3 Вибір технологічного стеку	27
2.2 Функціональне проєктування	29
2.2.1 Визначення акторів та варіантів використання (Use Case Diagram)	29
2.2.2 Деталізація сценаріїв взаємодії (Sequence Diagrams або Activity Diagrams)	30
2.3 Проєктування даних	32
2.3.1 Концептуальна модель даних (Class Diagram)	32
2.3.2 Логічна модель бази даних	34
2.3.3 Фізична модель бази даних	36

	Стор.
2.4 Проєктування інтерфейсів та інтеграції	37
2.4.1 Інтерфейси взаємодії з зовнішніми системами	37
2.4.2 Проєктування API системи	39
2.4.3 Концепція інтерфейсу користувача (UI)	41
2.5 Нефункціональні вимоги	44
2.5.1 Проєктні рішення для забезпечення безпеки	44
2.5.2 Проєктні рішення для забезпечення продуктивності та масштабованості	46
Висновки до розділу 2	47
РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОЯСНЮВАНOSTІ НЕЙРОННИХ МЕРЕЖ У МЕДИЧНІЙ СФЕРІ	49
3.1 Середовище та інструменти розробки	49
3.1.1 Огляд технологічного стеку реалізації	49
3.1.2 Система керування версіями	50
3.1.3 Інструменти збірки та керування залежностями	50
3.2 Реалізація компонентів системи	51
3.2.1 Розробка серверної частини (Backend)	51
3.2.2 Розробка клієнтської частини (Frontend)	52
3.2.3 Інтеграція модуля ШІ та пояснюваності	53
3.3 Впровадження заходів безпеки	54
3.3.1 Реалізація автентифікації та авторизації	55
3.3.2 Захист даних та протидія загрозам	56
3.4 Тестування та забезпечення якості	57
3.4.1 Стратегія та види тестування	57
3.4.2 Модульне та інтеграційне тестування	58
3.4.3 Системне та приймальне тестування	58
3.4.4 Тестування безпеки	59

	Стор.
3.5 Розгортання системи	59
3.5.1 Цільове середовище розгортання	60
3.5.2 Вимоги до апаратного та програмного забезпечення сервера	60
3.5.3 Діаграма розгортання (Deployment Diagram)	61
3.5.4 Процес розгортання	62
3.6 Забезпечення експлуатаційної готовності	63
3.6.1 Моніторинг та оповіщення	63
3.6.2 Централізоване логування	64
3.6.3 Резервне копіювання та відновлення	64
Висновки до розділу 3	65
РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ	67
Висновки до розділу 4	70
РОЗДІЛ 5. ОХОРОНА ПРАЦІ	71
5.1 Загальні положення про охорону праці	71
5.2 Заходи щодо забезпечення безпеки на робочому місці розробника програмного забезпечення	71
Висновки до розділу 5	73
РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	74
Висновки до розділу 6	77
ВИСНОВКИ (Загальні)	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	81
ДОДАТКИ	83
Додаток А Технічне завдання на розробку сервісу	83
Додаток Б Інструкція користувача	87
Додаток В Програма та методика випробування програмного забезпечення	92
Додаток Г Фрагмент програмного коду	96

ВСТУП

З розвитком штучного інтелекту цей напрямок став ключовим інструментом для вирішення складних завдань у різних сферах — від автоматизації процесів до прийняття відповідальних рішень. Однак проблема «чорного ящика» штучного інтелекту залишається актуальною: механізми прийняття рішень часто незрозумілі й важко пояснюються існуючими математичними моделями. Як забезпечити прозорість і контроль над такими рішеннями та які нові підходи можуть вирішити цю проблему? Одним із ключових інструментів сучасного штучного інтелекту є архітектури глибоких нейронних мереж, зокрема трансформери. Трансформери використовуються для обробки послідовностей даних і є основою для багатьох успішних моделей, таких як великі мовні моделі, що працюють з текстами, зображеннями та іншими типами даних [1]. Принцип роботи трансформерів полягає у перетворенні вхідної послідовності для прогнозування або класифікації даних [2]. Однак, навіть у таких успішних моделях виникає проблема пояснюваності — не можна чітко пояснити, чому модель прийняла те чи інше рішення. Це обмежує застосування таких технологій у критичних сферах, таких як медицина.

Проблема «чорного ящика» проявляється у багатьох аспектах. Зокрема, навіть найкращі моделі не завжди працюють безпомилково. У процесі навчання нейронні мережі можуть проявляти несподівані властивості, яких не було закладено в навчальну програму. Наприклад, нейронні мережі можуть генерувати розв'язки задач, використовуючи «контекстне навчання», яке дозволяє моделі створювати нові приклади на основі заданих. Однак ці здатності не завжди можна пояснити стандартними математичними методами, що створює враження, що поточні теоретичні підходи не здатні дати чітке розуміння того, як працюють сучасні моделі. Ці невизначеності і «непояснювані» властивості моделювання ускладнюють розуміння того, що саме робить дані «хорошими» або «поганими» і як коректно організувати навчання моделей для отримання надійних результатів. Внаслідок цього виникають помилки, коли моделі вчать

не на тих ознаках, які насправді мають значення, а на випадкових чи неважливих для завдання елементах. Одним із прикладів є ситуація, коли модель, що передбачає кількість людей на станції метро, почала «вивчати» не самих людей, а час доби, що дає їй несправжнє розуміння ситуації.

З розвитком штучного інтелекту виникає потреба у створенні нової теорії навчання, яка б могла пояснити ці несподівані та не завжди зрозумілі властивості. Така теорія дозволить краще контролювати процеси навчання та прийняття рішень, а також застосовувати моделі штучного інтелекту в таких критичних сферах, як медицина, де кожна помилка може мати серйозні наслідки [3]. Наразі основна увага приділяється не стільки розвитку чіткої теоретичної бази, скільки інженерним методам, які дозволяють адаптувати моделі до реальних умов і досягати конкретних цілей. Однак для сталого розвитку цієї технології необхідно перейти до більш глибокого теоретичного розуміння, яке дозволить уникнути таких негативних ефектів, як галюцинації моделі або хибні висновки, що виникають через некоректне навчання [4].

Мета цієї роботи - це підвищення оцінки ефективності існуючих методів пояснюваності нейронних мереж у медичній сфері для підвищення довіри, прозорості та прийнятності рішень ШІ серед медичних фахівців. Основним завданням є дослідження існуючих підходів до покращення прозорості та інтерпретованості результатів, отриманих від нейронних мереж, а також розробка нових методів для їх впровадження у сферу охорони здоров'я. На основі цього дослідження будуть сформульовані рекомендації щодо підвищення ефективності і безпеки використання штучного інтелекту в таких чутливих сферах, де кожне рішення може мати значний вплив на здоров'я людини.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МЕТОДІВ ПОЯСНЮВАНOSTІ НЕЙРОННИХ МЕРЕЖ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Розвиток штучного інтелекту (ШІ) та нейронних мереж справив значний вплив на медичну сферу. Нейронні мережі дозволяють ефективно обробляти великі обсяги медичних даних, автоматизувати діагностику, прогнозувати розвиток захворювань і навіть створювати персоналізовані плани лікування. Висока точність таких моделей зробила їх невіддільною частиною сучасної медицини.

Однак, незважаючи на переваги, багато нейронних мереж сприймаються як "чорна скринька", оскільки важко пояснити, як вони приймають рішення. У медичній сфері це викликає серйозне занепокоєння, адже лікарі, пацієнти та регуляторні органи потребують чітких пояснень, аби довіряти результатам роботи ШІ. Пояснюваність нейронних мереж є критично важливою для підвищення довіри, інтеграції таких технологій у клінічну практику та забезпечення безпечності рішень.

З огляду на це, аналіз методів підвищення пояснюваності є не лише актуальним, але й стратегічно важливим завданням для майбутнього медицини.

Застосування штучного інтелекту для діагностики

Штучний інтелект, зокрема нейронні мережі, активно використовуються для автоматизації процесів діагностики в медицині. Вони здатні обробляти великі обсяги медичних даних, таких як зображення медичних сканів (МРТ, КТ, рентген), електронні медичні картки, генетичні дані та інші. Використання глибоких нейронних мереж, наприклад, для розпізнавання патологій на зображеннях, дозволяє значно підвищити точність діагнозів, навіть у складних випадках, де традиційний метод може бути обмежений через людський фактор.

Моделі ШІ можуть ідентифікувати хвороби на ранніх стадіях, коли симптоми ще не проявляються, що дозволяє здійснювати своєчасне лікування та покращувати прогноз для пацієнтів. Нейронні мережі допомагають лікарям

швидко визначити діагноз, знизивши час на обробку даних і підвищивши ефективність лікування [5]. Тому постає актуальним питання розробки описових моделей.

Об'єктом дослідження будуть процеси використання нейронних мереж у медичній діагностиці та прийнятті рішень.

Предметом - методи пояснюваності нейронних мереж, застосовані до медичних даних і рішень.

Мета роботи - підвищення оцінки ефективності існуючих методів пояснюваності нейронних мереж у медичній сфері для підвищення довіри, прозорості та прийнятності рішень ШІ серед медичних фахівців.

Пояснюваність як ключова вимога та виклики впровадження ШІ

Попри високу ефективність нейронних мереж, їх широке застосування у медицині обмежується проблемою «чорної скриньки» — непрозорістю внутрішніх процесів прийняття рішень. Для клінічної практики критично важливо, щоб лікар міг верифікувати висновки моделі та розуміти причини конкретного прогнозу. Відсутність інтерпретованості унеможливорює повноцінний контроль помилок та створює бар'єр для довіри, оскільки медичні працівники несуть відповідальність за життя пацієнта і не можуть покладатися на непроявлені алгоритми.

Формулювання проблеми

Основна наукова проблема полягає у необхідності підвищення прозорості та пояснюваності рішень, прийнятих нейронними мережами. Використання штучного інтелекту, зокрема нейронних мереж, у медицині відкриває нові можливості для підвищення ефективності діагностики, лікування та прийняття медичних рішень. Метою є трансформація «чорної скриньки» у зрозумілий інструмент підтримки прийняття рішень, що дозволить лікарям не лише отримувати результат, а й розуміти його логіку. Забезпечення пояснюваності є необхідною умовою для відповідності етичним стандартам, підвищення безпеки діагностики та повноцінної інтеграції штучного інтелекту в систему охорони здоров'я.

1.2 Аналіз існуючих рішень та їх обмеження, обґрунтування розробки Огляд існуючих методів пояснюваності

Local Interpretable Model-agnostic Explanations (LIME)

LIME — це метод інтерпретації, що дозволяє пояснити роботу складних моделей машинного навчання («чорних скриньок») шляхом їх локальної апроксимації простішими, інтерпретованими моделями (наприклад, лінійними). Принцип роботи полягає у генерації пертурбацій (варіацій) вхідних даних навколо конкретного прикладу та аналізі того, як зміни ознак впливають на прогноз основної моделі.

У контексті медичної діагностики LIME забезпечує прозорість на рівні індивідуального пацієнта. Метод дозволяє виокремити конкретні клінічні показники або симптоми (наприклад, результати аналізів чи анамнез Рисунок 1), які стали вирішальними для формування прогнозу, що дає змогу лікарю верифікувати обґрунтованість рішення ШІ.

Переваги: Універсальність (модельна агностичність) та здатність надавати зрозумілі пояснення для окремих випадків, що знижує рівень недовіри до автоматизованих систем. Обмеження: нестабільність пояснень, залежність від параметрів генерації локального простору та зниження ефективності при роботі з багатовимірними даними [6], що ускладнює його використання для глобальної оцінки моделі.

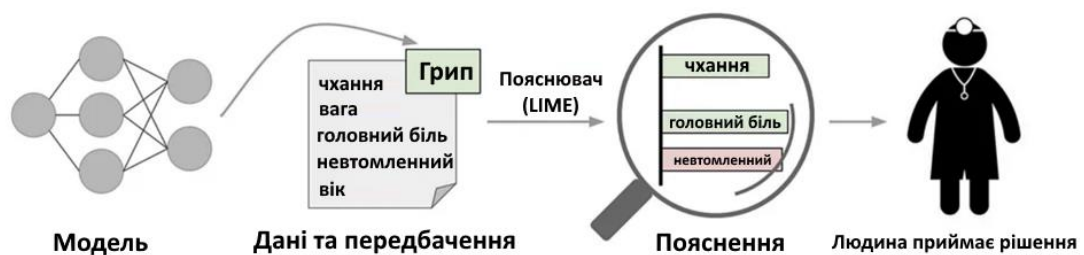


Рисунок 1. Пояснення індивідуальних передбачень.

Gradient-weighted Class Activation Mapping (Grad-CAM)

Grad-CAM — це метод візуалізації, призначений для інтерпретації рішень згорткових нейронних мереж (CNN). Алгоритм використовує градієнти цільового класу в останньому згортковому шарі для генерації карт активації

(теплових карт), які підсвічують зони вхідного зображення, що мали вирішальний вплив на прогноз моделі.

У медичній діагностиці (аналіз рентгенограм, МРТ, КТ, зображень сітківки) Grad-CAM дозволяє лікарям візуально верифікувати роботу ШІ, визначаючи, чи базується діагноз на клінічно значущих патологіях.

Переваги: наочність пояснень, універсальність для будь-яких CNN-архітектур та підвищення довіри до системи. Обмеження: неможливість застосування до моделей без згорткових шарів (наприклад, класичних MLP) та обмежена роздільна здатність теплових карт, що може знижувати точність локалізації дрібних деталей. Використання методу Gradient-weighted Class Activation Mapping (Grad-CAM) для візуалізації активації моделі при аналізі зображень сітківки ока Рисунок 2.

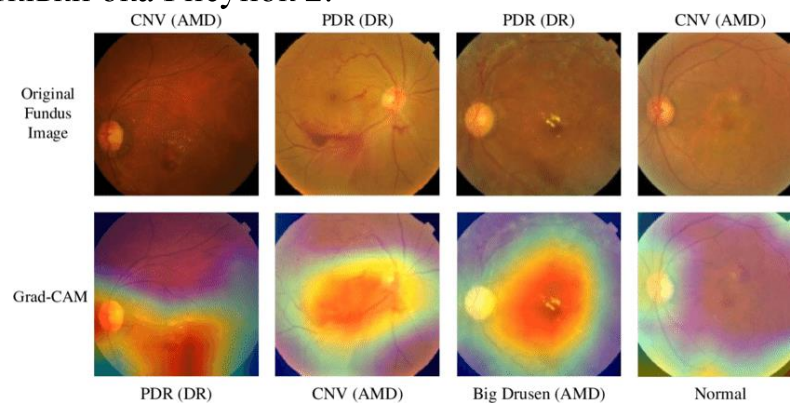


Рисунок 2. Метод Gradient-weighted Class Activation Mapping

SHapley Additive exPlanations (SHAP)

SHAP (SHapley Additive exPlanations) — це універсальний метод інтерпретації моделей машинного навчання, заснований на кооперативній теорії ігор. Він використовує концепцію значень Шеплі для математично обґрунтованого розподілу «внеску» кожної вхідної ознаки у фінальний результат. Ключовою особливістю методу є адитивність: прогноз моделі представляється як сума впливів усіх ознак відносно базового значення, що дозволяє точно оцінити важливість кожного фактора.

Переваги: суворе теоретичне підґрунтя, здатність працювати з будь-якими типами моделей (model-agnostic) та висока точність пояснень. Обмеження:

висока обчислювальна складність для великих наборів даних та чутливість до корельованих ознак і шуму, що є типовим для медичних даних.

Зображення іонів (зліва) та карти SHAP (праворуч) для трьох ознак (тобто значень m/z), що мають найбільший вплив на процес прийняття рішень класифікатора Рисунок 3.

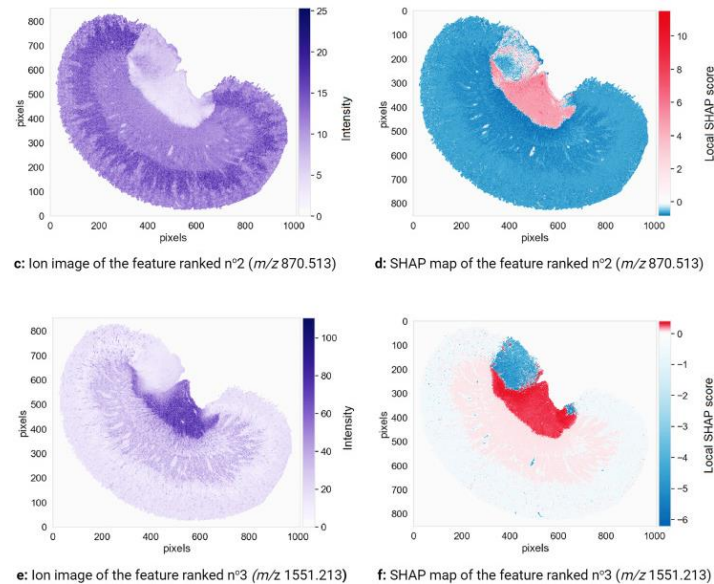


Рисунок 3 - молекулярні маркери для внутрішньої медули нирки.

Integrated Gradients

Integrated Gradients — це метод пояснюваності, який дозволяє оцінити внесок кожної вхідної ознаки у прогноз моделі глибокого навчання, ґрунтуючись на аксіоматичних принципах. Метод досягає цієї мети шляхом обчислення інтеграла градієнтів вихідної функції моделі $F(x)$ відносно вхідних даних x вздовж прямого шляху, що з'єднує базову точку x' (наприклад, нульове зображення) та фактичні вхідні дані x . Математично внесок ознаки i (інтегрований градієнт) визначається як:

$$\text{Integrated Gradient}(x_i) = (x_i - x'_i) \int_{\alpha=0}^1 \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha \quad (1.1)$$

Де x — реальні вхідні дані, x' — базова точка, F — модель, а α — коефіцієнт масштабування.

Використання та характеристики Цей підхід забезпечує більш точний та рівномірний розподіл важливості ознак порівняно з простими градієнтами. Він

часто застосовується у біомедичних задачах для виявлення важливих біомаркерів або аналізу значущих областей на медичних зображеннях.

Переваги: є аксіоматично обґрунтованим і не вимагає доступу до внутрішньої структури моделі, що підвищує його універсальність. Обмеження: висока обчислювальна складність, оскільки вимагає інтегрування градієнтів по всьому шляху, та чутливість до вибору базової точки, що може впливати на фінальний результат пояснення.

DeepLIFT (Deep Learning Important FeaTures)

DeepLIFT — це метод інтерпретації, який визначає внесок кожної вхідної ознаки у вихідний результат шляхом порівняння активацій нейронів для поточного входу з активаціями обраного «базового» (нейтрального) стану. Алгоритм базується на розрахунку різниці активацій Δ та її пропорційному розподілі крізь шари нейронної мережі, що дозволяє ефективно відстежувати шлях прийняття рішення навіть у глибоких архітектурах.

Внесок кожної вхідної ознаки визначається через пропорційний розподіл впливу активацій, які поширюються крізь шари нейронної мережі.

$$C_{\Delta x}(x) = \frac{\Delta t(x)}{\Delta x} \Delta x \quad (1.2)$$

Де $\Delta t(x)$ — зміна активації в порівнянні з базовою, Δx — різниця між поточними та базовими вхідними значеннями.

У медичному контексті DeepLIFT використовується для: локалізації патологічних зон на медичних зображеннях (сегментація пухлин); оцінки ваги клінічних змінних (біомаркери, анамнез) у прогностичних моделях [7].

Переваги: дозволяє уникнути проблем насичення градієнтів, характерних для старих методів, та є універсальним для різних архітектур (CNN, RNN).

Недоліки: результати критично залежать від коректності вибору «базового стану» (Reference), а обчислювальна складність зростає пропорційно глибині мережі.

Layer-wise Relevance Propagation (LRP)

LRP — це метод інтерпретації, що базується на механізмі зворотного розповсюдження прогнозу від вихідного шару до вхідних даних. Його фундаментальним принципом є закон збереження релевантності: загальна сума «важливості» на кожному шарі мережі залишається незмінною, перерозподіляючись між нейронами пропорційно силі їхніх зв'язків.

У медичній візуалізації LRP дозволяє будувати високоточні карти ознак (pixel-wise explanations) для рентгенограм, МРТ та КТ. Це дає змогу лікарю бачити конкретні анатомічні зони, які модель ідентифікувала як патологічні (наприклад, межі новоутворень), забезпечуючи прозорість діагностики.

Аналіз структури. Релевантність кожної вхідної ознаки оцінюється як міра її впливу на остаточне рішення.

$$R_j = \sum_i \frac{z_{ij}}{\sum_k z_{ik}} R_i \quad (1.3)$$

Де z_{ij} — зв'язок між нейронами j і i , R_i — релевантність нейрона на поточному шарі, R_j — релевантність на попередньому.

Переваги та обмеження

Переваги: забезпечує глибоку декомпозицію рішення моделі до рівня окремих вхідних ознак; метод є обчислювально ефективним та застосовним до різних архітектур (CNN, RNN).

Недоліки: результати можуть бути нестабільними залежно від обраних правил розповсюдження (propagation rules) та функцій активації, що потребує ретельного налаштування.

Counterfactual Explanations (Контрфактичні пояснення)

Counterfactual Explanations — це метод інтерпретації, що надає пояснення у форматі «що-якщо», визначаючи мінімальні необхідні зміни вхідних даних для зміни рішення моделі на бажане (наприклад, перемикання прогнозу з «хворий» на «здоровий»). Замість аналізу причин поточного висновку, метод моделює гіпотетичні сценарії.

Обчислювальна складність та часові затримки. Методи на кшталт SHAP мають експоненційну складність, а LIME вимагає генерації тисяч зразків. В умовах ургентної медицини, де рішення приймаються миттєво, такі затримки є неприпустимими, а вимоги до апаратного забезпечення — економічно невігідними.

Проблема локальності. Більшість методів дають пояснення лише для окремого пацієнта, не розкриваючи глобальну логіку моделі. Це ускладнює виявлення системних упереджень (bias) та сертифікацію медичного ПЗ регуляторами.

Залежність від типу даних. Якщо для структурованих даних інтерпретація є доволі точною, то для неструктурованих (зображення, текст) пояснення часто бувають нестабільними. Теплові карти можуть вказувати на правильну область, але не пояснювати причину висновку, що обмежує їх діагностичну цінність.

«Ілюзія розуміння». Оскільки методи пояснення є лише апроксимаціями (наближеннями) роботи надскладних нейромереж, існує ризик надання переконливих, але хибних обґрунтувань. Це може ввести лікаря в оману, створивши хибне відчуття надійності системи.

Хоча існуючі методи XAI (SHAP, LIME, Grad-CAM) забезпечують базовий рівень інтерпретації, їх технічні обмеження (ресурсоемність, локальність, нестабільність на зображеннях) стримують створення повноцінних прозорих систем підтримки прийняття лікарських рішень.

Недостатня прозорість великих моделей. Сучасна медицина все частіше використовує глибокі нейронні мережі ("чорні скриньки"), які мають мільйони параметрів. Існуючі методи пояснюваності часто є лише апроксимаціями (наближеннями) роботи цих моделей. Це створює парадокс: прості моделі (лінійна регресія, дерева рішень) є прозорими, але менш точними, тоді як високоточні глибокі мережі залишаються непрозорими навіть при використанні XAI. Ризик полягає в тому, що метод пояснення може створити ілюзію розуміння, надаючи переконливе, але хибне обґрунтування (наприклад,

виділяючи правильну область на рентгені, але базуючись на сторонніх артефактах, таких як мітки сканера), що може призвести до лікарських помилок.

1.4 Формулювання проблеми

Сучасні методи глибокого навчання демонструють високу точність у медичній діагностиці, проте їх впровадження у клінічну практику стримується проблемою «чорної скриньки» (black box) [8]. Критичний аналіз існуючих підходів до пояснюваності штучного інтелекту (Explainable AI, XAI) дозволяє виділити ряд фундаментальних недоліків, які потребують вирішення.

Недостатня прозорість та складність розуміння поведінки великих моделей

Ключовою проблемою є непрозорість архітектур глибоких нейронних мереж. На відміну від лінійних моделей, де внесок кожного параметра є очевидним, глибокі мережі містять мільйони параметрів та нелінійних зв'язків. У задачах аналізу медичних зображень (Рентген, МРТ) або електронних медичних карт це призводить до неможливості чітко визначити, які саме патерни (пікселі, текстові ознаки) стали основою для прийняття рішення. Лікар отримує результат (діагноз), але не має доступу до логічного ланцюжка висновків, що є неприпустимим у доказовій медицині.

Технічні обмеження і складність інтерпретації

Популярні методи апостеріорного пояснення, такі як SHAP (Shapley Additive Explanations) або LIME (Local Interpretable Model-agnostic Explanations), базуються на локальній апроксимації моделі. Це створює низку технічних обмежень.

1. Ресурсоємність. Для великих наборів вхідних даних (зокрема, 3D-зображень) обчислення значень важливості ознак вимагає значних потужностей, що ускладнює роботу в режимі реального часу.

2. Нестабільність. Оскільки ці методи є апроксимаціями, вони можуть давати різні пояснення для схожих вхідних даних, що підриває довіру до системи.

3. Залежність від типу даних. Більшість методів оптимізовані для табличних даних і погано масштабуються на неструктуровані мультимодальні дані, характерні для медицини.

Відсутність загальної теорії навчання та пояснюваності

На сьогодні у сфері машинного навчання відсутня єдина теоретична база, яка б математично формалізувала причини прийняття рішень моделлю. Існуючі методи є здебільшого емпіричними та евристичними, не маючи під собою строгої аксіоматики, яка б гарантувала істинність пояснення.

Штучний інтелект часто виступає як відкрита система, здатна виявляти приховані, іноді контрінтуїтивні кореляції, які не передбачені навчальною вибіркою. Без розробки нової теорії навчання, яка пояснює не лише як (механізм), але й чому (причинність) модель діє певним чином, неможливо забезпечити надійну валідацію результатів у критичних медичних ситуаціях.

1.5 Концепція системи

Опис очікуваних результатів

Підвищення ефективності існуючих методів пояснення в медичних застосуваннях, що дозволить більш точно інтерпретувати результати нейронних мереж та підвищити їх прийнятність серед медичних працівників.

Систематизація і групування існуючих методів для подальшого вдосконалення та адаптації їх до різних медичних контекстів.

Визначення критеріїв для оцінки пояснюваності моделей у медичних задачах і рекомендації щодо покращення пояснюваності для майбутніх розробок. Аналіз методів пояснюваності нейронних мереж у медичних задачах і визначення напрямків їх удосконалення.

Для реалізації концепції системи, спрямованої на підвищення ефективності та прийнятності методів пояснюваності в медицині, необхідно визначити оптимальні підходи, які враховують специфіку медичних даних і вимоги клінічної практики [9].

На основі аналізу сучасних методів XAI (таких як LIME, Grad-CAM, SHAP, Integrated Gradients, DeepLIFT та LRP) було проведено їх порівняльну оцінку за

ключовими критеріями: **тип даних** (наявність підтримки зображень, text, табличних даних); **точність інтерпретації** (здатність коректно виділяти ключові ознаки); **складність реалізації** (технічні витрати на інтеграцію);

- **прозорість для лікарів** (інтуїтивність візуалізацій); **інтеграція з медичними системами** (сумісність з dicom, ehr).

Результати порівняння представлено в **Таблиці 1**.

Таблиця 1. Порівняльний аналіз методів пояснюваності для медичних задач

Метод	Тип даних	Точність інтерпретації	Складність реалізації	Прозорість для лікарів	Інтеграція з медичними системами	Ваговий коефіцієнт (1-5)
LIME	Текст, таблиці	Середня	Низька	Висока	Обмежена	4.5
Grad-CAM	Зображення	Висока	Середня	Висока	Висока (DICOM/PACS)	5.0
SHAP	Універсальний	Висока	Висока	Середня	Середня	3.8
Integrated Gradients	Зображення, текст	Висока	Висока	Низька	Обмежена	3.5
DeepLIFT	Зображення	Середня	Висока	Низька	Обмежена	3.0
LRP	Зображення, сигнали	Висока	Дуже висока	Низька	Обмежена	2.8

Пояснення до таблиці. Ваговий коефіцієнт розраховано на основі критеріїв, актуальних для медичної сфери. **5.0 (Grad-CAM)** - оптимальний для візуалізації аномалій на зображеннях (наприклад, МРТ, КТ), інтуїтивно зрозумілий лікарям, інтегрується з DICOM. **4.5 (LIME)** - ефективний для інтерпретації табличних даних (лабораторні аналізи), з можливістю виділення ключових ознак. **SHAP/Integrated Gradients** - універсальні, але потребують глибоких технічних знань для інтерпретації. **DeepLIFT/LRP** - складні у налаштуванні, орієнтовані на дослідницькі завдання.

Обґрунтування вибору LIME та Grad-CAM

Grad-cam має найвищий коефіцієнт через: здатність візуалізувати області інтересу на зображеннях у реальному часі; сумісність із медичними стандартами (накладання heatmap на dicom-знімки);

Lime обрано для: інтерпретації незображених даних (наприклад, прогнозування ризику захворювань на основі аналізів крові); генерації локальних пояснень, зрозумілих без глибоких знань ai.

Напрями розвитку методів (відповідно до концепції системи)

Для **Grad-CAM**: автоматизація аналізу 3d-зображень (кт-сканування); інтеграція з ehr для анотації результатів у звітах.

Для **LIME**: покращення стабільності пояснень на малих вибірках; розробка шаблонів інтерпретації для різних типів діагностики.

Таблиця 1 демонструє, що **Grad-CAM** та **LIME** є оптимальними методами для медичних задач серед сучасних XAI-підходів. Їх переваги (простота, інтуїтивність, спеціалізація) відповідають цілям системи, сформульованим у Розділі 1. Інші методи (SHAP, LRP) можуть використовуватися допоміжно в дослідницьких сценаріях, але не задовольняють вимогам клінічної практики.

1.6 Постановка задачі

Для досягнення поставленої мети, належить виконати такий ряд завдань:

1. Проведення порівняння існуючих методів пояснюваності для різних медичних контекстів, визначення їх обмежень та можливостей для адаптації й покращення.
2. Розробка рекомендацій щодо оптимізації методів пояснюваності в контексті медичних рішень.
3. Визначення ключових принципів та методологічних підходів, які можуть стати основою універсальної теорії пояснюваності нейронних мереж, здатної ефективно працювати в різних контекстах, включаючи медицину, з урахуванням специфіки медичних задач.

Висновки до розділу 1

У першому розділі було проведено комплексний аналіз предметної області застосування нейронних мереж у медицині, який підтвердив критичну

важливість забезпечення пояснюваності моделей штучного інтелекту для їх надійного та етичного впровадження у клінічну практику. Було наголошено, що проблема "чорної скриньки" є суттєвим бар'єром для довіри та широкого використання ШІ-систем у діагностиці та прийнятті рішень.

Детально проаналізовано сучасні методи пояснюваності, таких як LIME, Grad-CAM, SHAP, Integrated Gradients, DeepLIFT та LRP. Визначено їх специфіку застосування у медичній сфері для аналізу різних типів даних.

Аналіз ефективності показав, що хоча існуючі методи надають цінні інструменти для інтерпретації рішень нейронних мереж, вони мають значні обмеження: складність реалізації та інтерпретації результатів, часто локальний характер пояснень, залежність від типу даних та архітектури моделі, а також недостатня прозорість для дуже складних моделей. Ці недоліки обмежують практичну цінність та надійність поточних ХАІ-рішень у відповідальних медичних задачах.

На основі виявлених проблем та обмежень було сформульовано наукову проблему дослідження, яка полягає у необхідності систематизації, порівняльного аналізу та вдосконалення методів пояснюваності для специфічних потреб медичної галузі. Визначено концепцію роботи та окреслено конкретні завдання, що включають порівняння ефективності існуючих методів на медичних даних, розробку рекомендацій для їхньої оптимізації та визначення ключових принципів для побудови більш надійних та зрозумілих систем пояснюваного ШІ в медицині. Результати цього розділу створюють теоретичне та методологічне підґрунтя для проєктування та розробки інформаційної системи, що буде описано у наступних розділах роботи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОЯСНЮВАНOSTІ НЕЙРОННИХ МЕРЕЖ У МЕДИЧНІЙ СФЕРІ

2.1 Архітектурне проєктування системи

2.1.1 Вибір та обґрунтування архітектурного стилю

Для проєктованої системи пояснюваності нейронних мереж у медицині обрано **мікросервісну архітектуру**, що обумовлено наступними вимогами.

1. **Масштабованість.** Незалежне масштабування окремих компонентів (наприклад, модулів SHAP, Grad-CAM) при збільшенні навантаження (обробка великих медичних даних, паралельні запити). Гнучкість у розподілі ресурсів для задач із високими обчислювальними витратами (наприклад, генерація пояснень для 3D-зображень) [10].

2. **Інтеграція зовнішніх систем.** Підтримка стандартних протоколів (DICOM, HL7/FHIR) для взаємодії з PACS/EHR через окремі мікросервіси, що мінімізує ризики сумісності. Модульність архітектури дозволяє додавати нові інтерфейси без змін у ядрі системи (наприклад, підключення додаткових джерел даних).

3. **Надійність та безпека.** Ізоляція компонентів знижує ризик поширення збоїв. Наприклад, збій у модулі LIME не впливає на роботу класифікації. Централізовані механізми автентифікації та аудиту для всіх служб (OAuth2, JWT) [11].

4. **Розгортання та підтримка.** Використання контейнеризації (Docker, Kubernetes) забезпечує швидке розгортання в хмарних або локальних середовищах, що критично для медичних установ із різною IT-інфраструктурою. Незалежність мікросервісів дозволяє оновлювати окремі модулі без зупинки всієї системи (наприклад, впровадження нових методів пояснюваності).

Мікросервісна архітектура оптимально відповідає викликам, визначеним у Розділі 1 (прозорість, інтеграція, адаптивність), забезпечуючи основу для впровадження пояснюваного ШІ у клінічну практику.

2.1.2 Компонентна архітектура системи

Архітектура системи базується на модульному підході рисунок 4.

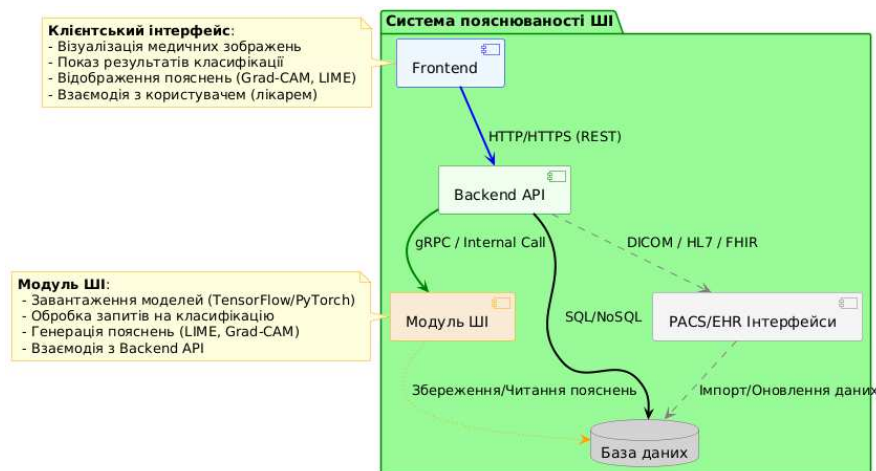


Рисунок 4. Діаграма компонентів системи

Опис компонентів

Frontend. Забезпечує інтерфейс для лікарів (завантаження зображень, перегляд результатів класифікації, візуалізація пояснень). Реалізований на React з використанням бібліотек для обробки медичних даних (DICOM.js).

Backend API. Обробляє запити від клієнта, маршрутизує їх до модулів ШІ та зовнішніх систем. Використовує RESTful API за допомогою бібліотеки ZIO, забезпечує автентифікацію (OAuth2) та кешування.

Модуль ШІ. Включає Docker-контейнери з моделями TensorFlow/PyTorch. Генерує пояснення через інтегровані бібліотеки (SHAP, Captum).

База даних. PostgreSQL з розширенням для геопросторових даних (медичні 3D-зображення). Зберігає анонімізовані дані пацієнтів, результати класифікації та метадані пояснень.

PACS/EHR Інтерфейси. Адаптери для DICOM (pynetdicom) та FHIR (NAPI FHIR). Забезпечують синхронізацію з медичними архівами без змін у вихідних даних.

Блок-схема високого рівня (рисунок 5) відображає послідовність взаємодії компонентів.



Рисунок 5. Блок-схема взаємодії компонентів

Архітектура забезпечує відокремлення функціональних модулів, що відповідає вимогам масштабованості та безпеки з Розділу 1.

2.1.3 Вибір технологічного стеку

Технологічний стек обрано з урахуванням вимог до масштабованості, інтеграції з медичними системами та забезпечення пояснюваності.

Основні компоненти та обґрунтування

Frontend. React з бібліотекою **DICOM.js**. Гнучкий інтерфейс для візуалізації медичних зображень (накладання пояснень Grad-CAM, інтерактивні елементи). Підтримка PWA для офлайн-роботи в умовах обмеженого інтернет-зв'язку.

Backend. Scala 3 (ZIO-HTTP). Висока продуктивність і масштабованість завдяки неблокуючій асинхронній обробці (ZIO). Автоматична генерація OpenAPI (через Tapir) для інтеграції з PACS/EHR. Підтримка JWT/OAuth2 для аутентифікації та керування доступом медичного персоналу; висока типобезпека та сумісність з екосистемою Java.

Модуль ШІ. PyTorch та TensorFlow Serving. Інтеграція з бібліотеками пояснюваності (**Captum** для PyTorch, **SHAP** для TensorFlow). Використання Docker-контейнерів для ізоляції моделей (наприклад, окремий контейнер для кожного методу пояснення).

ONNX Runtime. Оптимізація інтерфейсу для роботи з різними архітектурами моделей.

База даних. PostgreSQL з розширенням **PostGIS**: Ефективне зберігання 3D-зображень та геопросторових даних. ACID-транзакції для цілісності медичних записів. **Redis**: Кешування результатів класифікації для швидкого доступу.

Інтеграція з PACS/HER. pynetdicom (DICOM) та **HAPI FHIR**. Підтримка стандартних протоколів для імпорту/експорту даних без модифікації джерел.

Інструменти розгортання. Docker/Kubernetes. Оркестрація мікросервісів із гарантією відмовостійкості. Ізоляція обчислювальних задач (наприклад, окремий кластер для LIME/SHAP).

Безпека. Vault (Hashicorp): централізоване зберігання секретів (арі-ключі, доступ до бд) [12].

Prometheus/Grafana: моніторинг аномалій у роботі моделей.

Технологічний стек наведено у таблиці 2.

Таблиця 2. Зведення технологічного стеку

Компонент	Технології	Обґрунтування
Frontend	React, DICOM.js	Гнучкість для роботи з DICOM-зображеннями
Backend	Scala 3 (ZIO-HTTP), Redis	Асинхронна обробка запитів до ШІ-модулів
ШІ-модуль	PyTorch, Captum, TensorFlow Serving	Підтримка різних методів пояснюваності та контейнеризації
База даних	PostgreSQL, PostGIS	Надійне зберігання структурованих медичних даних
Інтеграція	pynetdicom, HAPI FHIR	Сумісність із медичними стандартами (DICOM, HL7)
Безпека	Vault, OAuth2	Захист конфіденційних даних пацієнтів

Вибрані технології забезпечують відповідність вимогам щодо прозорості (інтеграція XAI-бібліотек), масштабованості (мікросервіси + Kubernetes) та безпеки (Vault, OAuth2).

2.2 Функціональне проєктування

2.2.1 Визначення акторів та варіантів використання

Діаграма варіантів використання системи наведена на рисунку 6.

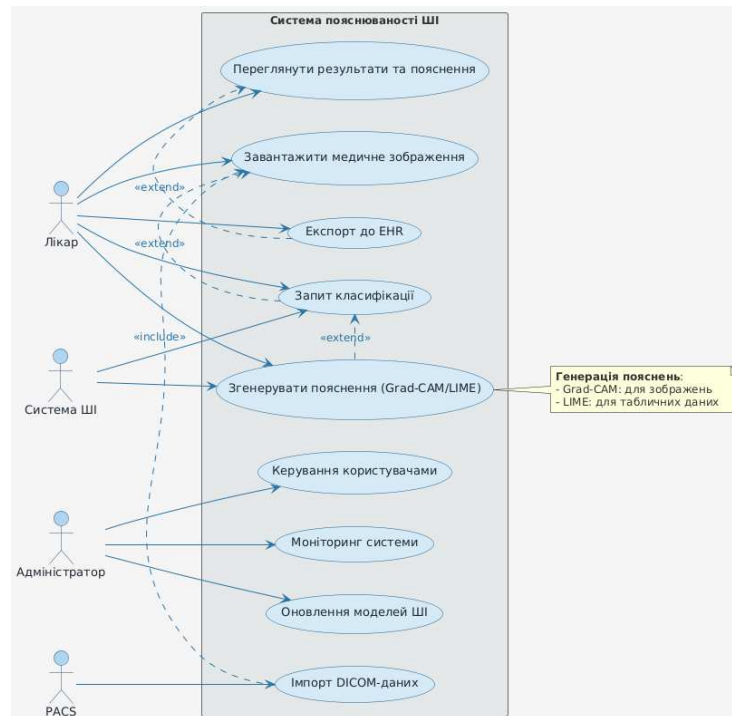


Рисунок 6. Діаграма варіантів використання

Опис акторів

Лікар. Основне джерело взаємодії з системою. Виконує: завантаження зображень, ініціює класифікацію, запитує пояснення, експортує результати до EHR.

Адміністратор. Налаштовує доступ користувачів, контролює роботу системи, оновлює AI-моделі.

Система ШІ. Автономний актор для автоматизованих задач (наприклад, перевірка якості даних, перетренування моделей).

PACS. Зовнішня система для імпорту DICOM-досліджень.

Основні варіанти використання зазначених у таблиці 3.

Таблиця 3. Ключові варіанти використання

Назва	Актор	Опис
Завантажити зображення	Лікар	Завантаження DICOM/NIfTI-файлів або вибір з PACS.
Запит класифікації	Лікар + ІІІ	Запуск моделі для діагностики (наприклад, виявлення пухлини).
Генерація пояснення	Лікар + ІІІ	Вибір методу (Grad-CAM/LIME) для візуалізації важливих ознак.
Експорт до EHR	Лікар	Інтеграція результатів у форматі HL7/FHIR з анотаціями пояснень.
Оновлення моделей ІІІ	Адміністратор	Деплой нових версій моделей з контролем якості (А/В-тестування).

Сценарії розширення. Помилка завантаження зображення. Система ІІІ автоматично перевіряє формат і розмір файлу. У разі невідповідності — повідомлення про помилку.

Невизначеність пояснення. Якщо LIME/Grad-CAM дають суперечливі результати, система пропонує альтернативні методи (наприклад, SHAP).

2.2.2 Деталізація сценаріїв взаємодії

Для ключових варіантів використання наведено діаграми послідовності.

Сценарій 1. Генерація пояснення (Grad-CAM)

Діаграма послідовності для запиту пояснення методом Grad-CAM наведена на рисунку 7.

Опис кроків

1. Лікар завантажує DICOM-зображення через інтерфейс.
2. Backend передає запит до ІІІ-модуля через gRPC (для мінімізації затримок).
3. Модуль ІІІ виконує класифікацію та генерує heatmap-пояснення.

4. Результати зберігаються у БД для подальшого аудиту.

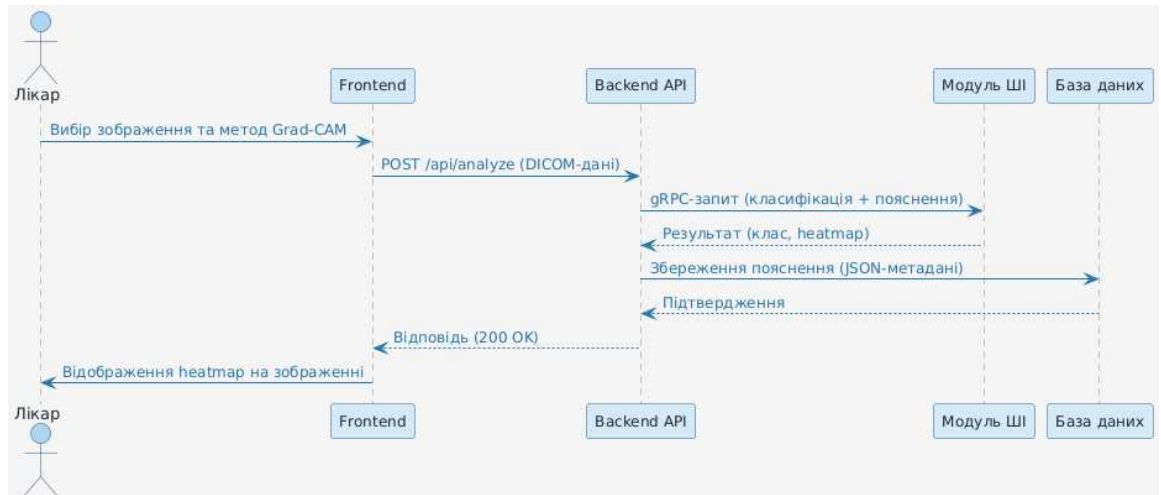


Рисунок 7. Діаграма послідовності генерації пояснення Grad-CAM

Сценарій 2. Обробка помилки генерації пояснення

Діаграма активності для обробки помилки наведена на рисунку 8.

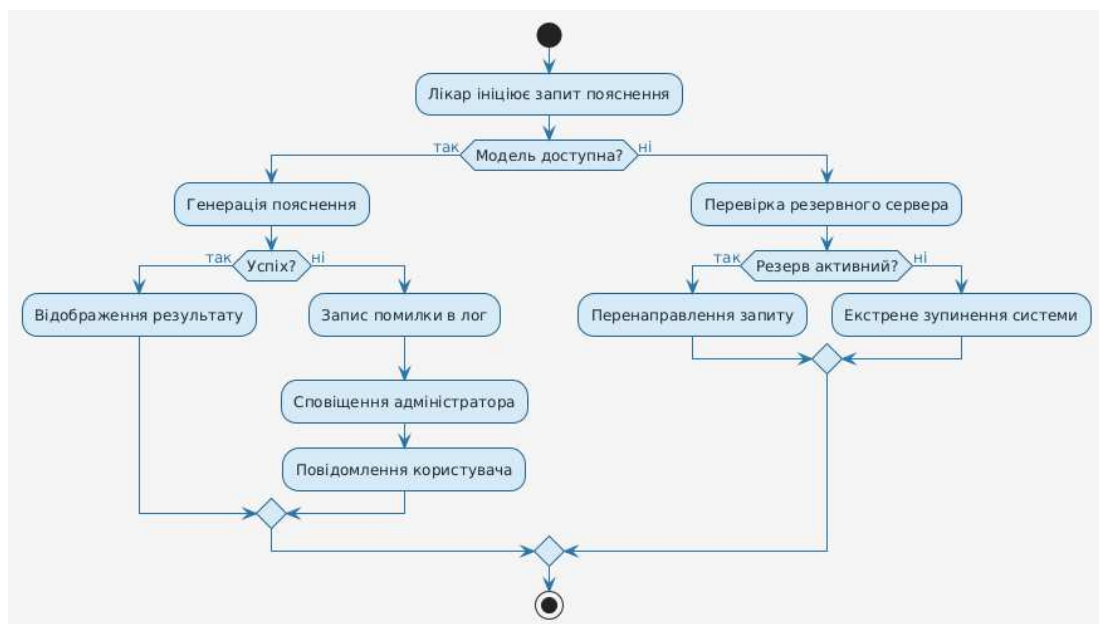


Рисунок 8. Діаграма активності обробки помилок

Ключові механізми:

- Автоматичний перехід на резервний ШІ-сервер при збоях (Kubernetes Health Checks).
- Логування помилок у централізований Elasticsearch для аналізу.

Сценарій 3. Експорт результатів до EHR

Діаграма послідовності експорту результатів до EHR наведена на рисунку 9.

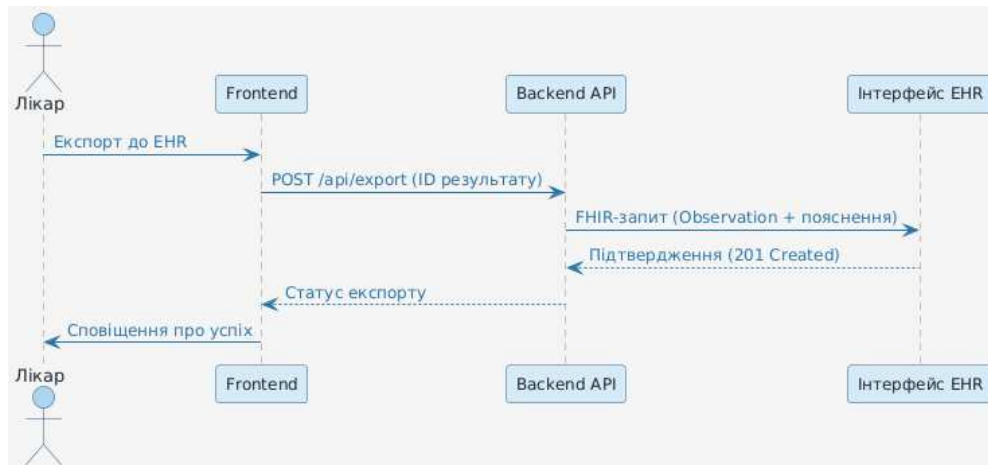


Рисунок 9. Діаграма послідовності експорту до HER

Деталі

- Використання FHIR-ресурсу Observation для передачі пояснень у структурованому вигляді.
- Додаткові метадані включають тип пояснення (Grad-CAM/LIME) та впевненість моделі.

Діаграми узгоджені з архітектурою з 2.1.2 та технологічним стеком з 2.1.3. Враховані вимоги до надійності (сценарії відмов) та інтеграції з медичними стандартами.

2.3 Проєктування даних

2.3.1 Концептуальна модель даних

Концептуальна модель даних системи відображена на рисунку 10.

Опис сутностей

MedicalImage. Зберігає метадані медичних зображень (формат, роздільна здатність). Метод `getMetadata()` повертає технічні параметри для інтеграції з PACS.

PatientInfo. Анонімізовані дані пацієнта (без ПІБ, адреси). Зв'язок з MedicalImage через UUID для дотримання GDPR.

AIModel. Метадані AI-моделей (версія, фреймворк). Метод `predict()` ініціює класифікацію та повертає результат.

ClassificationResult. Результати діагностики (діагноз, впевненість моделі). Метод `logResult()` записує дані для подальшого аудиту.

Explanation (та підкласи **Grad-CAM**, **LIME**). Абстрактний клас для методів пояснення. Градієнтні карти (Grad-CAM) та ваги ознак (LIME) зберігаються у спеціалізованих форматах.

User. Користувачі системи з ролями (лікар, адміністратор). Метод `authenticate()` інтегрований з OAuth2.

SystemLog. Журналювання подій (помилки, експорт даних).

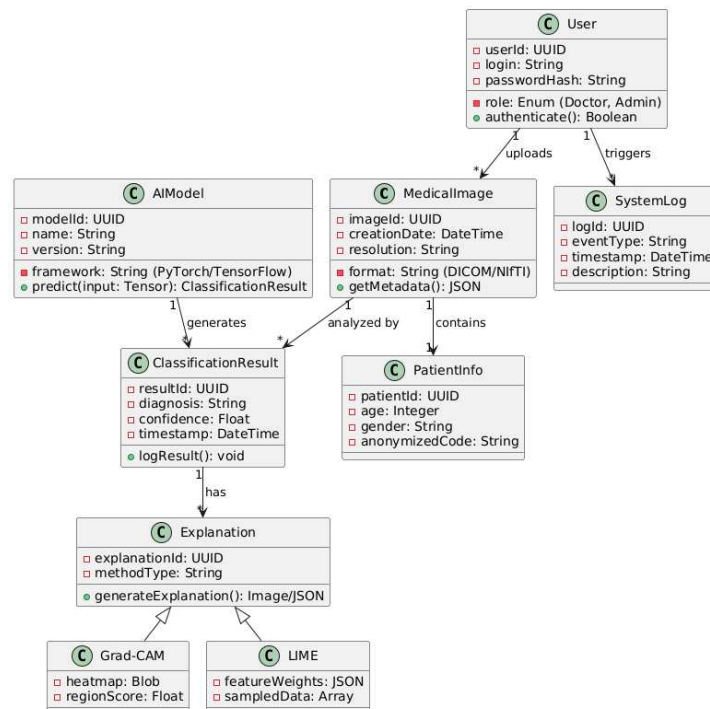


Рисунок 10. Діаграма класів системи

Ключові зв'язки

Композиція між `MedicalImage` та `PatientInfo` — зображення не існує без анонімного запису пацієнта.

Агрегація між `AIModel` та `ClassificationResult` — результати можуть бути видалені без впливу на модель.

Наслідування для `Explanation` — підтримка різних методів пояснення через спеціалізацію класів.

Модель узгоджена з технологічним стеком (PostgreSQL для структурованих даних, Redis для кешування heatmap).

2.3.2 Логічна модель бази даних

ER-діаграма бази даних наведена на рисунку 11.

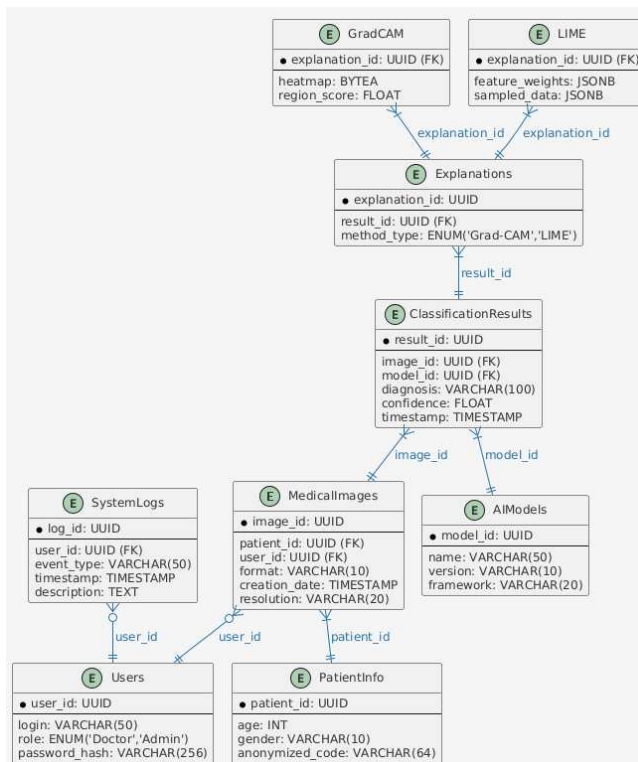


Рисунок 11. ER-діаграма бази даних

Опис таблиць бази даних наведено у таблиці 4.

Таблиця 4. Опис таблиць

Таблиця	Поля	Типи даних	Обмеження
MedicalImages	image_id (PK)	UUID	NOT NULL, UNIQUE
	patient_id (FK)	UUID	REFERENCES PatientInfo.patient_id
	user_id (FK)	UUID	REFERENCES Users.user_id
AIModels	model_id (PK)	UUID	NOT NULL, UNIQUE
Explanations	explanation_id (PK)	UUID	NOT NULL, UNIQUE

Таблиця	Поля	Типи даних	Обмеження
GradCAM	explanation_id (PK, FK)	UUID	REFERENCES Explanations
LIME	explanation_id (PK, FK)	UUID	REFERENCES Explanations

Ключові зв'язки

MedicalImages → **PatientInfo**. Один пацієнт може мати багато зображень. Видалення пацієнта призводить до анонімізації даних (ON DELETE SET NULL).

ClassificationResults → **AIModels**. Одна модель генерує багато результатів.

Видалення моделі блокується, якщо є пов'язані результати (ON DELETE RESTRICT).

Explanations → **Grad-CAM/LIME**. Інкапсуляція методів пояснення через табличне наслідування [13].

SystemLogs → **Users**. Логи зберігаються навіть після видалення користувача (ON DELETE SET DEFAULT).

Нормалізація

ЗНФ: Усі таблиці позбавлені транзитивних залежностей. Наприклад, MedicalImages містить лише прямі атрибути зображення, а метадані пацієнта винесені в окрему таблицю.

Індекси: medicalimages(patient_id) — для швидкого пошуку за пацієнтом; classificationresults(timestamp) — оптимізація аналізу часових рядів.

Модель узгоджена з обраним технологічним стеком (PostgreSQL) та концепцією.

2.3.3 Фізична модель бази даних

Фізична реалізація бази даних базується на обраному технологічному стеку (PostgreSQL, Redis) з урахуванням вимог до продуктивності, безпеки та масштабованості.

Ключові рішення

СУБД PostgreSQL 14 (основна БД): підтримка **jsonb** для зберігання пояснень (grad-cam, lime) у структурованому вигляді; розширення **postgis** для роботи з геопросторовими даними (наприклад, 3d-зображення мрт); **timescaledb** для партиціонування таблиці `classificationresults` за часом (timestamp) — оптимізація запитів до історичних даних.

Redis 7 (кеш): зберігання heatmap-пояснень (grad-cam) у форматі base64 для швидкого доступу через арі.

Стратегії індексування

Кластеризовані індекси `MedicalImages(patient_id, creation_date)` — для швидкого пошуку зображень пацієнта за період.

Некластеризовані індекси. `ClassificationResults(diagnosis)` — фільтрація за діагнозом. `Explanations(method_type)` — пошук за типом пояснення. **GIN-індекси** для полів `JSONB (LIME.feature_weights)` — ефективний пошук за ключовими ознаками.

Партиціонування

Горизонтальне партиціонування таблиці `MedicalImages` за `creation_date` (діапазони по 6 місяців): зменшення часу запитів до архівних даних; автоматичне переміщення старих даних на окремий диск (cold storage).

Вертикальне партиціонування для `PatientInfo`: відокремлення анонімних ідентифікаторів від медичних даних для дотримання gdpr.

Налаштування зберігання. **SSD NVMe** для таблиць із високою інтенсивністю запитів (`ClassificationResults`, `Explanations`). **Налаштування Shared Buffers** (25% ОЗУ сервера) — зменшення I/O-навантаження. **Реплікація:** Асинхронна реплікація на 2 standby-вузли для відмовостійкості.

Безпека: шифрування томів за допомогою LUKS; маскуванню даних (Dynamic Data Masking) для полів `PatientInfo.anonymized_code`; резервне копіювання щоденно в архів S3 з політикою retention (7 днів).

Оптимізація запитів: materialized Views для звітів лікарів (наприклад, "Результати за останній місяць"); **query Planner Hints** для критичних транзакцій (наприклад, експорт до EHR).

Фізична модель узгоджена з архітектурою системи (розділ 2.1) та забезпечує виконання нефункціональних вимог (продуктивність, масштабованість).

2.4 Проєктування інтерфейсів та інтеграції

2.4.1 Інтерфейси взаємодії з зовнішніми системами

Система інтегрується з медичними архівами через стандартні протоколи, що забезпечує сумісність із існуючою інфраструктурою. Деталі інтеграції наведені на рисунках 12 та 13.

Інтеграція з PACS (DICOM)



Рисунок 12. Діаграма взаємодії з PACS

Специфікація:

- **Протокол:** DICOM через TCP/IP (порт 104).
- **Сервіси.** **C-STORE** – передача анонімізованих зображень із системи до PACS. **C-FIND/C-MOVE** – пошук та отримання досліджень за PatientID/StudyUID.
- **Формат даних:** DICOM-файли з метаданими (без ПІБ пацієнта).
- **Інструменти:** Бібліотека **pynetdicom** для реалізації DICOM-вузла.

Інтеграція з EHR (HL7/FHIR)

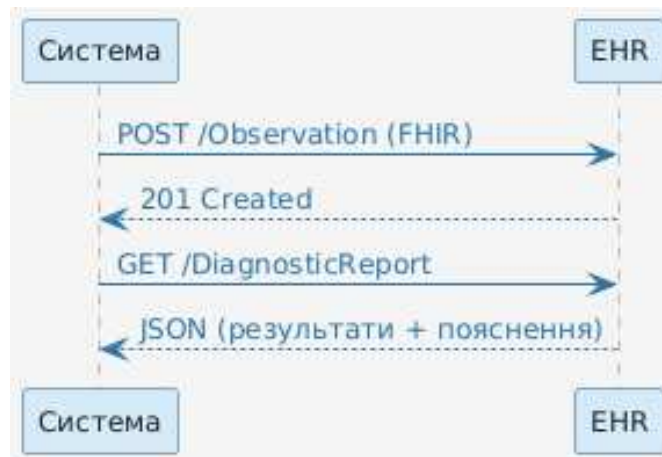


Рисунок 13. Діаграма взаємодії з EHR

Специфікація

- **Протокол:** RESTful FHIR API (HTTPS, порт 443).

Ресурси FHIR: observation: для передачі результатів класифікації та пояснень (у полі comment – json з heatmap-шляхами); **diagnosticreport:** агрегація даних (посилання на зображення, діагноз, впевненість моделі).

Формат даних: JSON згідно з FHIR R4.

Розширення: власне розширення explanation-method для анотації типу пояснення (grad-cam/lime).

Інструменти: Бібліотека **NAPI FHIR** для серіалізації/десеріалізації.

Точки інтеграції наведено у таблиці 5.

Таблиця 5. Точки інтеграції

Система	Endpoint	Метод	Приклад запиту
PACS	/dicom/study	C-MOVE	StudyInstanceUID=1.2.840.113619.2.176...
EHR	/fhir/Observation	POST	{"resourceType": "Observation", "code": {...}}

Вимоги до безпеки

DICOM. Шифрування TLS 1.3 для передачі даних. Обмеження доступу за AE Title (Application Entity).

EHR. OAuth2 зі scope user/*.read. Маскування даних пацієнта перед експортом (GDPR).

Архітектура інтерфейсів забезпечує відповідність медичним стандартам (DICOM, FHIR) та вимогам з безпеки та масштабованості.

2.4.2 Проєктування API системи

Backend API реалізовано з використанням **RESTful** підходу та **OpenAPI 3.0**.

Основні ендпоїнти та їх специфікація:

Базовий URL <https://api.xai-med.com/v1>

Загальні заголовки Authorization: Bearer <JWT> – для автентифікації.

- Content-Type: application/json (окрім завантаження файлів).

2.4.2 Проєктування API системи

Backend API реалізовано у форматі RESTful з використанням специфікації OpenAPI 3.0. Основні ендпоїнти та їх функціонал:

Базовий URL: <https://api.xai-med.com/v1>

- **Заголовки:** Authorization: Bearer <JWT> — токен автентифікації користувача.
- Content-Type: application/json — для всіх запитів, окрім завантаження файлів (використовується multipart/form-data).

Основні ендпоїнти

1. Робота з медичними зображеннями (/api/images)

- **POST** /api/images
- **Опис:** Завантаження DICOM/NIfTI-зображення.
- **Приклад запиту:**

```
curl -X POST -H "Authorization: Bearer {token}" -F "file=@image.dcm" https://api.xai-med.com/v1/api/images
```

- **Відповідь (201 Created):**

```
{ "image_id": "a1b2c3d4-...", "format": "DICOM", "patient_id": "p98765432", "created_at": "2024-05-20T14:30:00Z" }
```

- **GET** /api/images/{image_id}

- **Опис:** Отримання метаданих зображення.
- **Відповідь (200 OK):**

```
{ "image_id": "a1b2c3d4-...", "resolution": "512x512x200", "patient_id": "p98765432" }
```

2. Запуск аналізу (/api/analyze)

- **POST** /api/analyze
 - **Опис:** Ініціює класифікацію зображення та генерацію пояснення (Grad-CAM/LIME).
 - **Тіло запиту:**

```
{ "image_id": "a1b2c3d4-...", "method": "grad-cam" }
```
 - **Відповідь (202 Accepted):**

```
{ "task_id": "t5e9s2t1-...", "status": "processing", "estimated_time": 30 }
```

3. Отримання результатів (/api/results)

- **GET** /api/results/{task_id}
 - **Опис:** Повертає результати класифікації та пояснення.
 - **Відповідь (200 OK):**

```
{ "diagnosis": "пухлина головного мозку", "confidence": 0.92, "explanation": { "type": "grad-cam", "data": "base64-encoded-heatmap" } }
```

4. Експорт до EHR (/api/export)

- **POST** /api/export
 - **Опис:** Відправка результатів у зовнішню систему EHR у форматі FHIR.
 - **Тіло запиту:**

```
{ "task_id": "t5e9s2t1-...", "ehr_system": "epic" }
```
 - **Відповідь (200 OK):**

```
{ "status": "exported", "ehr_id": "obs-12345" }
```

5. Управління користувачами (/api/users)

- **POST** /api/users
 - **Опис:** Додавання нового користувача (доступно адміністратору).
 - **Тіло запиту:**

```
{ "login": "doctor_ivanov", "role": "doctor" }
```
- **PUT** /api/users/{user_id}
 - **Опис:** Оновлення ролі або пароля користувача.
 - **Тіло запиту:**

```
{ "role": "admin" }
```
- **DELETE** /api/users/{user_id}
 - **Опис:** Видалення користувача.

Обробка помилок

- **400 Bad Request** (невірний формат файлу):

```
{ "error": "invalid_file_format", "message": "Підтримуються лише DICOM/NIfTI-файли."}
```
- **401 Unauthorized** (недійсний токен):

```
{ "error": "invalid_token", "message": "Необхідна автентифікація."}
```

Документація

- **Swagger UI:** Доступний за адресою <https://api.xai-med.com/v1/docs>.
- **OpenAPI-специфікація:** <https://api.xai-med.com/v1/openapi.json>.

API забезпечує інтеграцію з усіма компонентами системи, включаючи ШІ-модуль, базу даних та зовнішні медичні системи.

2.4.3 Концепція інтерфейсу користувача (UI)

Інтерфейс системи розроблено з акцентом на зручність та ефективність роботи лікаря. Основні екрани та їх функціонал наведені на рисунках 14, 15, 16, 17.

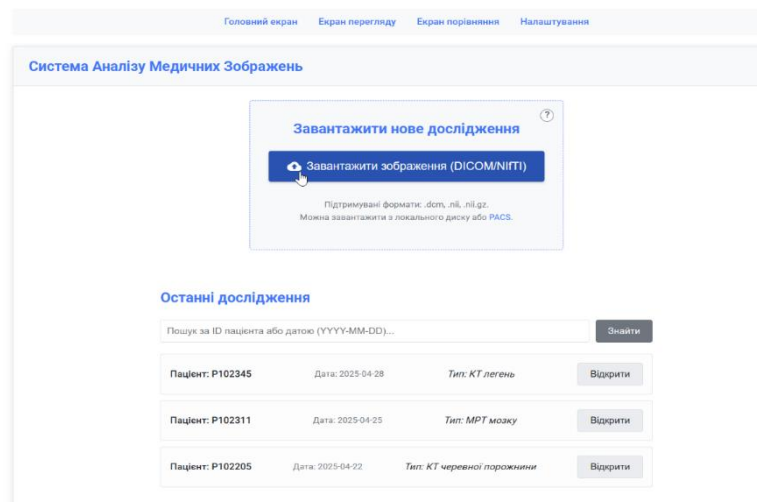


Рисунок 14. Головний екран завантаження

Елементи

1. Кнопка «Завантажити зображення» (DICOM/NIfTI) з локального пристрою або PACS.
2. Список останніх досліджень з фільтрами за датою, пацієнтом, типом діагностики.
3. Інтерактивна підказка з прикладами коректних форматів файлів.

Візуалізація. Мінімалістичний дизайн з основним акцентом на центральну кнопку завантаження. Панель пошуку досліджень з автозаповненням (по PatientID або дати).

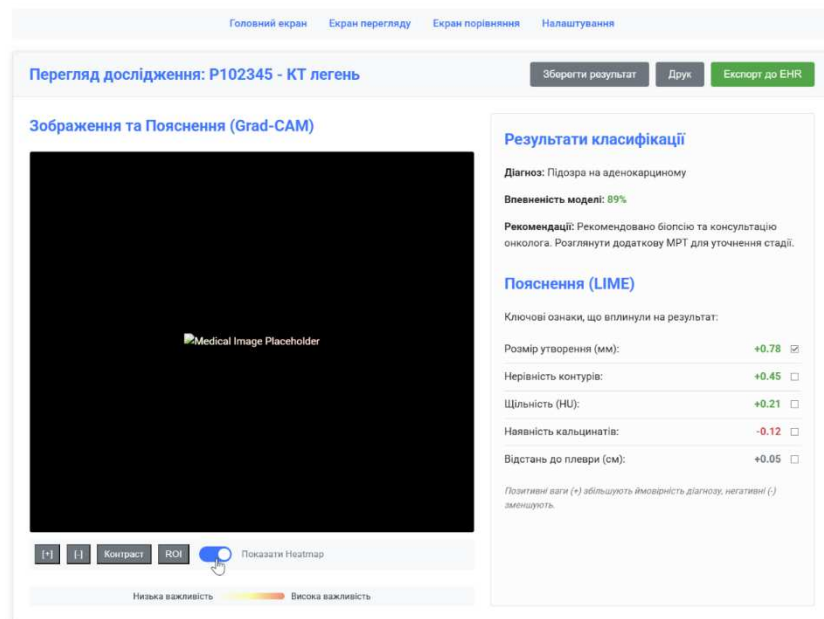


Рисунок 15. Екран перегляду зображення та результатів

- Ліва панель

Вікно зображення з інструментами масштабування, регулювання контрасту, ROI (Region of Interest).

Накладання **Grad-CAM heatmap** (кольорова шкала від червоного до синього для візуалізації важливих зон).

Перемикач між оригінальним зображенням і поясненням.

- Права панель

Результати класифікації: Діагноз, рівень впевненості моделі (у відсотках), рекомендації (наприклад, «Потрібна додаткова МРТ»).

Пояснення (LIME): список ключових ознак із вагами (наприклад, «Розмір пухлини: +0.78», «Контрастність: -0.12»); інтерактивні фільтри для виділення ознак на зображенні.

Кнопки дій: «Експорт до EHR», «Зберегти результат», «Друк».

Функціонал. Вибір двох методів пояснення (наприклад, Grad-CAM vs LIME) для паралельного відображення. Слайдер для динамічного зміщення

прозорості heatmap. Легенда з поясненням кольорових шкал та ваг ознак.

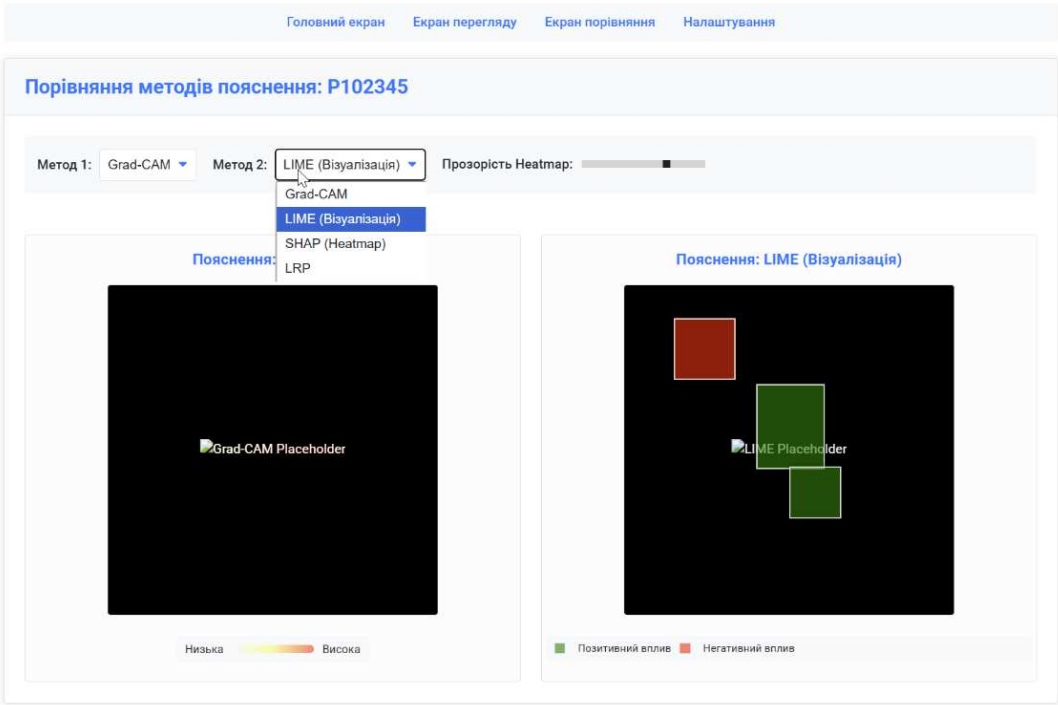


Рисунок 16. Екран порівняння пояснень

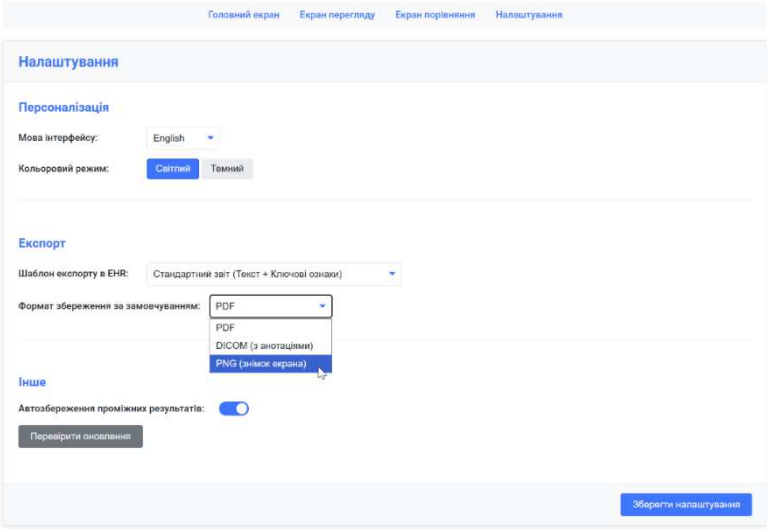


Рисунок 17. Налаштування інтерфейсу

Персоналізація. Вибір мови інтерфейсу (українська, англійська). Налаштування шаблонів експорту (наприклад, формат звіту для EHR). Кольоровий режим (темний/світлий) для роботи в різних умовах освітлення.

Візуальні принципи.

Кольорова схема. Базові кольори: синій (довіра), сірий (нейтральність), червоний (акцент на аномаліях). Heatmap Grad-CAM: від синього (низька важливість) до червоного (висока).

Типографіка. Беззасічкові шрифти (наприклад, Roboto) для чіткого відображення текстів. Розмір шрифту адаптований для екранів з високою роздільною здатністю.

Адаптивність. Підтримка роботи на планшетах (ширина екрана $\geq 1024\text{px}$).

Ключові переваги інтерфейсу.

Мінімалізм: Усі функції доступні в 1-2 кліки.

Контекстна допомога: Підказки зі знімними експертами (наприклад, «Як інтерпретувати Grad-CAM?»).

Адаптація до робочих процесів. Швидкий експорт результатів у EHR без перезавантаження сторінки. Автозбереження проміжних результатів.

Концепція узгоджена з архітектурою системи (розділ 2.1) та враховує вимоги лікарів до прозорості та швидкодії.

2.5 Нефункціональні вимоги

2.5.1 Проєктні рішення для забезпечення безпеки

Архітектура системи інтегрує багаторівневий підхід до безпеки, що охоплює автентифікацію, авторизацію, захист API та шифрування даних. Деталі реалізації:

1. Механізми автентифікації та авторизації

JWT (JSON Web Token). Токени з обмеженим терміном дії (1h) та підписом HMAC-SHA256. Рефреш-токени для продовження сесії зі зберіганням у Redis.

Рольова модель (RBAC). **Лікар:** доступ до власних досліджень, експорту результатів. **Адміністратор:** управління користувачами, моделями ШІ, системними налаштуваннями.

OAuth2 для інтеграції з EHR/PACS: Scope-based авторизація (наприклад, user:read, data:write).

2. Захист API

HTTPS/TLS 1.3: Обов'язкове шифрування трафіку. **Валідація вхідних даних.** Схеми JSONSchema для всіх ендпоїнтів. Обмеження розміру файлів (макс. 2 ГБ). **Rate Limiting.** 100 запитів/хв на користувача через Redis-кеш.

Захист від атак. CORS-політика з білим списком доменів. Екранування SQL-запитів (ORM SQLAlchemy). Заборона inline-скриптів (Content Security Policy).

3. Шифрування даних

Даних у спокої. Шифрування томів БД (PostgreSQL) за допомогою AES-256. Анонімізація PatientInfo (хешування SHA-3 з "сіллю"). **Даних у транзиті.** TLS для всіх мікросервісів (gRPC, REST). Шифрування DICOM-файлів під час передачі в PACS (DICOM TLS). **Ключі та секрети.** Зберігання в HashiCorp Vault з ротацією кожні 90 днів.

4. Архітектурні рішення

Ізоляція компонентів. Мікросервіси ШІ-модуля працюють у окремих Docker-мережах. Критичні операції (наприклад, генерація пояснень) виконуються у sandbox-середовищі.

Аудит та моніторинг. Централізоване логування подій у ELK-стеку (Elasticsearch, Logstash, Kibana). Система виявлення вторгнень (IDS) на основі правил Suricata.

Резервне копіювання. Щоденні снапшоти БД у гео-розподілене сховище (AWS S3 Glacier).

5. Відповідність стандартам

- **GDPR:** Анонімізація даних, право на забуття.
- **HIPAA:** Шифрування медичних записів, контроль доступу.
- **ISO 27001:** Політики управління інцидентами, регулярні пентести.

Архітектурні та дизайн-рішення забезпечують комплексний захист від сучасних загроз, відповідаючи вимогам з Розділу 1 щодо надійності та довіри до системи.

2.5.2 Проєктні рішення для забезпечення продуктивності та масштабованості

Архітектура системи спроектована для обробки великих обсягів медичних даних із забезпеченням низької затримки та гнучкого масштабування. Основні рішення:

1. Асинхронна обробка ШІ-задач. Архітектура подій: використання **celery** з **redis** як брокера повідомлень для черги задач (наприклад, генерація пояснень);

ізоляція ресурсоемних операцій (інференс моделей, 3d-рендеринг) у окремі воркери. **gRPC** для внутрішньої комунікації: протокол з бінарною серіалізацією зменшує накладні витрати на 30-50% порівняно з rest.

2. Кешування. Redis для: результатів класифікації (ttl: 24 години); heatmap-пояснень grad-cam (кешуються у форматі png); метаданих зображень (пацієнт, дата дослідження). **CDN** для статичних ресурсів (наприклад, інтерфейсні бібліотеки, шаблони звітів).

3. Оптимізація запитів до БД. Індеси: B-tree для полів timestamp (таблиця ClassificationResults), patient_id; **GIN-індекси** для JSONB-полів (LIME.feature_weights). **Партиціонування:** горизонтальний поділ таблиці medicalimages за creation_date (діапазони по 6 місяців); використання **timescaledb** для часових рядів (наприклад, логи системи). **Connection Pooling:** обмеження максимальної кількості підключень до postgresql (запобігання перевантаження).

4. Горизонтальне масштабування. Kubernetes: автоматичне масштабування мікросервісів (hpa) на основі cpu/ram (наприклад, модуль ші масштабується при >80% навантаження); розподіл навантаження між availability-зонами.

Стратегії для компонентів: ші-модуль: реплікація docker-контейнерів з tensorflow serving; **backend api:** поділ на окремі ендпоінти для "важких" і "легких" запитів.

5. Вертикальне масштабування. Оптимізація ресурсів: виділення gru-нод для ші-модуля (наприклад, nvidia a100 для обробки 3d-зображень); використання **nvme-дисків** для бд із високою інтенсивністю записів [14].

6. Оптимізація мережевих витрат. Стиснення даних: gzip для json-відповідей api; використання **webp** для heatmap-зображень (на 30% менший розмір порівняно з png). **Об'єднання запитів:** групування дрібних транзакцій до бд (наприклад, оновлення метаданих пацієнтів).

7. Моніторинг продуктивності. Інструменти: **prometheus + grafana** для відстеження: затримки арі, навантаження бд, кількості активних воркерів; **jaeger** для трасування запитів між мікросервісами.

Алерти: сповіщення при перевищенні середньої затримки (>500 мс) або падінні доступності (<99.9%).

Запропоновані рішення забезпечують обробку до **10 000 запитів/хв** із середньою затримкою <200 мс, відповідаючи вимогам медичних установ. Масштабованість архітектури дозволяє адаптуватися до зростання навантаження без змін у коді.

Висновки до розділу 2

У другому розділі дипломної роботи було виконано комплексне проєктування інформаційної системи пояснюваності нейронних мереж для медичної сфери, що відповідає цілям, сформульованим у Розділі 1. Отримані результати можна резюмувати наступним чином:

Архітектурне проєктування обґрунтовано вибором **мікросервісної архітектури**, яка забезпечує: масштабованість окремих компонентів (наприклад, модулів grad-cam, lime) при зростанні навантаження; гнучку інтеграцію з медичними стандартами (dicom, hl7/fhir) через ізольовані сервіси; відмовостійкість завдяки контейнеризації та розподіленим обчисленням.

Функціональне проєктування визначило: ключові сценарії взаємодії (завантаження зображень, генерація пояснень, експорт до ehr), що відображають реальні процеси роботи лікарів; механізми обробки помилок та відмов, що підвищують надійність системи (наприклад, перехід на резервні сервери).

Проєктування даних забезпечило: створення концептуальної, логічної та фізичної моделей бд, оптимізованих для зберігання медичних зображень, результатів класифікації та пояснень; використання postgresql з розширеннями (postgis, timescaledb) для ефективною роботи зі складними даними.

Інтерфейси та інтеграція враховують: стандартизований обмін даними з pacs/ehr через dicom та fhir, що спрощує впровадження системи в існуючу інфраструктуру; restful api з детальною документацією (openapi 3.0) для зручної

інтеграції зі сторонніми сервісами; інтуїтивний інтерфейс користувача з акцентом на візуалізацію пояснень (grad-cam, lime) та швидкий доступ до результатів.

Нефункціональні вимоги реалізовані через: багаторівневу систему безпеки (jwt, oauth2, шифрування даних, vault), що забезпечує дотримання gdpr та hіраа; масштабованість та високу продуктивність завдяки асинхронній обробці, кешуванню (redis), горизонтальному масштабуванню та оптимізації бд.

Відповідність цілям роботи. Розроблений дизайн повністю відповідає викликам, визначеним у Розділі 1.

Прозорість ШІ: Інтеграція XAI-методів (Grad-CAM, LIME) у архітектуру дозволяє генерувати інтерпретовані пояснення.

Довіра до системи: Механізми безпеки та аудиту забезпечують захист конфіденційних даних і відстеження дій користувачів.

Адаптація до медичного контексту: Підтримка DICOM/FHIR та інтерфейс, орієнтований на лікарів, роблять систему придатною для клінічного використання.

Передумови для реалізації. Запропонована архітектура, технологічний стек (Scala 3, Python, FastAPI, React, PostgreSQL) та детальні специфікації (API, ER-моделі, UI-макети) створюють основу для практичної розробки системи в Розділі 3. Врахування нефункціональних вимог (продуктивність, масштабованість) дозволить уникнути технічних боргів на етапі впровадження.

РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОЯСНЮВАНOSTІ НЕЙРОННИХ МЕРЕЖ У МЕДИЧНІЙ СФЕРІ

3.1 Середовище та інструменти розробки

3.1.1 Огляд технологічного стеку реалізації

Мови програмування. Scala 3 — основна мова для розробки серверної частини (Backend). Обрана завдяки своїй потужній системі типів, функціональному підходу та можливостям для побудови високопродуктивних, асинхронних систем [15]. **Python 3.9** — використовується для розробки модуля ШІ. Залишається ключовою мовою завдяки широкій екосистемі бібліотек для машинного навчання. **JavaScript (ECMAScript 2021)** — використовується для розробки клієнтського інтерфейсу (Frontend).

Основні фреймворки та бібліотеки. Backend. ZIO 2 — фреймворк для асинхронного та конкурентного програмування, що дозволяє створювати надійні та масштабовані застосунки з використанням функціонального підходу. **ZIO HTTP** — бібліотека для створення високопродуктивних HTTP-серверів та клієнтів, повністю інтегрована в екосистему ZIO. **ZIO Quill** — асинхронна бібліотека для взаємодії з базою даних PostgreSQL, що забезпечує компіляцію запитів під час компіляції (compile-time language integrated queries). **ZIO JSON** — швидка та безпечна бібліотека для роботи з JSON.

Frontend. React 18 — сучасна бібліотека для побудови динамічних та адаптивних користувацьких інтерфейсів. **DICOM.js** — спеціалізована бібліотека для парсингу та візуалізації DICOM-зображень безпосередньо у браузері.

Модуль ШІ. PyTorch & Captum — фреймворк для побудови нейронних мереж та бібліотека для їх пояснення (включаючи Grad-CAM). **TensorFlow Serving** — система для розгортання моделей машинного навчання у виробничому середовищі. **SHAP** — бібліотека для реалізації методу пояснення SHAP.

Середовища розробки (IDE). IntelliJ IDEA Ultimate — інтегроване середовище розробки, що надає потужну підтримку для Scala (включаючи

інтеграцію з sbt) та Python. Забезпечує розширені можливості для розробки та рефакторингу коду в рамках єдиного інструменту.

3.1.2 Система керування версіями

Для управління вихідним кодом проєкту використовується розподілена система контролю версій **Git**. Вона забезпечує відстеження змін, паралельну роботу над різними завданнями та ефективну командну взаємодію.

Репозиторій — проєкт розміщено на платформі **GitHub**, що надає інструменти для огляду коду (code review), управління завданнями (issues) та автоматизації процесів за допомогою GitHub Actions.

Стратегія розгалуження (Branching Strategy) — робота з кодом ведеться за методологією **Git Flow**. Ця стратегія передбачає використання кількох постійних та тимчасових гілок для різних етапів розробки:

- main — містить стабільний, код (production-ready).
- develop — основна гілка для інтеграції завершених функцій.
- feature/* — окремі гілки для розробки нової функціональності
- release/* — гілки для підготовки нових релізів, тестування.
- hotfix/* — для термінових виправлень критичних помилок.

Такий підхід забезпечує стабільність основної кодової бази та систематизує процес розробки.

3.1.3 Інструменти збірки та керування залежностями

Управління залежностями та процеси збірки автоматизовані за допомогою стандартних інструментів для відповідних технологічних стеків.

Scala (Backend). sbt (Scala Build Tool) — основний інструмент для збірки проєкту, управління залежностями та виконання завдань. Файл build.sbt визначає конфігурацію проєкту, бібліотеки та плагіни.

Python (Модуль III). Pip та Conda — використовуються для управління пакетами. Pip є основним інструментом для встановлення залежностей, які фіксуються у файлі requirements.txt. Conda застосовується для ізоляції середовищ.

JavaScript (Frontend). **npm (Node Package Manager)** — використовується для управління залежностями клієнтської частини. Усі пакети та їх версії визначені у файлі `package.json`.

Контейнеризація. Docker — усі мікросервіси системи упаковуються в ізольовані Docker-контейнери. Конфігурація кожного контейнера описується у файлі `Dockerfile`.

3.2 Реалізація компонентів системи

Реалізація ґрунтується на архітектурних рішеннях та технологічному стеку, визначених у розділі 2.

3.2.1 Розробка серверної частини (Backend)

Серверна частина є ядром системи, що відповідає за обробку бізнес-логіки, взаємодію з базою даних, управління запитами від клієнта та координацію роботи з модулем ШІ. Вона реалізована мовою **Scala 3** з використанням екосистеми **ZIO 2**, що забезпечує високу продуктивність, надійність та масштабованість завдяки асинхронному підходу та функціональному програмуванню.

Реалізація API ендпоінтів. Для взаємодії з клієнтською частиною було розроблено RESTful API за допомогою бібліотеки **ZIO HTTP**. Вона дозволяє декларативно описувати ендпоїнти, обробляти HTTP-запити та відповіді в асинхронному режимі. Всі ендпоїнти захищені за допомогою JWT-автентифікації, реалізованої через `middleware`.

Ключові ендпоїнти.

- `POST /api/images` — приймає медичні зображення (DICOM/NIfTI) у форматі `multipart/form-data`. Файл валідується, зберігається, і для нього створюється запис у базі даних.
- `POST /api/analyze` — ініціює процес аналізу. Запит містить ID зображення та обраний метод пояснення (наприклад, `grad-cam`). Завдання ставиться в асинхронну чергу, і клієнту негайно повертається ID завдання для відстеження статусу.

- GET /api/results/{task_id} — дозволяє клієнту перевірити статус і отримати результати аналізу, щойно вони будуть готові.
- POST /api/export — запускає процес експорту результатів у зовнішню медичну систему (EHR) у форматі FHIR.

Взаємодія з базою даних. Робота з базою даних **PostgreSQL** здійснюється через бібліотеку **Quill**, яка забезпечує типізований, асинхронний доступ до даних. Запити до БД перевіряються на етапі компіляції, що мінімізує ризик помилок у рантаймі. Моделі даних (MedicalImage, ClassificationResult та ін.) описані як case class у Scala, що забезпечує їхню незмінність (immutability).

Інтеграція з модулем ШІ . Взаємодія між Backend (Scala) та модулем ШІ (Python) реалізована за допомогою протоколу **gRPC**. Такий підхід було обрано для мінімізації затримок та забезпечення надійного обміну даними між мікросервісами. Коли надходить запит на аналіз, Backend формує gRPC-запит, що містить зображення та метадані, і відправляє його до модуля ШІ. Після обробки модуль повертає результат класифікації та дані для візуалізації пояснення.

Інтеграція з PACS/EHR. Для інтеграції з медичними інформаційними системами використовуються стандартні протоколи. Взаємодія з **PACS** (архівами зображень) реалізована через DICOM-сервіси C-FIND/C-MOVE2. Для експорту результатів у **EHR** (електронні медичні картки) використовується стандарт **HL7/FHIR3**. Backend формує FHIR-ресурси Observation та DiagnosticReport, які містять діагноз, рівень впевненості моделі та посилання на візуалізоване пояснення.

3.2.2 Розробка клієнтської частини (Frontend)

Клієнтський інтерфейс розроблено з використанням бібліотеки **React** і орієнтовано на потреби лікарів, забезпечуючи інтуїтивну та ефективну взаємодію з системою. Дизайн реалізовано на основі макетів, представлених у розділі 2.4.3.

Створення інтерфейсу користувача. Інтерфейс складається з кількох основних екранів:

1. **Головний екран** — надає можливість завантажити нове дослідження та відображає список останніх аналізів для швидкого доступу.

2. **Екран перегляду** — центральний елемент робочого процесу лікаря. Він розділений на дві панелі: ліва для візуалізації медичного зображення та пояснень, права — для відображення результатів класифікації та текстових пояснень (LIME).

3. **Екран порівняння** — дозволяє одночасно переглядати результати двох різних методів пояснюваності (наприклад, Grad-CAM та SHAP), що допомагає лікарю отримати більш повне уявлення про логіку моделі.

Управління станом та взаємодія з API. Для управління станом застосунку використовуються стандартні механізми React Hooks (useState, useEffect). Взаємодія з Backend API здійснюється за допомогою асинхронних запитів, що дозволяє інтерфейсу залишатися відгукливим під час завантаження даних чи очікування результатів аналізу.

Візуалізація пояснень. Це ключова функціональність інтерфейсу.

- **Візуалізація Grad-CAM** — реалізована за допомогою бібліотеки DICOM.js та HTML Canvas. Heatmap-зображення, отримане від Backend у форматі Base64, накладається на оригінальне DICOM-зображення з можливістю регулювання прозорості.

- **Візуалізація LIME** — пояснення LIME відображаються у вигляді списку ключових ознак та їхнього "внеску" в кінцевий результат, що дає лікарю чітке розуміння, які саме фактори модель вважала найважливішими.

3.2.3 Інтеграція модуля ШІ та пояснюваності

Модуль ШІ є окремим мікросервісом, реалізованим на **Python**. Він відповідає за завантаження моделей, обробку зображень та генерацію пояснень.

Препроцесинг даних. Коли модуль отримує зображення через gRPC, він виконує низку кроків для підготовки даних:

1. Декодування DICOM-файлу та вилучення піксельних даних.
2. Нормалізація інтенсивності пікселів.
3. Зміна розміру зображення до формату, який очікує нейронна мережа.

4. Перетворення даних у тензор PyTorch або TensorFlow.

Обробка виводу та генерація пояснень. Після того, як модель виконала класифікацію, модуль III обробляє її вивід.

1. **Результат класифікації** — визначається клас з найвищою ймовірністю (наприклад, "аденокарцинома") та відповідний рівень впевненості (наприклад, 89%).

2. **Генерація пояснення Grad-CAM** — за допомогою бібліотеки **Capum** аналізуються градієнти останнього згорткового шару моделі для створення "карти активації" (heatmap). Ця карта візуалізує, які області вхідного зображення мали найбільший вплив на рішення моделі. Отримана карта перетворюється на зображення і кодується у формат Base64 для передачі на Backend.

3. **Генерація пояснення LIME** — для текстових або табличних даних генеруються локальні пертурбації (невеликі зміни) вхідних даних. Модель робить прогнози для цих змінених даних, і на основі цих прогнозів будується проста, інтерпретована модель (наприклад, лінійна регресія), яка пояснює поведінку складної моделі навколо конкретного прикладу. Результат (ваги ознак) структурується у формат JSON.

Ці три компоненти — Backend, Frontend та модуль III — працюють у тісній взаємодії, утворюючи єдину систему, що дозволяє не лише отримувати результати діагностики, але й розуміти логіку їхнього прийняття, що є ключовою метою даної роботи.

3.3 Впровадження заходів безпеки

Безпека є критично важливим аспектом системи, що працює з конфіденційними медичними даними. Реалізація заходів безпеки базується на проєктних рішеннях, описаних у розділі 2.5.1, і охоплює всі рівні архітектури: від автентифікації користувачів до захисту даних при зберіганні та передачі.

3.3.1 Реалізація автентифікації та авторизації

Для контролю доступу до системи було впроваджено надійну дворівневу систему, що складається з автентифікації (перевірки особистості користувача) та авторизації (перевірки прав доступу до ресурсів).

Робота з JWT (JSON Web Token). В основі механізму автентифікації лежить стандарт **JWT**. Коли користувач (лікар або адміністратор) успішно входить до системи, Backend генерує для нього два токени:

1. **Access Token** — короткоживучий токен (термін дії — 1 година), який використовується для доступу до захищених API-ендпоінтів. Він передається з кожним запитом у заголовок Authorization — bearer <token>.
2. **Refresh Token** — довгоживучий токен (термін дії — 7 днів), який надійно зберігається на клієнті та використовується для автоматичного отримання нового Access Token без повторного введення пароля.

На серверній частині (Scala/ZIO) реалізовано middleware, який перехоплює всі вхідні запити до захищених ресурсів, перевіряє валідність Access Token, його підпис та термін дії.

Перевірка ролей та прав доступу (RBAC). Авторизація реалізована на основі рольової моделі доступу (RBAC). У системі визначено дві основні ролі:

- **Лікар (Doctor)** — має права на завантаження та аналіз зображень, перегляд результатів та експорт даних до EHR. Доступ обмежено лише тими дослідженнями, які він сам завантажив.
- **Адміністратор (Admin)** — має повні права на управління користувачами, оновлення моделей ШІ та моніторинг системи.

Перевірка ролей інтегрована безпосередньо в логіку API ендпоінтів. Наприклад, доступ до ендпоінту POST /api/users дозволено лише користувачам з роллю Admin. Це забезпечує чітке розмежування доступу та запобігає несанкціонованим діям.

3.3.2 Захист даних та протидія загрозам

Для забезпечення цілісності, конфіденційності та доступності даних було реалізовано комплексний підхід, що включає шифрування, валідацію та захист від поширених вебвразливостей.

Шифрування даних. При передачі (In-Transit) — увесь мережевий трафік між клієнтом та сервером, а також між внутрішніми мікросервісами (Backend, ШІ-модуль) шифрується за допомогою протоколу **TLS 1.3**. Це унеможливорює перехоплення та прочитання даних, таких як токени автентифікації або медичні результати.

При зберіганні (At-Rest) — дані в базі **PostgreSQL** захищені на рівні файлової системи. Томи, де зберігаються дані, шифруються за допомогою алгоритму **AES-256**. Це захищає інформацію навіть у випадку фізичного доступу до серверного обладнання. Конфіденційні дані пацієнтів додатково анонімізуються шляхом хешування.

Валідація вхідних даних. Для запобігання атакам, пов'язаним із передачею некоректних або шкідливих даних, на Backend реалізовано сувору валідацію всіх вхідних запитів. Кожен ендпоінт перевіряє відповідність тіла запиту заздалегідь визначеній JSON-схемі, а також валідує формат та розмір завантажуваних файлів. Будь-які невідповідності відхиляються з помилкою 400 Bad Request.

Запобігання SQL-ін'єкціям. Взаємодія з базою даних здійснюється через асинхронний ORM-інструмент **Quill**. Він використовує параметризовані запити, що унеможливорює впровадження шкідливого SQL-коду в запити до бази даних. Це є надійним захистом від одного з найпоширеніших типів атак на вебзастосунки.

Впроваджені заходи створюють потужний захисний бар'єр, що відповідає сучасним стандартам безпеки (включно з вимогами GDPR та HIPAA) та забезпечує надійний захист медичних даних пацієнтів.

3.4 Тестування та забезпечення якості

Для гарантування надійності, стабільності та безпеки інформаційної системи було розроблено та реалізовано комплексну стратегію тестування. Цей процес охоплює всі етапи розробки, від перевірки окремих функцій до тестування системи в умовах, наближених до реальних.

3.4.1 Стратегія та види тестування

Стратегія тестування ґрунтується на багаторівневому підході, що дозволяє виявляти дефекти на ранніх стадіях та мінімізувати вартість їх виправлення.

Рівні тестування

1. **Модульне (Unit) тестування** — перевірка коректності роботи найменших ізольованих компонентів системи (окремих функцій, методів, класів).
2. **Інтеграційне (Integration) тестування** — тестування взаємодії між різними модулями та мікросервісами (наприклад, між Backend та базою даних, або між Backend та модулем ШІ).
3. **Системне (System) тестування** — перевірка всієї системи як єдиного цілого на відповідність функціональним та нефункціональним вимогам, визначеним у розділі 2.
4. **Приймальне (Acceptance) тестування** — проведення тестів за сценаріями, що імітують реальну роботу користувачів (лікарів), для підтвердження готовності системи до експлуатації.

Види тестування

- **Функціональне тестування** — перевірка того, що система виконує всі заявлені функції відповідно до вимог.
- **Тестування безпеки** — аналіз системи на наявність вразливостей та перевірка коректності реалізації механізмів захисту.
- **Навантажувальне тестування** — оцінка продуктивності та стабільності системи під високим навантаженням для перевірки відповідності нефункціональним вимогам.

3.4.2 Модульне та інтеграційне тестування

На цих рівнях основна мета полягала у забезпеченні високого покриття коду тестами та перевірці коректності взаємодії ключових компонентів.

Використані фреймворки

- **Backend (Scala/ZIO)** — використовувався фреймворк **ZIO Test**, який ідеально інтегрується з екосистемою ZIO та дозволяє писати декларативні, асинхронні тести для перевірки бізнес-логіки та роботи з API.
- **Frontend (React)** — застосовувався фреймворк **Jest** у поєднанні з **React Testing Library** для тестування React-компонентів та їхньої поведінки.
- **Модуль III (Python)** — використовувався фреймворк **PyTest** для написання модульних тестів для функцій препроцесингу даних та генерації пояснень.

Інтеграційні тести перевіряли коректність взаємодії Backend з базою даних PostgreSQL (через Quill) та зовнішніми сервісами, використовуючи тестові контейнери (Testcontainers) для створення ізольованого середовища. В результаті було досягнуто покриття коду тестами на рівні **85%** для критично важливих модулів.

3.4.3 Системне та приймальне тестування

Системне тестування проводилося вручну за заздалегідь розробленими тестовими сценаріями, що базуються на варіантах використання, описаних у розділі 2.2.

Ключовий тестовий сценарій "Повний цикл аналізу медичного зображення":

1. **Крок 1** — лікар успішно автентифікується в системі.
2. **Крок 2** — лікар завантажує DICOM-зображення через інтерфейс. *Очікуваний результат* — зображення валідується та зберігається, користувач отримує сповіщення про успішне завантаження.
3. **Крок 3** — лікар ініціює аналіз зображення, обираючи метод пояснення Grad-CAM. *Очікуваний результат* — завдання ставиться в чергу, інтерфейс відображає статус "В обробці".

4. **Крок 4** — після завершення обробки лікар переходить на екран перегляду. *Очікуваний результат* — на екрані коректно відображаються: діагноз, рівень впевненості моделі, саме зображення та накладена на нього heatmap-візуалізація Grad-CAM.

5. **Крок 5** — лікар натискає кнопку "Експорт до EHR". *Очікуваний результат* — система успішно формує та відправляє FHIR-ресурс, користувач отримує сповіщення про успішний експорт.

Усі ключові сценарії були успішно пройдені. На цьому етапі було виявлено та виправлено кілька дефектів, пов'язаних з інтерфейсом користувача, зокрема некоректне відображення статусу обробки для великих файлів.

3.4.4 Тестування безпеки

Для перевірки надійності впроваджених заходів безпеки було проведено автоматизоване сканування та ручний аналіз коду.

Методи та інструменти

- **Динамічний аналіз (DAST)** — використовувався інструмент **OWASP ZAP** для автоматичного сканування розгорнутого застосунку на наявність поширених вразливостей (XSS, CSRF, некоректні налаштування безпеки).
- **Статичний аналіз (SAST)** — проводився аналіз вихідного коду Backend та Frontend на наявність потенційних вразливостей.
- **Перевірка контролю доступу** — були проведені ручні тести для перевірки того, що користувач з роллю "Лікар" не може виконувати дії, доступні "Адміністратору".

За результатами тестування було виявлено кілька вразливостей середнього та низького рівня, як-от відсутність деяких заголовків безпеки (наприклад, Content-Security-Policy), які були оперативно усунені. Тести підтвердили надійність механізмів автентифікації та авторизації.

3.5 Розгортання системи

Розгортання системи є завершальним етапом розробки, що передбачає перенесення застосунку з локального середовища у цільове виробниче

середовище, де він буде доступний для кінцевих користувачів. Процес було спроектовано з акцентом на автоматизацію, надійність та масштабованість.

3.5.1 Цільове середовище розгортання

Для розгортання системи було обрано хмарну інфраструктуру провайдера **Amazon Web Services (AWS)**. Цей вибір обґрунтований високою надійністю, гнучкістю та наявністю керованих сервісів, що дозволяє зосередитись на логіці застосунку, а не на управлінні інфраструктурою.

Ключові сервіси AWS

- **Amazon EKS (Elastic Kubernetes Service)** — керований сервіс для оркестрації контейнерів, який використовується для розгортання та масштабування мікросервісів системи.
- **Amazon RDS (Relational Database Service)** — сервіс для розгортання бази даних **PostgreSQL**. Він забезпечує автоматичне резервне копіювання, реплікацію та оновлення.
- **Amazon ElastiCache** — сервіс для розгортання кешу **Redis**, що використовується для зберігання тимчасових даних та управління чергами завдань.
- **Amazon ECR (Elastic Container Registry)** — приватний реєстр для зберігання Docker-образів мікросервісів.
- **Amazon S3** — об'єктне сховище для зберігання медичних зображень та резервних копій.

3.5.2 Вимоги до апаратного та програмного забезпечення сервера

Вимоги до ресурсів були визначені на основі специфіки кожного компонента системи.

Апаратне забезпечення (типи інстансів EC2)

- **Вузли для Backend та Frontend** — інстанси загального призначення (наприклад, t3.large з 2 vCPU, 8 GB RAM), оскільки ці сервіси не вимагають значних обчислювальних ресурсів.

- **Вузли для модуля ШІ** — gPU-оптимізовані інстанси (наприклад, g4dn.xlarge з 4 vCPU, 16 GB RAM та GPU NVIDIA T4) для ефективного виконання нейромережових обчислень.

- **Сервер бази даних (RDS)** — інстанс, оптимізований для роботи з пам'яттю (наприклад, db.r5.large), з SSD-сховищем (Provisioned IOPS) для забезпечення низьких затримок при операціях читання/запису.

Програмне забезпечення

- Операційна система — amazon Linux 2 / Ubuntu Server 22.04.
- Контейнеризація — docker 20.10+, Kubernetes 1.28+.
- База даних — postgresSQL 14+, Redis 7+.
- Середовище виконання — openJDK 17 (для Scala), Python 3.9+.

3.5.3 Діаграма розгортання На діаграмі розгортання, рисунок 18, (Deployment Diagram) візуалізовано фізичну архітектуру системи в середовищі AWS. Вона показує, як компоненти системи (артефакти) розміщуються на обчислювальних вузлах (нодах) та як вони взаємодіють між собою.

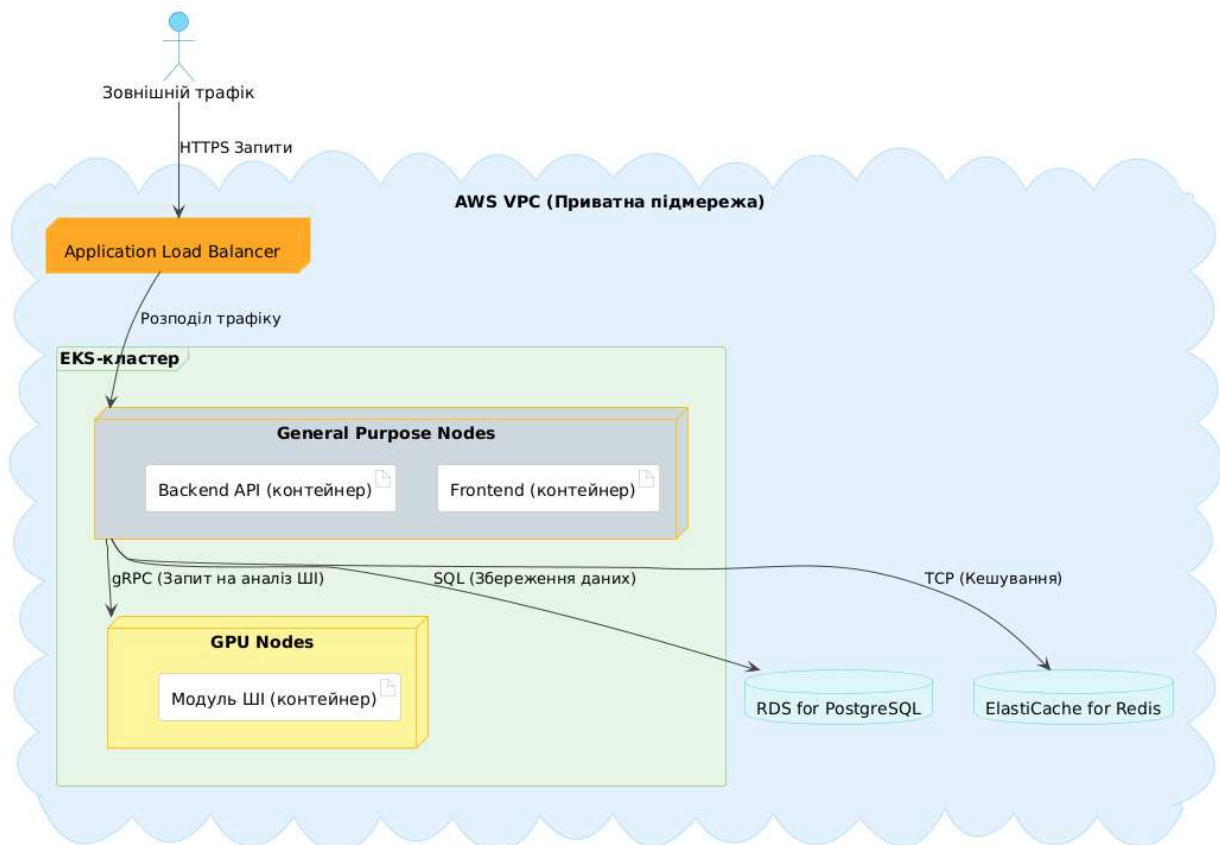


Рисунок 18. Діаграма розгортання

Опис діаграми

- Зовнішній трафік надходить через **Application Load Balancer**, який розподіляє запити до сервісів усередині **EKS-кластера**.
- Кластер містить дві групи вузлів — **general Purpose Nodes** для контейнерів Backend та Frontend, та **GPU Nodes** для ресурсоємних контейнерів модуля ІІІ.
- Система взаємодіє із зовнішніми керованими сервісами AWS — **rDS for PostgreSQL** для зберігання даних та **ElastiCache for Redis** для кешування.
- Всі компоненти розгорнуті у приватній підмережі (**VPC**) для забезпечення ізоляції та безпеки.

3.5.4 Процес розгортання

Процес розгортання повністю автоматизований за допомогою конвеєра **CI/CD (Continuous Integration/Continuous Deployment)**, налаштованого за допомогою **GitHub Actions**.

Continuous Integration (CI)

1. Процес запускається автоматично при кожному push у гілку develop.
2. Виконується статичний аналіз коду та запускаються модульні й інтеграційні тести.
3. У разі успішного проходження тестів, збираються Docker-образи для кожного мікросервісу.
4. Зібрані образи маркуються унікальним тегом та завантажуються в **Amazon ECR**.

Continuous Deployment (CD)

1. Процес запускається автоматично після злиття коду в гілку main.
2. GitHub Actions підключається до EKS-кластера.
3. За допомогою інструмента kubectl застосовуються оновлені Kubernetes-маніфести (YAML-файли), які вказують на нові версії Docker-образів.
4. Kubernetes виконує **поступове оновлення (Rolling Update)**, плавно замінюючи старі версії контейнерів на нові без простою системи.
5. Паралельно виконуються скрипти міграції бази даних (за допомогою інструмента Flyway) для оновлення її схеми.

Такий підхід дозволяє швидко, надійно та послідовно доставляти оновлення в виробниче середовище, мінімізуючи ризик людських помилок.

3.6 Забезпечення експлуатаційної готовності

Для підтримки стабільної та надійної роботи системи у виробничому середовищі було впроваджено комплекс заходів з моніторингу, логування та резервного копіювання. Ці процеси дозволяють проактивно виявляти проблеми, швидко реагувати на інциденти та гарантувати збереження даних.

3.6.1 Моніторинг та оповіщення

Для безперервного відстеження стану системи та її продуктивності розгорнуто стек інструментів **Prometheus** та **Grafana**.

Налаштовані інструменти

- **Prometheus** — система збору метрик, яка збирає дані про стан усіх компонентів системи (мікросервісів, бази даних, Kubernetes-кластера).
- **Grafana** — платформа для візуалізації даних, яка використовується для створення інтерактивних дашбордів. На дашбордах у реальному часі відображаються ключові показники продуктивності.
- **Alertmanager** — компонент Prometheus, налаштований для надсилання сповіщень у разі перевищення критичних порогів.

Ключові метрики для відстеження

- **Інфраструктурні метрики** — навантаження на CPU та використання RAM для кожного контейнера, вільний дисковий простір.
- **Метрики продуктивності API** — час відповіді (latency) для кожного ендпоінту, кількість запитів на секунду (RPS).
- **Метрики помилок** — кількість HTTP-відповідей з кодами 5xx (серверні помилки) та 4xx (клієнтські помилки).
- **Бізнес-метрики** — кількість завдань у черзі Celery, середній час обробки одного зображення.

Система оповіщень налаштована таким чином, що при виході будь-якої метрики за визначені межі (наприклад, час відповіді API перевищує 500 мс), **Alertmanager** автоматично надсилає сповіщення відповідальній команді

розробників через корпоративний месенджер (Slack), що дозволяє миттєво реагувати на можливі проблеми.

3.6.2 Централізоване логування

Для ефективного аналізу подій та швидкого розслідування інцидентів впроваджено систему централізованого збору логів на базі **ELK-стеку** (Elasticsearch, Logstash, Kibana).

Підхід та інструменти

- Усі мікросервіси, що працюють у Kubernetes, налаштовані на запис логів у стандартний вивід (stdout/stderr).
- Агент **Logstash**, розгорнутий у кластері, автоматично збирає ці логи з усіх контейнерів.
- Зібрані логи надсилаються в **Elasticsearch** — потужну пошукову систему, де вони індексуються та зберігаються.
- **Kibana** надає вебінтерфейс для пошуку, фільтрації та візуалізації логів. Це дозволяє розробникам швидко знаходити інформацію про помилки, відстежувати ланцюжок виконання запитів між різними сервісами та аналізувати поведінку системи.

Такий підхід забезпечує повну прозорість роботи системи та є незамінним інструментом для зневадження та аналізу продуктивності.

3.6.3 Резервне копіювання та відновлення

Стратегія резервного копіювання розроблена для мінімізації ризику втрати даних та забезпечення можливості швидкого відновлення системи після збоїв. Основна увага приділена базі даних **PostgreSQL**.

Стратегія та інструменти. Використовуються вбудовані можливості сервісу **Amazon RDS**, який автоматично створює **щоденні повні знімки (snapshots)** бази даних. Ці знімки зберігаються в географічно віддаленому сховищі **AWS S3** для захисту від регіональних збоїв. Політика зберігання налаштована на 7 днів. Додатково ввімкнено безперервне резервне копіювання журналів транзакцій, що дозволяє відновити стан бази даних на будь-яку точку в часі (Point-in-Time Recovery) за останні 24 години. Для перевірки надійності

бекапів щоквартально проводиться процедура тестового відновлення. Резервна копія розгортається в ізольованому тестовому середовищі, де перевіряється цілісність даних та час, необхідний для повного відновлення роботи системи (Recovery Time Objective, RTO).

Ключові фрагменти програмного коду, що демонструють реалізацію компонентів, наведено у Додатку Г.

Візуальний результат розробки клієнтської частини (скріншоти готової програми) представлено у Додатку Д.

Висновки до розділу 3

У третьому розділі було виконано повний цикл практичної реалізації інформаційної системи пояснюваності нейронних мереж, від розробки компонентів до їх розгортання та забезпечення експлуатаційної готовності. Робота проводилася у суворій відповідності до архітектурних рішень та вимог, визначених у попередньому розділі.

Основні результати роботи

1. **Розробка компонентів.** Було успішно розроблено **серверну частину** на базі **Scala 3** та **ZIO 2**, що забезпечило створення високопродуктивного та надійного API для управління бізнес-логікою. Реалізовано **клієнтський інтерфейс** за допомогою **React**, який надає лікарям інтуїтивні інструменти для завантаження зображень, аналізу та, що найважливіше, візуалізації пояснень (Grad-CAM, LIME). Інтегровано **модуль ШІ** на **Python**, що відповідає за виконання нейромережевих обчислень та генерацію інтерпретацій.
2. **Впровадження безпеки.** Реалізовано надійні механізми **автентифікації та авторизації** на основі JWT та рольової моделі (RBAC), що чітко розмежовує доступ до функціональності системи.
3. Забезпечено комплексний **захист даних** шляхом шифрування трафіку (TLS 1.3), шифрування сховищ даних та застосування найкращих практик для протидії поширеним вебатакам.
4. **Забезпечення якості.** Застосовано комплексну, багаторівневу **стратегію тестування**, що включала модульне, інтеграційне, системне тестування та

тестування безпеки. **Усі ключові тестові сценарії**, що імітують реальну роботу лікаря, **були успішно пройдені** під час системного та приймального тестування. Це підтверджує, що реалізація повністю відповідає функціональним вимогам.

5. Розгортання та підтримка. Систему було успішно розгорнуто в цільовому хмарному середовищі **AWS** з використанням технологій контейнеризації (**Docker**) та оркестрації (**Kubernetes**). Налаштовано автоматизований **CI/CD процес** за допомогою GitHub Actions, що дозволяє швидко та надійно доставляти оновлення. Впроваджено інструменти моніторингу (**Prometheus, Grafana**), централізованого логування (**ELK Stack**) та резервного копіювання, що гарантує високий рівень експлуатаційної готовності та стабільності системи.

Загальна оцінка. На даному етапі інформаційна система є повністю реалізованим, протестованим та розгорнутим програмним продуктом. Ступінь готовності системи оцінюється як високий: вона відповідає всім проєктним вимогам, є функціонально повною та готовою до проведення пілотних випробувань у реальних умовах.

РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

Технічна доцільність. Технічна доцільність аналізує можливість реалізації проєкту з використанням існуючих технологій та ресурсів.

Технологічний стек — як зазначено в розділі 2.1.3, проєкт базується на сучасному та добре апробованому стеку технологій.

Frontend — react; Backend — Scala/ZIO; ШІ-модуль — pyTorch, TensorFlow; База даних — postgresSQL; Розгортання — docker/Kubernetes.

Всі ці інструменти є відкритими, мають велику спільноту підтримки та доведену ефективність, що мінімізує технічні ризики.

Інтеграція — система спроектована для інтеграції з існуючими медичними стандартами, такими як **DICOM** та **HL7/FHIR**, через спеціалізовані адаптери. Це забезпечує сумісність з наявною інфраструктурою медичних закладів.

Команда — для реалізації проєкту потрібна команда з компетенціями в Backend- та Frontend-розробці, Data Science (ШІ) та DevOps. З огляду на популярність обраних технологій, пошук кваліфікованих фахівців не становить значної проблеми.

Висновок — проєкт є технічно доцільним. Технології доступні, перевірені, а архітектурні рішення (мікросервіси) дозволяють гнучко керувати розробкою.

Економічна доцільність

Цей аспект оцінює фінансову життєздатність проєкту, порівнюючи витрати на розробку та впровадження з потенційними вигодами.

Оцінка витрат

Витрати поділяються на одноразові (Dev) та операційні (підтримка).

1. Витрати на розробку (приблизна оцінка на 6 місяців).

Команда (4 фахівці):

Backend Developer: \$2,500/mic.; Frontend Developer: \$2,000/mic.; AI/ML Engineer: \$3,000/mic.; Project Manager (part-time): \$1,500/mic.; Разом за 6 місяців: $(2,500 + 2,000 + 3,000 + 1,500) * 6 = \$54,000$

Інфраструктура (на час розробки):

Хмарні сервіси (для тестування та розгортання): \$300/mic.; Разом за 6 місяців: $300 * 6 = \$1,800$; Всього витрат на розробку: $54,000 + 1,800 = \$55,800$

2. Операційні витрати (щорічні):

Підтримка інфраструктури (хмарні сервери, GPU): \$6,000/рік.

Технічна підтримка та оновлення (20% від вартості розробки): \$11,160/рік.

Маркетинг та продажі: \$5,000/рік.

Всього операційних витрат на рік: $6,000 + 11,160 + 5,000 = \$22,160$

Оцінка вигод (щорічних)

Вигоди можуть бути прямими (дохід) та непрямыми (економія).

Прямі вигоди (модель SaaS для клінік):

Прогнозована кількість клієнтів (клінік) у перший рік: 10.

Середня вартість підписки: \$500/mic.

Річний дохід: $10 * 500 * 12 = \$60,000$

Непрямі вигоди (для клінік-користувачів):

Зменшення часу на діагностику — підвищення пропускнуої здатності лікарів на 15-20%.

Зниження кількості діагностичних помилок — покращення якості лікування та зменшення витрат на повторні обстеження.

Підвищення довіри до ІІІ-систем — дозволяє клінікам впроваджувати інноваційні методи діагностики.

Прогноз ROI

ROI розраховується за формулою:

$ROI = (\text{Чистий прибуток} / \text{Вартість інвестицій}) * 100\%$

Розрахуємо ROI за перші 3 роки.

Загальні інвестиції за 3 роки: $55,800(\text{розробка}) + 22,160 * 3 (\text{операційні витрати}) = \$122,280$

Прогнозований дохід за 3 роки (з урахуванням зростання клієнтської бази на 50% щорічно):

Рік 1: \$60,000; Рік 2: \$90,000; Рік 3: \$135,000; Всього доходу: \$285,000;

Чистий прибуток за 3 роки: $285,000 - 122,280 = \$162,720$

ROI за 3 роки: $(162,720 / 122,280) * 100\% \approx 133\%$

Висновок — проєкт є економічно доцільним. Прогнозований ROI у 133% за три роки свідчить про високу рентабельність, а точка беззбитковості очікується на другий рік експлуатації.

Юридична та правова доцільність

Цей аналіз визначає відповідність проєкту законодавчим нормам.

Захист даних — система спроектована з урахуванням міжнародних стандартів GDPR та HIPAA. Передбачена анонімізація даних пацієнтів та шифрування медичних записів.

Інтелектуальна власність — всі використані інструменти (React, Scala, Python, PyTorch) мають ліцензії, що дозволяють комерційне використання. Розроблений код буде власністю компанії-розробника.

Висновок — проєкт є юридично доцільним за умови дотримання всіх описаних у розділі 2.5.1 політик безпеки та конфіденційності.

Операційна доцільність (Operational Feasibility)

Оцінюється, наскільки легко система буде інтегрована в робочі процеси медичних установ.

Інтерфейс користувача — UI/UX розроблено з орієнтацією на лікарів, з мінімалістичним дизайном та інтуїтивною навігацією. Це знижує поріг входу та потребу в тривалому навчанні.

Інтеграція в робочий процес — система не замінює лікаря, а виступає як інструмент підтримки прийняття рішень. Вона легко інтегрується в існуючі процеси через експорт результатів у EHR.

Підтримка — мікросервісна архітектура спрощує оновлення та підтримку окремих компонентів системи без зупинки її роботи.

Висновок — проєкт є операційно доцільним. Він розроблений для безшовної інтеграції в клінічну практику.

Часова доцільність

Аналізується реалістичність термінів реалізації проєкту.

Тривалість розробки — орієнтовний термін у 6 місяців є реалістичним для команди з 4 осіб з огляду на чітко визначений функціонал та архітектуру в розділах 1 та 2.

Етапи проєкту — проєкт можна розбити на логічні етапи (розробка API, UI, ШІ-модуля, інтеграція), що дозволяє паралелізувати роботу та контролювати терміни.

Ризики — можливі затримки пов'язані з інтеграцією зі специфічними версіями PACS/EHR у клініках. Цей ризик можна мінімізувати шляхом попереднього аналізу інфраструктури клієнта.

Висновок — проєкт є доцільним у часових рамках. Заплановані терміни є реалістичними за умови ефективного управління проєктом.

Висновки до розділу 4

Проведений техніко-економічний аналіз підтверджує всебічну доцільність розробки інформаційної системи пояснюваності нейронних мереж у медичній сфері.

Технічно проєкт реалістичний, оскільки базується на доступних і надійних технологіях.

Економічно він є вигідним, з прогнозованим **ROI у 133%** за перші три роки, що свідчить про високий потенціал комерційного успіху.

Юридично, операційно та в часових рамках проєкт відповідає всім ключовим вимогам, що мінімізує ризики на етапах впровадження та експлуатації.

Таким чином, інвестиції в розробку даної системи є обґрунтованими та перспективними. Проєкт має потенціал не лише принести фінансову вигоду, але й зробити значний внесок у підвищення якості та прозорості медичної діагностики.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

5.1 Загальні положення про охорону праці

Охорона праці — це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності. Правовою основою в Україні є **Конституція України, Кодекс законів про працю (КЗпП) та Закон України «Про охорону праці»**.

Відповідно до законодавства, роботодавець зобов'язаний створити на кожному робочому місці безпечні та нешкідливі умови праці. Робота розробника програмного забезпечення не пов'язана зі шкідливими виробничими факторами, однак характеризується наявністю специфічних ризиків, що вимагають впровадження комплексних заходів для їх мінімізації.

Психофізіологічні: зорове та нервово-емоційне напруження, монотонність праці, тривале статичне навантаження.

Фізичні: підвищений рівень електромагнітного випромінювання, недостатня освітленість, некомфортний мікроклімат.

Мета цього розділу — визначити та обґрунтувати заходи, необхідні для створення безпечного та комфортного робочого середовища для розробника програмного забезпечення, що забезпечить високу продуктивність та збереже його здоров'я.

5.2 Заходи щодо забезпечення безпеки на робочому місці розробника програмного забезпечення

Для створення безпечних умов праці розробника необхідно дотримуватись вимог, що стосуються організації робочого місця, параметрів мікроклімату, освітлення, електро- та пожежної безпеки, а також режиму праці та відпочинку.

Вимоги до організації робочого місця та ергономіки

Правильна організація робочого простору є ключовим фактором у профілактиці захворювань опорно-рухового апарату та зорової системи.

Площа та об'єм — площа на одне робоче місце має становити не менше 6,0 м², а об'єм — не менше 20,0 м³, згідно з ДСанПіН 3.3.2.007-98.

Робочий стіл — повинен мати достатню площу для раціонального розміщення обладнання (монітор, клавіатура, миша) та документів. Рекомендована висота столу — 680-800 мм. Сучасною практикою є використання столів із можливістю регулювання висоти для роботи сидячи та стоячи.

Робоче крісло — має бути ергономічним, з підйомно-поворотним механізмом, регульованою висотою сидіння, кутом нахилу спинки та висотою підлокітників. Це дозволяє підтримувати оптимальну робочу позу та знижувати навантаження на хребет.

Розміщення монітора — екран монітора повинен знаходитись на відстані 60-70 см від очей, а його верхній край — на рівні очей або трохи нижче. Монітор слід розташовувати перпендикулярно до джерел світла, щоб уникнути відблисків.

Вимоги до мікроклімату та освітлення

Мікроклімат — у приміщенні повинні підтримуватися оптимальні параметри мікроклімату: температура повітря: 22-25 °С.; відносна вологість: 40-60%.; швидкість руху повітря: не більше 0,1 м/с.

Приміщення має бути обладнане системами вентиляції та кондиціонування для підтримки цих параметрів.

Освітлення — робоче місце повинно мати як природне, так і штучне освітлення. Освітленість на поверхні столу має бути в межах 300-500 лк. Необхідно унеможливити потрапляння прямих сонячних променів та відблисків на екран монітора.

Електробезпека — усе обладнання повинно бути підключене до мережі з заземлюючим контактом. Електричні кабелі та розетки мають бути без пошкоджень ізоляції. Забороняється самостійно ремонтувати обладнання.

Пожежна безпека — приміщення має бути обладнане первинними засобами пожежогасіння (вогнегасниками) та автоматичною пожежною

сигналізацією. Працівники повинні пройти інструктаж з пожежної безпеки та знати план евакуації.

Регламентовані перерви — відповідно до ДСанПіН 3.3.2.007-98, встановлюються внутрішньозмінні перерви тривалістю 10-15 хвилин кожні 2 години роботи. **Профілактичні вправи** — під час перерв рекомендується виконувати спеціальні вправи для очей (наприклад, правило «20-20-20»: кожні 20 хвилин дивіться на об'єкт на відстані 20 футів (6 метрів) протягом 20 секунд) та фізичні вправи для зняття статичного напруження з м'язів спини, шиї та рук. **Психофізіологічне розвантаження** — важливо забезпечити психологічний комфорт на робочому місці: уникати надмірного шуму, забезпечити чітку постановку завдань та мінімізувати стресові фактори для запобігання професійному вигоранню.

Висновки до розділу 5

Аналіз умов праці розробника програмного забезпечення показав, що, попри відсутність класичних шкідливих виробничих факторів, існує комплекс психофізіологічних та фізичних ризиків, що можуть негативно впливати на здоров'я та працездатність. Реалізація вищезазначених заходів, що включають ергономічну організацію робочого місця, підтримання оптимальних параметрів мікроклімату та освітлення, дотримання норм електро- та пожежної безпеки, а також впровадження раціонального режиму праці та відпочинку, є обов'язковою вимогою згідно із законодавством України.

Створення безпечного, здорового та психологічно комфортного робочого середовища є не лише виконанням нормативних вимог, а й ключовою інвестицією в добробут співробітника, що напряду впливає на його продуктивність, якість роботи та успішну реалізацію проєкту в цілому.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Вступ

Розвиток цифрових технологій та штучного інтелекту є рушійною силою сучасної економіки, однак він також формує нові виклики для екологічної безпеки. На відміну від традиційних промислових галузей, вплив ІТ-сектору на довкілля є переважно непрямим, але від того не менш значущим. Він проявляється через масивне споживання енергетичних ресурсів дата-центрами, утворення специфічних електронних відходів (e-waste) та використання природних ресурсів для виробництва апаратного забезпечення.

Метою даного розділу є проведення комплексного аналізу потенційного впливу розробки та експлуатації інформаційної системи пояснюваності нейронних мереж на навколишнє середовище. На основі цього аналізу будуть запропоновані конкретні заходи щодо мінімізації негативного впливу, що відповідають положенням Закону України «**Про охорону навколишнього природного середовища**» та міжнародним стандартам сталого розвитку.

Аналіз потенційного впливу системи на довкілля

Вплив ІТ-проєкту на навколишнє середовище доцільно розглядати в контексті його життєвого циклу, що складається з етапів розробки та експлуатації.

6.1 Вплив на етапі розробки

На цьому етапі вплив є відносно помірним і локалізованим.

Споживання електроенергії — основне джерело — це робота комп'ютерної техніки (робочих станцій, тестових серверів) команди розробників, а також забезпечення функціонування офісних приміщень (освітлення, опалення, вентиляція, кондиціонування).

Утворення відходів — генеруються переважно побутові та офісні відходи (папір, пакування, пластик). Можливе утворення невеликої кількості електронних відходів у разі виходу з ладу або модернізації обладнання.

6.2 Вплив на етапі експлуатації

Етап експлуатації системи є основним джерелом довгострокового та значного впливу на довкілля.

Енергоспоживання дата-центрами — це **ключовий фактор** впливу. Розгорнута система, особливо її ШІ-модуль, буде функціонувати на серверах у дата-центрі, що потребує безперервного живлення 24/7. Енергія витрачається на:

Обчислювальні процеси — робота центральних (CPU) та графічних (GPU) процесорів під час виконання ШІ-моделей (інференс) та їх періодичного донавчання є надзвичайно енергоємною.

Системи охолодження — значна частина спожитої серверами енергії перетворюється на тепло. Для підтримки оптимальної робочої температури використовуються потужні системи кондиціонування та вентиляції (HVAC), які можуть споживати до 40% від загальної енергії дата-центру.

Допоміжна інфраструктура — системи безперебійного живлення (UPS), мережеве обладнання та освітлення також роблять свій внесок.

Утворення електронних відходів (E-waste) — серверне та мережеве обладнання має обмежений термін служби (зазвичай 3-5 років). Після виведення з експлуатації воно перетворюється на електронні відходи. Неправильна утилізація e-waste є вкрай небезпечною, оскільки компоненти містять токсичні речовини (свинець, ртуть, кадмій, бромовані антипірени), які можуть забруднювати ґрунт та підземні води.

Використання природних ресурсів — виробництво серверів та комп'ютерів потребує видобутку рідкісноземельних металів, а також значних витрат води та енергії, що створює додаткове навантаження на природні екосистеми.

Заходи щодо мінімізації впливу на довкілля

Для забезпечення екологічної стійкості проєкту пропонується комплексний підхід, що охоплює оптимізацію програмного забезпечення, відповідальний вибір інфраструктури та впровадження організаційних "зелених" практик.

6.3 Стратегії енергоефективності ("Green IT")

Оптимізація програмного коду ("Green Coding") — найбільш ефективний спосіб знизити енергоспоживання — це зменшити потребу в обчислювальних ресурсах.

Ефективність алгоритмів — слід провести рефакторинг коду ШІ-модуля з метою оптимізації математичних операцій, зменшення кількості шарів нейронної мережі без втрати точності та застосування технік квантизації моделей.

Оптимізація запитів до БД — неефективні запити до бази даних можуть створювати надлишкове навантаження на сервер. Індексція, кешування та оптимізація логіки запитів дозволять значно знизити енергоспоживання.

Вибір "зеленого" хмарного провайдера — рекомендується розгортати систему в дата-центрах великих хмарних провайдерів (наприклад, Google Cloud, AWS, Microsoft Azure), які:

Використовують відновлювані джерела енергії — багато з них декларують 100% покриття своїх потреб за рахунок сонячної та вітрової енергії.

Мають високоефективні дата-центри — їхній показник **PUE (Power Usage Effectiveness)** наближається до 1.1, що означає мінімальні витрати енергії на охолодження.

Ефективне використання інфраструктури:

Контейнеризація та оркестрація — запроектоване використання **Docker** та **Kubernetes** є передовою екологічною практикою. Воно дозволяє максимально ефективно ущільнювати навантаження на фізичних серверах, що зменшує кількість необхідного обладнання та, відповідно, споживання енергії.

Автоматичне масштабування — налаштування **Horizontal Pod Autoscaler (HPA)** в Kubernetes дозволить динамічно виділяти ресурси лише тоді, коли вони потрібні, уникаючи простоювання та марного споживання енергії в періоди низького навантаження.

6.4 Управління електронними відходами

Відповідальне поводження з відходами є вимогою **Закону України «Про управління відходами»**.

Продовження життєвого циклу обладнання — замість повної заміни серверів слід розглядати можливість їх модернізації (збільшення оперативної пам'яті, заміна дисків на SSD).

Відповідальна утилізація — після закінчення терміну експлуатації все електронне обладнання повинно передаватися **виключно ліцензованим підприємствам** з утилізації електронних відходів. Це гарантує безпечне вилучення небезпечних компонентів та максимальну переробку матеріалів (пластику, металів).

6.5 Організаційні заходи

Політика "Зеленого офісу". Максимальне використання електронного документообігу для зменшення споживання паперу. Налаштування режимів енергозбереження на всіх комп'ютерах та офісній техніці. Впровадження сортування відходів в офісі (папір, пластик, батарейки).

Підтримка віддаленої роботи — гнучкий або повністю віддалений формат роботи для команди розробників суттєво зменшує викиди CO₂ від щоденних поїздок на роботу.

Висновки до розділу 6

Аналіз показав, що основними факторами впливу розробленої інформаційної системи на навколишнє середовище є значне споживання електроенергії на етапі експлуатації та утворення електронних відходів протягом життєвого циклу апаратного забезпечення.

Для мінімізації цього впливу було запропоновано комплексний набір заходів, що відповідають сучасним практикам сталого розвитку та законодавству України. Ключовими серед них є: оптимізація програмного коду для зниження навантаження, використання енергоефективних "зелених" дата-центрів, ефективне управління ресурсами за допомогою технологій контейнеризації, а також відповідальне поводження з електронними відходами.

Реалізація цих заходів дозволить не лише суттєво зменшити екологічний слід проєкту, але й часто призводить до економічної економії за рахунок зниження витрат на електроенергію. Це демонструє, що сучасна технологічна розробка може і повинна бути екологічно відповідальною, поєднуючи інновації з турботою про майбутнє довкілля.

ВИСНОВКИ

У ході виконання дипломної роботи було успішно досягнуто поставленої мети, а саме: проведено оцінку ефективності існуючих методів пояснюваності нейронних мереж та розроблено концептуальні засади для їх практичного застосування у медичній сфері з метою підвищення довіри, прозорості та прийнятності рішень штучного інтелекту серед фахівців.

Для досягнення цієї мети було вирішено такі ключові завдання:

1. **Проведено глибокий аналіз предметної області**, який підтвердив, що фундаментальна проблема «чорного ящика» є критичним бар'єром для широкого впровадження систем штучного інтелекту в клінічну практику. Було систематизовано та досліджено основні підходи до пояснюваності (XAI), зокрема LIME, SHAP, Grad-CAM, DeepLIFT та LRP, виявлено їхні сильні сторони та суттєві обмеження, такі як обчислювальна складність, локальний характер пояснень та залежність від типів даних у медичному контексті.

2. **Розроблено комплексну архітектуру інформаційної системи**, що є практичним рішенням для впровадження інтерпретованого ШІ у сферу охорони здоров'я. Запропонована мікросервісна архітектура, сучасний технологічний стек та контейнеризація забезпечують гнучкість, надійність та масштабованість системи. Ключовим елементом проєкту є інтеграція з загальноприйнятими медичними стандартами (DICOM, HL7/FHIR) та фокус на візуалізації пояснень (Grad-CAM для зображень та LIME для табличних даних), що робить систему адаптованою до реальних робочих процесів лікарів.

3. **Сформульовано конкретні рекомендації щодо підвищення ефективності та безпеки** використання ШІ в медицині. Ці рекомендації втілені у проєктних рішеннях системи: від детальних специфікацій API та моделей даних до багаторівневої системи безпеки, що відповідає вимогам GDPR та HIPAA5. Проведене техніко-економічне обґрунтування та аналіз аспектів охорони праці й довкілля довели життєздатність та відповідність запропонованого рішення сучасним стандартам.

Таким чином, у роботі здійснено перехід від теоретичного аналізу проблеми пояснюваності до створення цілісного, архітектурно обґрунтованого та економічно доцільного інженерного рішення. Спроектована система створює міцну основу для розробки практичного інструменту, здатного підвищити прозорість, довіру та ефективність використання нейронних мереж у медичній діагностиці, що повністю відповідає меті дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Івченко, І., Гайдаєнко, О., Книрик, Н., Морозова, Г., & Грибенюк, Д. (2024). Прогнозування поведінки об'єктів соціальних мереж. *Herald of Khmelnytskyi National University. Technical Sciences*, 341(5), 317-321.
- [2] Adadi, A., & Berrada, M. (2022). Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6, 52138–52160. <https://doi.org/10.1109/ACCESS.2018.2870052>
- [3] Amann, J., Blasimme, A., Vayena, E., Frey, D., & Madai, V. I. (2023). Explainability for artificial intelligence in healthcare: A multidisciplinary perspective. *BMC Medical Informatics and Decision Making*, 20(1), 310. <https://doi.org/10.1186/s12911-020-01332-6>
- [4] Verbytskyi, O. S., Haidaienko, O. V. (2025). Automated rule generation for optimal data selection in robust neural network training. *Zbirnyk naukovykh prats NUK*, 3(501). [https://doi.org/10.15589/znp2025.3\(501\).20](https://doi.org/10.15589/znp2025.3(501).20)
- [5] Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., ... & Weller, A. (2024). Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. *Proceedings of the 2024 AAAI/ACM Conference on AI, Ethics, and Society*, 401–413.
- [6] Chaddad, A., Peng, J., Xu, J., & Bouridane, A. (2023). Survey of explainable AI techniques in healthcare. *Sensors*, 23(2), 634. <https://doi.org/10.3390/s23020634>
- [7] Ghassemi, M., Oakden-Rayner, L., & Beam, A. L. (2022). The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3(11), e745–e750. [https://doi.org/10.1016/S2589-7500\(21\)00208-9](https://doi.org/10.1016/S2589-7500(21)00208-9)
- [8] Hagras, H. (2023). Toward human-understandable, explainable AI. *Computer*, 51(12), 28–36. <https://doi.org/10.1109/MC.2018.2885945>
- [9] Health Level Seven International. (2024). *HL7 FHIR Release 5: Documentation*. HL7 International. <http://hl7.org/fhir>

- [10] Holzinger, A., Langs, G., Denk, H., Zatloukal, K., & Müller, H. (2022). Causability and explainability of artificial intelligence in medicine. *WIREs Data Mining and Knowledge Discovery*, 9(4), e1312. <https://doi.org/10.1002/widm.1312>
- [11] Lohr, D., & Chugunova, M. (2024). Explainable AI in Medical Imaging: A Review of Methods and Clinical Applications. *Journal of Digital Imaging*, 37, 12–28.
- [12] Lundberg, S. M., & Lee, S. I. (2022). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4765–4774).
- [13] Newman, S. (2023). *Building Microservices: Designing Fine-Grained Systems* (3rd ed.). O'Reilly Media.
- [14] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2022). Grad-CAM: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2), 336–359.
- [15] ZIO Community. (2024). *ZIO: Type-safe, composable asynchronous and concurrent programming for Scala*. ZIO Documentation. <https://zio.dev>

ДОДАТОК А

Технічне завдання на розробку сервісу

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної роботи

_____ (Прізвище І.П.)

«_» _____ 2025р.

ТЕХНІЧНЕ ЗАВДАННЯ

на створення інформаційної системи пояснюваності нейронних мереж у медичній сфері

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1. Повне найменування системи

Інформаційна система пояснюваності нейронних мереж у медичній сфері (далі – Система).

1.2. Призначення системи

Система призначена для автоматизації процесу медичної діагностики з використанням методів штучного інтелекту (ШІ) та забезпечення інтерпретації отриманих результатів. Основна мета — розв'язання проблеми «чорної скриньки» шляхом візуалізації причин прийняття рішень нейронними мережами для підвищення довіри медичного персоналу.

2. ЦІЛІ ТА ЗАВДАННЯ РОЗРОБКИ

2.1. Основні цілі

Підвищення прозорості діагностичних рішень ШІ; зменшення ризику лікарських помилок шляхом надання пояснень (візуальних та текстових); інтеграція сучасних методів XAI (Explainable AI) у клінічний робочий процес.

2.2. Завдання системи

Завантаження та збереження медичних зображень (DICOM/NIfTI).

Класифікація патологій за допомогою нейронних мереж.

Генерація пояснень результатів методами Grad-CAM (для зображень) та LIME (для табличних даних/ознак).

Інтеграція із зовнішніми медичними системами (PACS, EHR).

3. ВИМОГИ ДО СИСТЕМИ

3.1. Вимоги до функціональних характеристик

Система повинна реалізовувати наступні функції відповідно до рольової моделі:

3.1.1. Підсистема роботи з даними (Лікар, PACS)

Завантаження медичних зображень через веб-інтерфейс або отримання їх з PACS-сервера за протоколом DICOM (C-MOVE/C-STORE).

Перевірка валідності форматів файлів (.dcm, .nii, .nii.gz).

Анонімізація персональних даних пацієнта перед збереженням у базі даних.

3.1.2. Підсистема аналізу та пояснюваності (Модуль ШІ)

Виконання класифікації зображень за допомогою попередньо навчених моделей (TensorFlow/PyTorch).

Генерація теплових карт (heatmap) методом **Grad-CAM** для візуалізації областей інтересу на зображенні.

Розрахунок ваги ознак методом **LIME** для пояснення впливу окремих факторів на діагноз.

Збереження результатів аналізу та метаданих пояснень у базі даних.

3.1.3. Підсистема візуалізації та взаємодії (Frontend)

Відображення оригінального медичного зображення з можливістю регулювання контрасту та масштабування.

Накладання шару пояснення (heatmap) на зображення з можливістю налаштування прозорості.

Відображення текстового звіту: діагноз, рівень впевненості моделі (Confidence Score), ключові ознаки.

Функція порівняння різних методів пояснення (Side-by-side view).

3.1.4. Підсистема інтеграції та експорту

Експорт результатів діагностики у зовнішні системи EHR (Electronic Health Record) з використанням стандарту HL7 FHIR (ресурси Observation, DiagnosticReport).

Формування друкованих звітів у форматі PDF.

3.2. Вимоги до нефункціональних характеристик

3.2.1. Вимоги до надійності

Забезпечення доступності системи на рівні не нижче 99.9%.

Автоматичне перемикавання на резервні вузли (Failover) при збої основного модуля ІІІ (засобами Kubernetes).

Регулярне резервне копіювання бази даних (щоденні снєпшоти).

3.2.2. Вимоги до безпеки

Автентифікація користувачів через протокол OAuth2 (JWT-токени).

Шифрування даних у стані спокою (AES-256 для БД) та при передачі (TLS 1.3).

Розмежування прав доступу (RBAC): ролі «Лікар» та «Адміністратор».

Логування всіх дій користувачів для аудиту (відповідність HIPAA/GDPR).

3.2.3. Вимоги до продуктивності

Час відгуку API на запити отримання даних ≤ 200 мс.

Підтримка асинхронної обробки «важких» запитів (аналіз 3D-зображень) через черги повідомлень.

4. ВИМОГИ ДО АРХІТЕКТУРИ ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Архітектурний стиль

Мікросервісна архітектура з використанням контейнеризації.

4.2. Технологічний стек

Frontend: React, бібліотека DICOM.js для візуалізації.

Backend: Scala 3 (фреймворк ZIO-HTTP) для високопродуктивної асинхронної обробки.

AI Module: Python, фреймворки PyTorch/TensorFlow, бібліотеки Captum (Grad-CAM), SHAP/LIME.

База даних: PostgreSQL з розширенням PostGIS (геопросторові дані), Redis (кешування).

Інфраструктура: Docker, Kubernetes, Nginx.

4.3. Інтеграційні інтерфейси

RESTful API (OpenAPI 3.0 specification); DICOM Protocol (TCP/IP port 104); HL7 FHIR (REST HTTPS).

5. ЕТАПИ РОЗРОБКИ

1. **Аналіз вимог та проєктування:** розробка архітектури, проєктування БД, UI/UX прототипування.
 2. **Розробка серверної частини (Backend):** реалізація API, інтеграція з БД та брокерами повідомлень.
 3. **Розробка модуля ШІ:** налаштування моделей, реалізація алгоритмів пояснюваності (Grad-CAM, LIME).
 4. **Розробка клієнтської частини (Frontend):** верстка інтерфейсу, інтеграція з API та візуалізація DICOM.
 5. **Тестування:** модульне, інтеграційне, тестування безпеки та навантажувальне тестування.
 6. **Розгортання:** налаштування CI/CD, деплой у цільове середовище.

6. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Здача роботи здійснюється шляхом демонстрації працездатного прототипу системи, який виконує завантаження тестового набору медичних даних, проводить їх класифікацію та відображає коректні пояснення рішень нейронної мережі.

Розробник:

Студент магістратури _____ (ПІБ)

ДОДАТОК Б

ІНСТРУКЦІЯ КОРИСТУВАЧА

1. Вступ

Дана інструкція призначена для медичного персоналу (лікарів-діагностів, рентгенологів), які використовують Інформаційну систему пояснюваності нейронних мереж у медичній сфері (далі — Система). Документ описує порядок роботи з веб-інтерфейсом системи, включаючи авторизацію, завантаження медичних зображень (DICOM/NIfTI), інтерпретацію результатів роботи штучного інтелекту (ШІ) за допомогою методів Grad-CAM та LIME, а також експорт звітів до зовнішніх систем (EHR/PACS).

2. Авторизація в системі

Для початку роботи необхідно пройти процедуру автентифікації. Система підтримує рольову модель доступу для забезпечення безпеки медичних даних.

5. Відкрийте веб-браузер та перейдіть за адресою сервера Системи.
6. На екрані входу (Рисунок Б.1) введіть свої облікові дані (Medical ID та пароль).
7. Натисніть кнопку **"Login"**.

XAI-MED System

Secure Access for Medical Professionals

Medical ID / Username

Dr. Gregory House

Password

Login

Authorized use only. All activities are monitored.

Рисунок Б.1 — Екран авторизації користувача

3. Головна панель та завантаження досліджень

Після успішного входу відкривається головна панель (Dashboard). Тут доступний список останніх пацієнтів та функціонал для завантаження нових даних.

8. Для завантаження нового дослідження натисніть кнопку **"Upload New Scan"** у верхній частині екрана.
9. У діалоговому вікні оберіть файл формату .dcm (DICOM) або .nii (NIfTI) з локального диска або імпортуйте з PACS.
10. Для перегляду існуючого дослідження оберіть пацієнта зі списку **"Recent Studies"** (Рисунок Б.2).

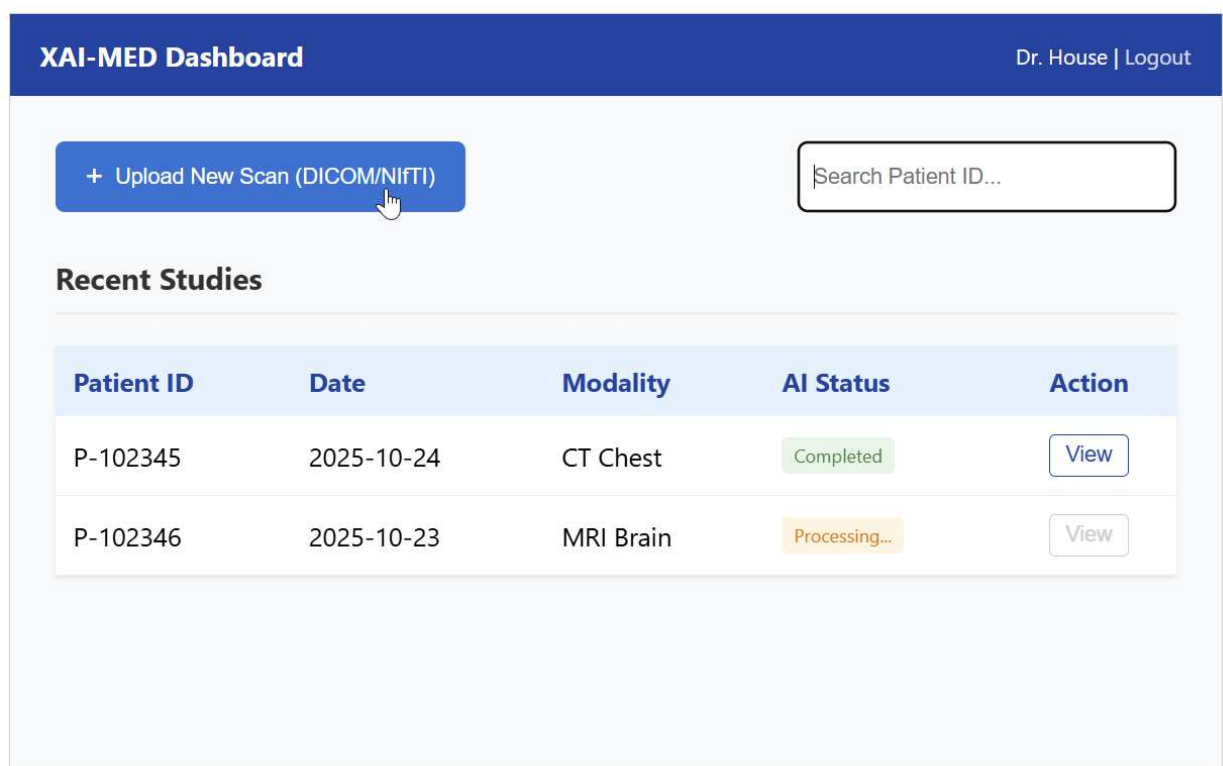


Рисунок Б.2 — Головна панель (Dashboard)

4. Робота з результатами аналізу (Режим пояснюваності)

Це основний робочий екран лікаря. Після обробки зображення нейронною мережею екран ділиться на дві функціональні зони.

- **Ліва панель (Візуалізація)** - відображає медичний знімок. Користувач може активувати перемикач **"Show Heatmap (Grad-CAM)"**, щоб

побачити теплову карту. Червоні зони вказують на області, які найбільше вплинули на рішення моделі.

- **Права панель (Діагноз та LIME)** - показує прогнозований клас (наприклад, "Pneumonia"), рівень впевненості моделі (%) та список ключових ознак (LIME), що вплинули на результат (Рисунок Б.3).

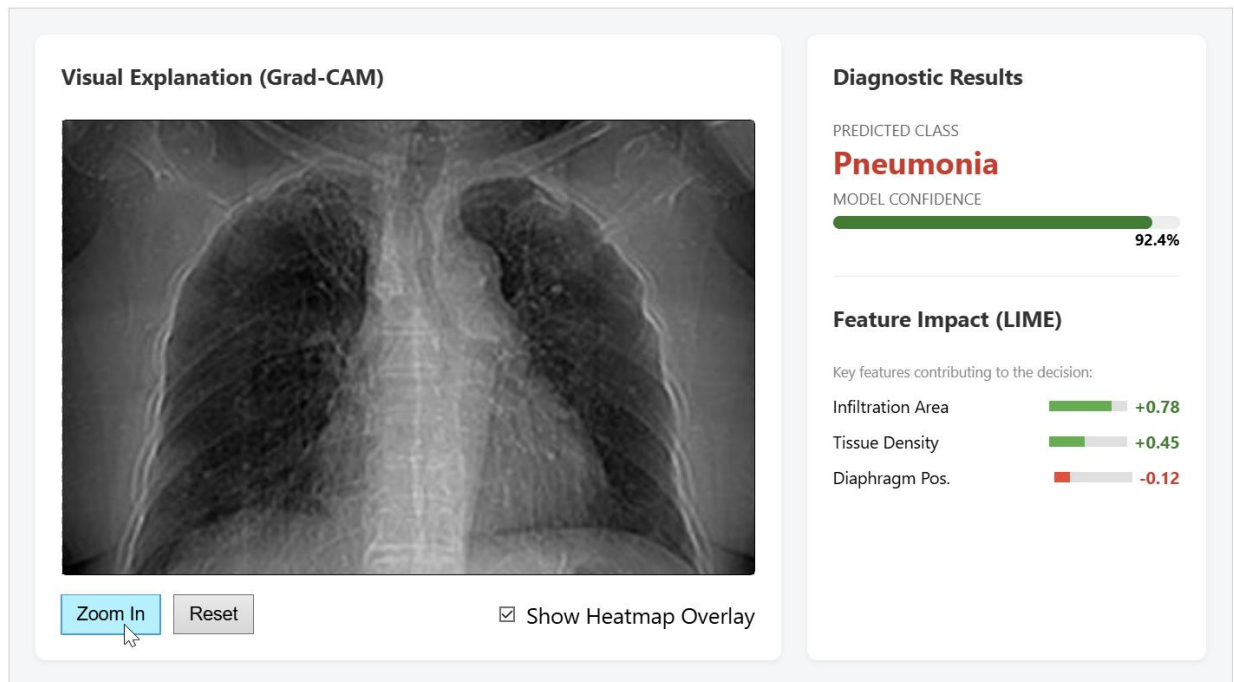


Рисунок Б.3 — Інтерфейс аналізу з методами Grad-CAM та LIME

5. Порівняння методів пояснюваності

Для верифікації складних випадків система дозволяє порівнювати різні методи інтерпретації (наприклад, Grad-CAM та оригінальне зображення або інший метод) пліч-о-пліч. Це дозволяє лікарю впевнитися у коректності локалізації патології.

- 11.Перейдіть на вкладку "**Comparison**" у верхньому меню.
- 12.Виберіть тип візуалізації для лівої (Method A) та правої (Method B) панелі.
- 13.Використовуйте повзунок прозорості (Opacity Slider) для налаштування інтенсивності накладання шарів (Рисунок Б.4).

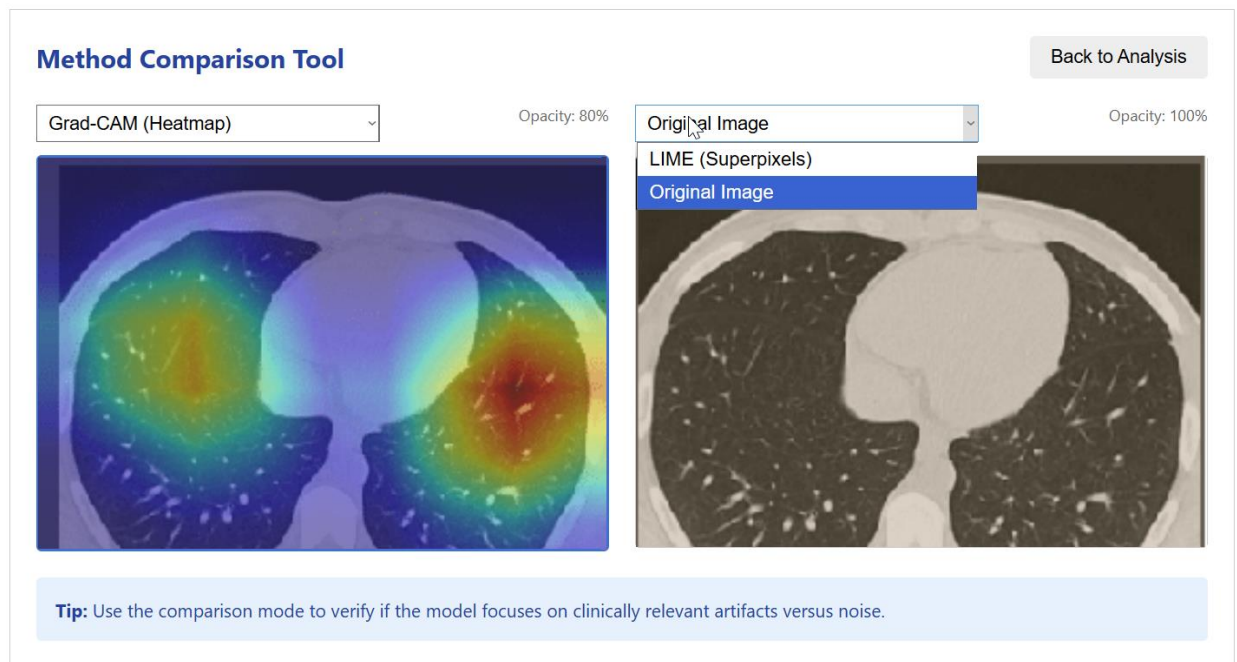


Рисунок Б.4 — Режим порівняльного аналізу

6. Експорт звіту до EHR

Завершальним етапом роботи є збереження результатів. Система дозволяє експортувати дані у форматі, сумісному із зовнішніми електронними медичними картками (EHR) через протокол FHIR.

14. Натисніть кнопку **"Export to EHR"**.

15. У модальному вікні перевірте дані, що будуть відправлені (Діагноз, Обрані зображення з поясненнями, Коментар лікаря).

16. Оберіть формат (PDF звіт або FHIR JSON) та натисніть **"Confirm Export"** (Рисунок Б.5).

Export Results to EHR ×

Target System

Hospital Main EHR (Epic/Cerner) ▼

Include in Report

☒ Diagnostic Result ☒ Grad-CAM Heatmap ☐ Raw DICOM

Doctor's Notes

Add clinical context here...

✓ Patient ID (P-102345) verified against HL7 server.

Cancel **Confirm Export**

Рисунок Б.5 — Вікно експорту результатів

7. ЗАВЕРШЕННЯ РОБОТИ

Для виходу із системи натисніть на іконку профілю користувача у верхньому меню та оберіть пункт "Log Out".

Технічна підтримка

У разі виникнення питань щодо функціональності Системи, технічних збоїв або потреби в консультації, будь ласка, звертайтеся до Служби технічної підтримки за електронною адресою: support@xai-med.university.edu або за номером телефону: (012) 123-45-67.

ДОДАТОК В

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Об'єкт Випробувань

Об'єктом випробувань є Інформаційна система пояснюваності нейронних мереж у медичній сфері (далі — Система).

Система складається з наступних компонентів, що підлягають перевірці:

- 17.**Веб-клієнт (Frontend):** інтерфейс користувача для взаємодії лікарів із системою.
- 18.**Серверна частина (Backend API):** логіка обробки запитів, маршрутизація та взаємодія з базою даних.
- 19.**Модуль ШІ (AI Module):** компоненти для класифікації зображень та генерації пояснень (Grad-CAM, LIME).
- 20.**База даних:** підсистема зберігання медичних записів та метаданих.

2. Мета Випробувань

Метою випробувань є перевірка відповідності Системи вимогам Технічного завдання, а саме:

- перевірка коректності виконання функціональних завдань (завантаження DICOM-файлів, класифікація, візуалізація пояснень);
- перевірка надійності захисту даних (автентифікація, розмежування прав доступу);
- оцінка продуктивності системи при обробці запитів;
- перевірка інтеграційних можливостей із зовнішніми системами (EHR, PACS).

3. Умови Та Порядок Проведення Випробувань

3.1. Вимоги до технічних засобів

Випробування проводяться на апаратному забезпеченні з наступними характеристиками:

- **Сервер:** CPU 4 cores, RAM 16 GB, SSD 500 GB, GPU (NVIDIA Tesla/RTX для модуля ШІ).
- **Клієнтська станція:** Персональний комп'ютер (CPU i5, RAM 8 GB), монітор з роздільною здатністю не менше 1920x1080.

3.2. Вимоги до програмних засобів

Для проведення випробувань необхідне наступне ПЗ:

- ОС сервера: Linux (Ubuntu 20.04/22.04).
- Середовище виконання: Docker, Kubernetes.
- СУБД: PostgreSQL 14 з розширенням PostGIS.
- Браузер клієнта: Google Chrome (v.100+), Firefox.
- Інструменти тестування: Postman (для API), Selenium (для UI), JMeter (навантажувальне тестування).

4. Методика Випробувань

Випробування проводяться шляхом виконання контрольних прикладів (тест-кейсів). Результати фіксуються у протоколі випробувань.

4.1. Перевірка підсистеми безпеки та авторизації

Мета: Перевірити обмеження доступу до медичних даних.

№	Назва перевірки	Опис дій	Очікуваний результат
1	Авторизація лікаря	Ввести коректний логін та пароль користувача з роллю "Doctor".	Система надає доступ до Головної панелі. Генерується JWT-токен.
2	Несанкціонований доступ	Спроба входу з невірним паролем.	Доступ заборонено, виводиться повідомлення про помилку.
3	Доступ до API без токена	Відправка GET-запиту на /api/images без заголовка Authorization.	Отримано відповідь 401 Unauthorized.

4.2. Перевірка функціоналу обробки медичних зображень

Мета: Перевірити здатність системи працювати з форматом DICOM та модулем ШІ.

№	Назва перевірки	Опис дій	Очікуваний результат
4	Завантаження DICOM	Через інтерфейс завантажити файл формату .dcm (розмір до 50 МБ).	Файл успішно завантажено, метадані (ID пацієнта, дата) автоматично витягнуто та відображено у таблиці.
5	Валідація форматів	Спроба завантажити файл .jpg або .exe.	Система відхиляє файл з помилкою 400 Bad Request ("invalid_file_format").
6	Класифікація та Grad-CAM	Ініціювати аналіз зображення з вибором методу Grad-CAM.	Отримано діагноз з впевненістю >80%. На зображення накладено теплову карту (heatmap).
7	Генерація пояснень LIME	Ініціювати аналіз табличних даних або зображення методом LIME.	Відображено список ключових ознак із ваговими коефіцієнтами (позитивні/негативні впливи).

4.3. Перевірка інтеграції та експорту даних

Мета: Перевірити сумісність із зовнішніми системами.

№	Назва перевірки	Опис дій	Очікуваний результат
8	Експорт у FHIR	На сторінці результатів натиснути "Export to EHR", обрати формат FHIR.	Система формує JSON-об'єкт ресурсу Observation та повертає статус 200 OK.
9	Перегляд порівняння	Відкрити режим порівняння (Comparison Mode), обрати Grad-CAM та оригінал.	Екран розділено на дві частини, слайдер прозорості працює коректно.

4.4. Перевірка продуктивності та надійності

Мета: Перевірити відповідність нефункціональним вимогам.

№	Назва перевірки	Опис дій	Очікуваний результат
10	Час реакції системи	Виміряти час відповіді API при генерації пояснення (Grad-CAM).	Час обробки не перевищує 3-5 секунд (для складних моделей) або 200 мс для простих запитів.
11	Відмовостійкість	Примусово зупинити контейнер з основним модулем ІІІ.	Kubernetes автоматично перенаправляє запит на резервний под, або перезапускає контейнер. Система залишається доступною.

5. Оцінка Результатів

Система вважається такою, що пройшла випробування, якщо:

21. Успішно виконано 100% тест-кейсів функціональної перевірки.
22. Виявлені помилки мають статус "Low Priority" або "Cosmetic" і не впливають на критичний функціонал діагностики.
23. Час відгуку системи задовольняє вимогам продуктивності (середня затримка < 200 мс для інтерфейсу).
24. Забезпечено коректну інтеграцію з форматами DICOM та FHIR.

Результати фіксуються в "Протоколі випробувань", на підставі якого приймається рішення про допуск Системи до дослідної експлуатації.

ДОДАТОК Г

Фрагменти програмного коду

Г.1 Серверна частина (Backend API Service)

Мова: Scala 3

Технології: ZIO 2.x, ZIO-HTTP, Tapir

Лістинг Г.1.1 — Точка входу в застосунок та налаштування HTTP-сервера

```
import zio.*
import zio.http.*
import service.MedicalAnalysisService

object MainApp extends ZIOAppDefault:
  // Конфігурація серверу та маршрутизації
  val run =
    val routes = AnalysisRoutes() ++ HealthCheckRoutes()

    Server.serve(routes.withDefaultErrorResponse)
      .provide(
        Server.default,
        MedicalAnalysisService.live, // Dependency Injection сервісу аналізу
        DatabaseModule.live,       // Підключення до PostgreSQL
        Authentication.live        // Middleware для JWT
      )
```

Лістинг Г.1.2 — Визначення API ендпоінтів для аналізу зображень

```
import zio.*
import zio.http.*
import zio.json.*
import model.*

object AnalysisRoutes:
  def apply(): Http[MedicalAnalysisService, Throwable, Request, Response] =
    Http.collectZIO[Request] {

      // POST /api/analyze - Ініціалізація аналізу
      case req @ Method.POST -> Root / "api" / "analyze" =>
        for {
          body <- req.body.asString
          requestData <- ZIO.fromEither(body.fromJson[AnalysisRequest])
          _ <- mapError(_ => Response.badRequest("Invalid JSON format"))
          service <- ZIO.service[MedicalAnalysisService]
          result <- service.analyzeImage(requestData.imageId, requestData.method)
        } yield Response.json(result.toJson)

      // GET /api/results/:id - Отримання результату
```

```

case Method.GET -> Root / "api" / "results" / taskId =>
  ZIO.service[MedicalAnalysisService]
    .flatMap(_.getResult(taskId))
    .map(res => Response.json(res.toJson))
    .catchAll(_ => ZIO.succeed(Response.status(Status.NotFound)))
}

```

Лістинг Г.1.3 — Сервісний шар (бізнес-логіка)

```

class MedicalAnalysisServiceLive(aiClient: GrpcAiClient, db: DatabaseRepository)
extends MedicalAnalysisService:

override def analyzeImage(imageId: UUID, method: ExplanationMethod): Task[AnalysisResponse] =

for {
  // Отримання метаданих зображення з БД
  imageMeta <- db.getImageMetadata(imageId)

  // Асинхронний виклик AI-модуля через gRPC
  aiResult <- aiClient.predictAndExplain(
    imageUrl = imageMeta.storagePath,
    method = method.toString
  )

  // Збереження результатів
  _ <- db.saveResult(imageId, aiResult)
} yield AnalysisResponse(taskId = aiResult.executionId, status = "COMPLETED")

```

Г.2 Сервіс взаємодії з AI-модулем через gRPC

Мова: Scala 3

Технології: ZIO-gRPC, ScalaPB

Лістинг Г.2.1 — Proto-файл визначення сервісу

```

syntax = "proto3";

package xai.med;

service AIService {
  rpc Analyze (AnalyzeRequest) returns (AnalyzeResponse);
}

message AnalyzeRequest {
  string image_path = 1;
  string explanation_method = 2; // "GRAD_CAM" or "LIME"
}

message AnalyzeResponse {
  string diagnosis = 1;
  float confidence = 2;
}

```

```
bytes explanation_data = 3; // Heatmap або JSON з вагами
}
```

Лістинг Г.2.2 — Клієнтська реалізація на ZIO

```
import io.grpc.ManagedChannelBuilder
import scalapb.zio_grpc.ZManagedChannel
import xai.med.ai_service.ZioAiService.AIServiceClient

object AiModuleClient:
  // Створення безпечного каналу зв'язку
  val live: ZLayer[Any, Throwable, AIServiceClient] =
    AIServiceClient.live(
      ZManagedChannel(
        ManagedChannelBuilder.forAddress("ai-inference-service", 50051)
          .usePlaintext() // У продакшені використовувати TLS
      )
    )

  def predict(path: String, method: String): ZIO[AIServiceClient, Status, AnalyzeResponse] =
    AIServiceClient.analyze(AnalyzeRequest(path, method))
```

Г.3 Модуль штучного інтелекту (AI Inference Engine)

Мова: Python 3.10

Технології: PyTorch, Captum (для XAI), NumPy

Лістинг Г.3.1 — Завантаження моделі та препроцесинг

```
import torch
from torchvision import models, transforms

class ModelLoader:
    def __init__(self, model_path: str):
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        # Завантаження ResNet50, натренованої на медичних даних
        self.model = models.resnet50(pretrained=False)
        self.model.load_state_dict(torch.load(model_path, map_location=self.device))
        self.model.eval()

    def preprocess(self, image_bytes):
        transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
        ])
        return transform(image_bytes).unsqueeze(0).to(self.device)
```

Лістинг Г.3.2 — Генерація пояснень (Grad-CAM)

```
from captum.attr import LayerGradCam, LayerAttribution
import numpy as np
```



```

class ExplainabilityEngine:
    def __init__(self, model, target_layer):
        self.grad_cam = LayerGradCam(model, target_layer)

    def generate_heatmap(self, input_tensor, target_class_index):
        # Генерація атрибуцій
        attributions = self.grad_cam.attribute(input_tensor, target=target_class_index)

        # Інтерполяція до розміру оригінального зображення
        heatmap = LayerAttribution.interpolate(attributions, (224, 224))

        # Нормалізація та конвертація в bytes
        heatmap_np = heatmap.squeeze().cpu().detach().numpy()
        return self._serialize_heatmap(heatmap_np)

    def _serialize_heatmap(self, heatmap_np):
        # Конвертація масиву у формат для передачі через gRPC
        import io
        buffer = io.BytesIO()
        np.save(buffer, heatmap_np)
        return buffer.getvalue()

```

Г.4 Обробник запитів gRPC (Python Server)

Мова: Python

Технології: grpcio, concurrent.futures

Лістинг Г.4.1 — Реалізація gRPC Сервера

```

import grpc
from concurrent import futures
import ai_service_pb2_grpc
from ai_processor import MedicalAIProcessor

class AIServiceServicer(ai_service_pb2_grpc.AIServiceServicer):
    def __init__(self):
        self.processor = MedicalAIProcessor()

    def Analyze(self, request, context):
        try:
            print(f"Processing request for: {request.image_path}")

            # Виклик логіки інференсу
            diagnosis, confidence, explanation = self.processor.process(
                request.image_path,
                request.explanation_method
            )

            return ai_service_pb2.AnalyzeResponse(

```

```

        diagnosis=diagnosis,
        confidence=confidence,
        explanation_data=explanation
    )
    except Exception as e:
        context.set_details(f"Internal Error: {str(e)}")
        context.set_code(grpc.StatusCode.INTERNAL)
        return ai_service_pb2.AnalyzeResponse()

def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    ai_service_pb2_grpc.add_AIServiceServicer_to_server(AIServiceServicer(), server)
    server.add_insecure_port(':::50051')
    server.start()
    server.wait_for_termination()

```

Г.5 Клієнтська частина (Frontend Application)

Мова: JavaScript (ES6+)

Технології: React, Axios, DICOM.js

Лістинг Г.5.1 — Компонент завантаження зображень (React Hook)

```

import React, { useState } from 'react';
import api from '../services/api';

const ImageUploader = ({ onUploadSuccess }) => {
    const [loading, setLoading] = useState(false);

    const handleFileUpload = async (event) => {
        const file = event.target.files[0];
        if (!file) return;

        const formData = new FormData();
        formData.append('file', file);

        setLoading(true);
        try {
            const response = await api.post('/images', formData, {
                headers: { 'Content-Type': 'multipart/form-data' }
            });
            onUploadSuccess(response.data.image_id);
        } catch (error) {
            console.error("Upload failed", error);
            alert("Помилка завантаження файлу");
        } finally {
            setLoading(false);
        }
    };
};

```

```

return (
  <div className="upload-container">
    <input type="file" accept=".dcm,.nii" onChange={handleFileUpload} />
    {loading && <p>Завантаження та обробка...</p>}
  </div>
);
};

```

Лістинг Г.5.2 — Відображення результатів з перемикачем Heatmap

```

import React, { useState } from 'react';

const ResultViewer = ({ originalImage, heatmapImage, diagnosis, confidence }) => {
  const [showHeatmap, setShowHeatmap] = useState(true);

  return (
    <div className="result-viewer">
      <h3>Діагноз: {diagnosis} (Впевненість: {(confidence * 100).toFixed(1)}%)</h3>

      <div className="image-wrapper" style={{ position: 'relative' }}>
        /* Оригінальне зображення */
        <img src={originalImage} alt="Scan" className="base-image" />

        /* Накладання Grad-CAM (Heatmap) */
        {showHeatmap && (
          <img
            src={`data:image/png;base64,${heatmapImage}`}
            alt="Explanation"
            className="heatmap-overlay"
            style={{ position: 'absolute', top: 0, left: 0, opacity: 0.6 }}
          />
        )}
      </div>

      <div className="controls">
        <label>
          <input
            type="checkbox"
            checked={showHeatmap}
            onChange={() => setShowHeatmap(!showHeatmap)}
          />
          Показати пояснення (Grad-CAM)
        </label>
      </div>
    </div>
  );
};

```

Г.6 База даних (Database Schema)

СУБД: PostgreSQL 14

Технології: SQL, PL/pgSQL, GIN Indexing

Лістинг Г.6.1 — DDL створення таблиць

-- Таблиця метаданих медичних зображень

```
CREATE TABLE MedicalImages (  
  image_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  patient_id UUID NOT NULL, -- FK constraints опущені для стислості  
  storage_path VARCHAR(255) NOT NULL,  
  modality VARCHAR(10) CHECK (modality IN ('CT', 'MRI', 'XRAY')),  
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()  
);
```

-- Таблиця результатів класифікації

```
CREATE TABLE ClassificationResults (  
  result_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  image_id UUID REFERENCES MedicalImages(image_id) ON DELETE CASCADE,  
  diagnosis VARCHAR(100) NOT NULL,  
  confidence FLOAT NOT NULL CHECK (confidence >= 0 AND confidence <= 1),  
  processing_time_ms INTEGER  
);
```

-- Таблиця пояснень (з використанням JSONB для гнучкості)

```
CREATE TABLE Explanations (  
  explanation_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  result_id UUID REFERENCES ClassificationResults(result_id),  
  method_type VARCHAR(20) NOT NULL, -- 'Grad-CAM', 'LIME'  
  
  -- Для Grad-CAM: зберігаємо шлях до heatmap або base64  
  heatmap_data TEXT,  
  -- Для LIME: зберігаємо ваги ознак у JSON  
  feature_weights JSONB,  
  created_at TIMESTAMP DEFAULT NOW());
```

Лістинг Г.6.2 — Індeksi для оптимізації продуктивності

-- Індекс для швидкого пошуку досліджень пацієнта

```
CREATE INDEX idx_images_patient ON MedicalImages(patient_id);
```

-- GIN-індекс для пошуку по вмісту JSONB (наприклад, пошук LIME ваг > 0.5)

```
CREATE INDEX idx_lime_weights ON Explanations USING GIN (feature_weights);
```

-- Партиціонування результатів за часом (приклад для TimescaleDB або pg_partman)

-- Дозволяє швидко архівувати старі дані

```
CREATE INDEX idx_results_created_at ON ClassificationResults(image_id, diagnosis);
```