**MedVision**

**Software Requirements Specification**

Version 1.0

15.04.2025

Holovashenko Illia

Lead Backend Software Engineer

Gorishnia Kateryna

Lead Frontend Software Engineer

## Зміст

1. Вступ.................................................................................................................4

1.1 Огляд продукту..........................................................................................4

1.2 Мета............................................................................................................4

1.3 Межі............................................................................................................4

1.4 Посилання..................................................................................................5

1.5 Означення та абревіатури.........................................................................5

2. Загальний опис................................................................................................6

2.1 Перспективи продукту..............................................................................6

2.2 Функції продукту.......................................................................................6

2.3 Характеристики користувачів..................................................................8

2.4 Загальні обмеження...................................................................................8

2.5 Припущення і залежності.........................................................................9

2.6 Розподіл завдань між учасниками розробки...........................................9

3. Конкретні вимоги..........................................................................................10

3.1 Вимоги до зовнішніх інтерфейсів..........................................................10

3.1.1 Інтерфейс користувача.....................................................................10

3.1.2 Апаратний інтерфейс.......................................................................11

3.1.3 Програмний інтерфейс.....................................................................11

3.1.4 Комунікаційний протокол...............................................................11

3.1.5 Обмеження пам’яті...........................................................................12

3.1.6 Операції.............................................................................................12

3.1.7 Функції продукту..............................................................................13

3.1.8 Припущення й залежності...............................................................14

3.2 Властивості програмного продукту.......................................................14

3.3 Атрибути програмного продукту ..........................................................15

3.3.1 Надійність..........................................................................................15

3.3.2 Доступність.......................................................................................15

3.3.3 Безпека...............................................................................................16

3.3.4 Супроводжуваність..........................................................................16

3.3.5 Переносимість...................................................................................17

3.3.6 Продуктивність.................................................................................17

3.4 Вимоги до бази даних.............................................................................18

3.5 Інші вимоги..............................................................................................18

4. Додаткові матеріали......................................................................................19

4.1 Аналітичні моделі....................................................................................19

4.4.1 Use Case Diagram..............................................................................19

4.4.2 Activity Diagram................................................................................21

4.4.3 Class Diagram....................................................................................23

4.4.4 Data Flow Diagram………................................................................25

4.4.5 Sequence Diagram..............................................................................26

4.4.6 Component Diagram..........................................................................28

4.4.7 Deployment Diagram.........................................................................30

4.2. Процес управління змінами...................................................................32

## 1. Вступ

## 1.1 Огляд продукту

MedVision – це інтелектуальна веб-орієнтована система для аналізу медичних томографічних зображень з використанням алгоритму Support Vector Machine. Система надає можливість завантаження, попередньої обробки, класифікації та візуалізації результатів медичних зображень.

Основними користувачами системи є лікарі, пацієнти та адміністратори. Вона дозволяє лікарям здійснювати детальний аналіз зображень, додавати коментарі, формувати діагнози та переглядати динаміку змін. Пацієнти можуть переглядати результати аналізів, отримувати сповіщення та експортувати звіти. Адміністратори мають можливість керувати обліковими записами користувачів і їх ролями.

MedVision не є діагностичним інструментом, але підтримує прийняття клінічних рішень, автоматизуючи аналіз знімків та забезпечуючи зручний доступ до історії досліджень.

**1.2 Мета**

Цей документ є повним описом функціональних і нефункціональних вимог до проєкту MedVision – інтелектуальної системи аналізу медичних зображень на основі методу опорних векторів.

Метою цього документа є створення структурованої основи для розробки, впровадження та тестування системи. Документ слугуватиме довідковим матеріалом для розробників, тестувальників, архітекторів систем, зацікавлених сторін проєкту та експертів у галузі медицини й медичних технологій.

**1.3 Межі**

MedVision є веб-орієнтованою системою, призначеною для автоматизованого аналізу медичних зображень з використанням алгоритму Support Vector Machine. Основне функціональне призначення системи – надання лікарям і пацієнтам інструментів для завантаження зображень, їх обробки, класифікації та перегляду результатів.

Система охоплює лише процеси, пов’язані з візуалізацією результатів, збереженням історії аналізів і керуванням користувачами. Вона не є засобом для постановки медичних діагнозів, не замінює фахову медичну оцінку, і не передбачає автономного використання без підключення до Інтернету. Також MedVision не інтегрується з іншими медичними інформаційними системами без додаткових засобів синхронізації через API.

**1.4 Посилання**

У процесі розробки програмної системи MedVision використовувалися такі стандарти, фреймворки та технічна документація:

IEEE Standard 830-1998 – IEEE Recommended Practice for Software Requirements Specifications.

Angular Documentation: https://angular.io – офіційна документація до фреймворку Angular.

Spring Boot Project: https://spring.io/projects/spring-boot – документація до фреймворку Spring Boot.

JavaCV Library: https://github.com/bytedeco/javacv – бібліотека для обробки зображень на Java.

MySQL Documentation: https://dev.mysql.com/doc/ – офіційна документація до системи керування базами даних MySQL.

GDPR Regulation: https://gdpr.eu/ – Загальний регламент захисту даних ЄС.

**1.5 Означення та абревіатури**

Нижче подано перелік термінів і скорочень, що використовуються в цьому документі:

SVM (Support Vector Machine) – алгоритм машинного навчання для задач класифікації.

UI (User Interface) – інтерфейс користувача.

REST API – архітектурний стиль для обміну даними між клієнтом і сервером.

SPA (Single Page Application) – односторінковий веб-застосунок.

HTTPS – захищений протокол передачі гіпертексту.

JSON (JavaScript Object Notation) – формат обміну структурованими даними.

JPG / PNG / DICOM – формати медичних зображень.

JavaCV – обгортка OpenCV для Java.

MySQL – реляційна система керування базами даних.

GDPR (General Data Protection Regulation) – регламент Європейського Союзу щодо захисту персональних даних.

JWT (JSON Web Token) – токен для аутентифікації та передачі інформації між сторонами.

**2. Загальний опис**

## 2.1 Перспективи продукту

MedVision розробляється як автономна веб-орієнтована система, що виконує інтелектуальний аналіз томографічних медичних зображень з використанням алгоритмів машинного навчання. Система взаємодіє з реляційною базою даних, REST API, а також модулем обробки зображень, побудованим на основі JavaCV та OpenCV.

Продукт є частиною дослідницького проєкту у галузі медичних інформаційних технологій та може бути інтегрований до інших медичних платформ через API. Система не є клінічним діагностичним інструментом, але підтримує прийняття медичних рішень шляхом надання структурованих результатів аналізу.

MedVision підтримує архітектуру типу клієнт-сервер із розподілом на frontend-клієнт (SPA) та backend-сервер на основі Spring Boot. Компоненти системи масштабуються та можуть бути розгорнуті як у хмарному середовищі, так і на локальному сервері установи.

**2.2 Функції продукту**

Система MedVision реалізує повний набір функцій для підтримки процесу обробки медичних зображень, взаємодії між користувачами та забезпечення прийняття клінічних рішень.

Основні функціональні можливості включають:

* Аутентифікація та авторизація користувачів. Передбачено реєстрацію та вхід для пацієнтів і лікарів з розмежуванням доступу до функціоналу відповідно до ролей.
* Завантаження медичних зображень. Користувачі (передусім лікарі) можуть завантажувати томографічні знімки через веб-інтерфейс із валідацією формату та розміру файлів.
* Попередня обробка зображень. Перед класифікацією система автоматично застосовує фільтрацію, нормалізацію та масштабування для підвищення якості аналізу.
* Класифікація зображень. Оброблені знімки аналізуються з використанням алгоритму SVM з метою виявлення патологічних ознак. Результати представлені у вигляді метрик (точність, повнота тощо) та теплових карт (heatmaps).
* Система нотаток і анотацій. Лікарі мають змогу додавати текстові коментарі, розмічати зображення, виділяти важливі ділянки та створювати персоналізовані примітки.
* Історія аналізів. Усі результати зберігаються у базі даних, доступні для перегляду, сортування, фільтрації та порівняння між собою.
* Візуалізація динаміки. Побудова графіків зміни показників аналізу або результатів класифікації у часі, що дозволяє оцінити ефективність лікування або прогрес захворювання.

Панель пацієнта включає персоналізований інтерфейс із можливістю:

* переглядати результати аналізів;
* отримувати сповіщення про нові або змінені висновки;
* відслідковувати динаміку стану;
* експортувати результати у форматі PDF.

Інтерфейс лікаря забезпечує:

* завантаження нових зображень;
* прив’язку результатів до конкретного пацієнта;
* створення висновків із використанням SVM;
* керування нотатками;
* порівняння аналізів одного пацієнта;
* формування PDF-звітів;
* перегляд історії змін діагнозів.

Функціонал адміністратора включає доступ до інструментів управління користувачами:

* редагування або видалення облікових записів;
* зміна ролей користувачів.

## 2.3 Характеристики користувачів

Система MedVision орієнтована на три основні категорії користувачів, кожна з яких має власні особливості доступу до функціоналу.

Лікарі. Це медичні фахівці, які мають досвід роботи з томографічними зображеннями та навички клінічного аналізу. Очікується володіння базовими навичками користування комп’ютером і веб-інтерфейсами. Лікарі взаємодіють із системою через персональну панель для завантаження знімків, аналізу результатів, формування діагнозів і керування історією пацієнтів.

Пацієнти. Кінцеві користувачі системи, які отримують доступ до результатів власних медичних аналізів. Від користувачів очікується базова комп’ютерна грамотність. Основними функціями для пацієнтів є перегляд результатів, порівняння аналізів, отримання повідомлень і завантаження PDF-звітів.

Адміністратори. Технічний персонал, відповідальний за підтримку, конфігурацію та управління обліковими записами. Адміністратори взаємодіють із системою переважно через інтерфейс адміністрування, однак не виконують безпосередню роботу з медичними зображеннями чи результатами аналізів.

## 2.4 Загальні обмеження

Функціонування системи MedVision передбачає дотримання низки загальних технічних та нормативних обмежень. Система повинна працювати виключно в сучасних веб-браузерах, зокрема таких як Google Chrome, Mozilla Firefox або Microsoft Edge, що забезпечують підтримку необхідних вебтехнологій. Доступ до всіх функцій системи має здійснюватися лише через захищене з’єднання з використанням протоколу HTTPS, що гарантує конфіденційність та цілісність переданих даних.

Обробка і зберігання персональної інформації користувачів системи повинні відповідати вимогам Загального регламенту захисту даних (GDPR), а також вимогам національного законодавства, що регламентує захист персональних даних. Окремо слід зазначити, що результати, які надає система в процесі аналізу медичних зображень, не мають юридичної сили як медичний діагноз. Вони носять виключно інформативний характер і призначені для підтримки клінічних рішень, а не для самостійного використання в медичній практиці без участі фахівця.

## 2.5 Припущення і залежності

У процесі розробки, розгортання та експлуатації системи MedVision приймаються певні припущення щодо середовища функціонування та технічної підтримки. Передбачається, що система буде розгорнута або в хмарному середовищі, або на локальному сервері медичної установи, залежно від потреб проєкту. Для забезпечення повноцінної роботи системи користувачі повинні мати стабільне підключення до Інтернету.

На серверній інфраструктурі мають бути встановлені всі необхідні компоненти, зокрема Java, Spring Boot, MySQL та бібліотека JavaCV для обробки зображень. Середовище обробки повинно підтримувати формати зображень JPG та PNG, які є основними у медичній візуалізації, що застосовується в системі. Крім того, розробникам надається доступ до повної документації API, відповідних бібліотек, а також прикладів томографічних знімків, необхідних для налаштування, тестування та валідації функціоналу системи.

## 2.6 Розподіл завдань між учасниками розробки

Розробка системи MedVision здійснюється двома основними фахівцями: бекенд-розробником та фронтенд-розробником, кожен з яких відповідає за окремі аспекти функціональності програмного забезпечення.

Бекенд-розробник відповідає за проєктування і реалізацію серверної логіки, включаючи побудову бази даних на основі MySQL, створення REST API за допомогою Spring Boot, а також реалізацію модуля класифікації зображень на базі алгоритму Support Vector Machine (SVM). До його обов’язків також входить формування PDF-звітів, побудова теплових карт (heatmaps), забезпечення аутентифікації користувачів за допомогою JWT-токенів, реалізація версіонування діагнозів, збереження нотаток і управління історією аналізів. Крім того, бекенд-розробник відповідає за завантаження та зберігання зображень у хмарному сховищі, а також за розгортання системи у середовищі Microsoft Azure.

Фронтенд-розробник забезпечує створення односторінкового клієнтського додатка (SPA) за допомогою Angular, розробку користувацьких інтерфейсів для лікарів, пацієнтів і адміністраторів, а також повну інтеграцію з REST API через захищений протокол HTTPS із використанням формату JSON. Він реалізує відображення результатів класифікації й теплових карт, створення форм для введення діагнозів і додавання приміток, а також обробку стану аутентифікації, керування токенами та ролями користувачів. Крім того, фронтенд-розробник відповідає за візуалізацію системи сповіщень та графічне представлення динаміки аналізів у часовому вимірі.

## 3. Конкретні вимоги

Цей розділ містить детальний опис функціональних і нефункціональних вимог до системи MedVision. Вимоги сформульовані таким чином, щоб їх можна було перевірити, простежити, оцінити за пріоритетністю та уникнути неоднозначностей. Вони слугують основою для подальшого проєктування, реалізації та тестування програмного забезпечення.

## 3.1 Вимоги до зовнішніх інтерфейсів

## 3.1.1 Інтерфейс користувача

Система MedVision повинна забезпечувати інтуїтивно зрозумілий та доступний веб-інтерфейс із повною підтримкою української мови. Інтерфейси розробляються з урахуванням потреб трьох основних категорій користувачів: пацієнтів, лікарів і адміністраторів.

Інтерфейс пацієнта дозволяє здійснювати перегляд результатів аналізів, отримувати сповіщення про нові або оновлені висновки, переглядати детальні результати кожного аналізу разом із відповідними тепловими картами, а також експортувати результати у форматі PDF. Крім того, пацієнт має доступ до персонального профілю з можливістю перегляду й редагування особистої інформації.

Інтерфейс лікаря забезпечує функції завантаження зображень для аналізу, управління пацієнтами, перегляду результатів класифікації, включаючи аналітичні метрики та теплові карти. Лікар може додавати коментарі та діагнози, отримувати підказки щодо ймовірних захворювань, створювати нові аналізи, а також порівнювати два аналізи одного пацієнта для візуалізації динаміки змін. Крім цього, система дозволяє лікарю генерувати звіти у форматі PDF як для окремого аналізу, так і для порівняння, та змінювати статус діагнозу за потреби.

Інтерфейс адміністратора орієнтований на технічне обслуговування системи. Він включає модулі для перегляду зареєстрованих користувачів, редагування їхніх ролей, а також функціональність для видалення облікових записів.

## 3.1.2 Апаратний інтерфейс

Система MedVision не висуває жорстких вимог до апаратного забезпечення на стороні клієнта. Для повноцінної роботи достатньо наявності персонального комп’ютера або ноутбука з встановленим сучасним веб-браузером, який підтримує сучасні вебстандарти та забезпечує коректне відображення інтерфейсу.

Серверна частина системи повинна мати доступ до обчислювальних ресурсів, достатніх для обробки зображень та виконання алгоритмів машинного навчання. Зокрема, за потреби підвищення продуктивності класифікації, рекомендовано використовувати сервер із доступом до графічного процесора (GPU). Також критичною є наявність достатнього обсягу оперативної пам’яті для стабільного функціонування сервісів обробки, збереження та аналізу медичних зображень.

## 3.1.3 Програмний інтерфейс

Архітектура програмної системи MedVision базується на клієнт-серверній моделі з використанням сучасних вебтехнологій. Клієнтська частина реалізована у вигляді односторінкового застосунку (SPA), створеного з використанням фреймворку Angular. Вона взаємодіє з сервером через REST API, який реалізовано за допомогою Spring Boot.

Для обробки та класифікації медичних зображень серверна частина інтегрує Java-бібліотеку JavaCV у зв’язці з OpenCV. Це дозволяє виконувати попередню обробку знімків, аналіз даних та генерацію теплових карт. Зберігання структурованих даних (користувачі, результати аналізів, діагнози, нотатки) реалізовано на основі реляційної бази даних MySQL.

Система підтримує роботу з основними форматами медичних зображень, зокрема JPG, PNG та DICOM, що забезпечує гнучкість у прийомі вхідних даних з різних джерел.

**3.1.4 Комунікаційний протокол**

Для забезпечення безпечної та ефективної взаємодії між клієнтською та серверною частинами системи MedVision використовується протокол HTTPS, який гарантує захищену передачу даних шляхом шифрування трафіку. Це є критично важливим для захисту персональної інформації та медичних зображень, що передаються між користувачем і сервером.

Усі запити та відповіді між frontend-застосунком (Angular) і backend-сервером (Spring Boot) формуються у форматі JSON. Обраний формат є легким для обробки, добре структурованим і широко підтримуваним сучасними вебтехнологіями, що дозволяє забезпечити ефективну обробку запитів та узгоджене представлення даних на всіх рівнях системи.

## 3.1.5 Обмеження пам’яті

Система MedVision передбачає обробку медичних зображень високої роздільної здатності, тому серверна частина має бути забезпечена достатнім обсягом оперативної пам’яті для стабільного функціонування алгоритмів класифікації та попередньої обробки знімків. Особливо це стосується компонентів, які використовують бібліотеки JavaCV та OpenCV для фільтрації, масштабування, нормалізації та побудови теплових карт.

Зберігання оригінальних зображень і результатів аналізу реалізовано із застосуванням хмарного файлового сховища (Azure Blob Storage), що дозволяє зменшити навантаження на базу даних та серверну файлову систему. Структуровані дані (користувачі, аналізи, діагнози, нотатки) зберігаються у реляційній базі даних MySQL. Обов’язковою є підтримка механізмів регулярного резервного копіювання для запобігання втраті інформації.

Зображення приймаються у форматах JPG, PNG і DICOM. Їх розмір обмежується згідно з валідаційними правилами, вбудованими у фронтенд і бекенд, щоб уникнути перевантаження системи при обробці файлів надмірного обсягу. Передбачено автоматичну валідацію завантажених зображень на стороні клієнта та сервера.

## 3.1.6 Операції

У системі MedVision передбачено низку типових операцій, які реалізуються відповідно до ролі користувача та охоплюють повний цикл роботи з медичними зображеннями – від завантаження до формування діагнозу та перегляду результатів.

Користувачі проходять процедуру реєстрації та авторизації, після чого отримують доступ до персоналізованого інтерфейсу залежно від своєї ролі. Пацієнти мають змогу переглядати історію власних аналізів, ознайомлюватися з результатами класифікації зображень, переглядати теплові карти, порівнювати попередні результати та завантажувати PDF-звіти. Лікарі, у свою чергу, здійснюють завантаження томографічних знімків, ініціюють їх обробку, додають діагнози та коментарі, розмічають важливі області зображення, порівнюють кілька аналізів одного пацієнта, а також генерують підсумкові звіти.

Операції, пов’язані з аналізом зображень, включають перевірку формату та розміру файлу, автоматичну попередню обробку (масштабування, фільтрацію, нормалізацію), запуск класифікатора SVM, генерацію аналітичних метрик (точність, повнота, recall), створення теплової карти та збереження результатів у базі даних.

Лікар може редагувати діагнози та статуси аналізів (наприклад, «Перевірено» або «Потребує доопрацювання»), а також додавати або змінювати нотатки, що пов’язані з конкретними знімками. Адміністратор має доступ до операцій із керування обліковими записами: перегляд списку користувачів, зміна ролей і видалення облікових записів.

Таким чином, система підтримує повний набір операцій для трьох ролей – пацієнта, лікаря та адміністратора – забезпечуючи повноцінну функціональність у межах визначених повноважень.

## 3.1.7 Функції продукту

Система MedVision надає користувачам набір функцій для автоматизованого аналізу медичних зображень із використанням алгоритмів машинного навчання. Основні функції включають реєстрацію та авторизацію користувачів із розмежуванням доступу за ролями, завантаження томографічних знімків, попередню обробку зображень, класифікацію за допомогою методу опорних векторів, а також візуалізацію результатів у вигляді аналітичних метрик і теплових карт.

Система підтримує введення та редагування діагнозів, додавання текстових і графічних нотаток, збереження та перегляд історії аналізів, порівняння результатів у динаміці, а також генерацію звітів у форматі PDF. Лікарі мають розширені можливості керування аналізами пацієнтів, тоді як пацієнти можуть переглядати власні результати та отримувати повідомлення. Адміністратори відповідають за керування обліковими записами та ролями користувачів.

Функціональність системи реалізована у вигляді клієнт-серверної архітектури з веб-інтерфейсом, що забезпечує зручну та безпечну взаємодію з користувачем.

## 3.1.8 Припущення й залежності

Функціонування системи MedVision базується на низці припущень щодо зовнішнього середовища, інфраструктури та доступності допоміжних ресурсів. Передбачається, що система буде розгорнута або у хмарному середовищі, зокрема в Microsoft Azure, або на локальному сервері медичної установи з відповідними технічними параметрами.

Стабільне підключення до Інтернету є обов’язковим для всіх користувачів, оскільки система реалізована як веборієнтований клієнт-серверний застосунок. На серверній частині повинні бути встановлені всі необхідні компоненти – Java, Spring Boot, MySQL та JavaCV – для забезпечення обробки зображень, керування базою даних та реалізації REST API.

Крім того, вважається, що середовище обробки підтримує формати зображень JPG, PNG і DICOM. У процесі розробки передбачається, що розробники мають доступ до документації API, зовнішніх бібліотек, прикладів медичних знімків, а також інструментів для тестування та налагодження системи.

## 3.2 Властивості програмного продукту

Властивості програмного продукту визначають його нефункціональні характеристики, які є критично важливими для успішного використання, супроводу та подальшого розвитку системи. До таких властивостей належать загальні якості програмного забезпечення, що впливають на його поведінку за умов реальної експлуатації.

Система MedVision має відповідати вимогам до надійності, доступності, безпеки, супроводжуваності, переносимості та продуктивності. Ці атрибути відображають очікування до якості програмного продукту з боку кінцевих користувачів, розробників та обслуговуючого персоналу.

Детальний опис кожного атрибуту наведено у підрозділі 3.3 Атрибути програмного продукту.

## 3.3 Атрибути програмного продукту

## 3.3.1 Надійність

Система MedVision повинна забезпечувати стабільну та передбачувану роботу в умовах нормального та інтенсивного навантаження. Надійність програмного продукту полягає у його здатності безвідмовно виконувати функції аналізу зображень, зберігання результатів та взаємодії з користувачами протягом тривалого часу.

Очікується, що не менше 95% запитів до системи будуть завершуватися успішно без виникнення критичних помилок. У разі виникнення помилок або збоїв, система повинна зберігати цілісність даних і надавати користувачу відповідні повідомлення без порушення загальної працездатності інтерфейсу.

Критичні функції (наприклад, завантаження зображень, класифікація, збереження результатів) мають бути побудовані з урахуванням обробки винятків і автоматичного відновлення після збоїв на рівні API або внутрішніх сервісів. Надійність також передбачає відсутність втрат даних під час штатної роботи або в разі короткочасної втрати з’єднання.

## 3.3.2 Доступність

Система MedVision повинна бути доступною для користувачів у режимі 24/7, за винятком періодів запланованого технічного обслуговування. Високий рівень доступності є критично важливим для забезпечення безперервного доступу лікарів та пацієнтів до результатів аналізів, історії медичних зображень та супутньої інформації.

Інтерфейс користувача має бути доступним з будь-якого пристрою, що підтримує сучасний веб-браузер. Всі функції повинні бути повністю доступні з моменту входу до системи та не залежати від клієнтської платформи чи операційної системи. У випадку тимчасової недоступності сервісу має бути реалізований механізм повідомлення користувача про технічні причини з очікуваним часом відновлення.

Також доступність передбачає, що інтерфейс має бути зручним для користувачів із базовими цифровими навичками, включно з пацієнтами. Система повинна забезпечувати мінімальний час завантаження основних сторінок і функцій незалежно від навантаження.

## 3.3.3 Безпека

Система MedVision обробляє конфіденційні персональні та медичні дані, тому питання безпеки є пріоритетним. Усі обмінювані дані між клієнтом і сервером повинні передаватися виключно через захищене з’єднання за протоколом HTTPS із використанням сучасних методів шифрування.

Аутентифікація користувачів здійснюється за допомогою механізму JSON Web Token (JWT), що забезпечує безпечне збереження сесії та авторизований доступ до ресурсів системи. Кожен запит до захищених API має супроводжуватися перевіркою токена та прав доступу згідно з роллю користувача (пацієнт, лікар, адміністратор).

Доступ до даних має бути обмеженим згідно з принципом мінімальних привілеїв: пацієнти можуть переглядати лише власну інформацію, лікарі – дані пов’язаних пацієнтів, а адміністратори – тільки функції управління обліковими записами.

Система повинна відповідати вимогам Загального регламенту захисту даних та національного законодавства України щодо обробки персональної інформації. Має бути передбачено механізм журналювання критичних дій (аудит), а також регулярне резервне копіювання даних.

## 3.3.4 Супроводжуваність

Архітектура системи MedVision має забезпечувати простоту внесення змін, оновлень і розширення функціональності без необхідності істотної переробки існуючого коду. Для досягнення цього програмний продукт реалізовано з використанням модульного підходу: фронтенд і бекенд функціонують як незалежні частини з чітко визначеними API.

Кодова база повинна бути структурованою, добре задокументованою та відповідати загальноприйнятим стандартам програмування. Має бути забезпечена можливість швидкої локалізації та виправлення помилок, а також розгортання нових версій без порушення роботи існуючої системи.

Тестування має здійснюватися на основі автоматизованих модульних та інтеграційних тестів. Супровід системи має здійснюватися технічним персоналом, що має доступ до вихідного коду, бази знань і журналів подій, а також до інструментів моніторингу стану системи у хмарному середовищі або на сервері установи.

## 3.3.5 Переносимість

Система MedVision повинна бути гнучкою щодо середовища розгортання. Програмний продукт має підтримувати як хмарну інфраструктуру (Microsoft Azure), так і локальні сервери медичних установ за умови дотримання вимог до серверного оточення (Java, Spring Boot, MySQL, JavaCV).

Фронтенд реалізовано як SPA-додаток на Angular, що дозволяє запускати клієнтську частину на будь-якому пристрої з сучасним браузером (Chrome, Firefox, Edge) без потреби встановлення додаткового ПЗ. Це забезпечує сумісність з різними операційними системами – Windows, macOS, Linux.

Серверна частина має бути контейнеризованою, що спрощує міграцію між середовищами та автоматизує процес розгортання. Всі зовнішні залежності мають бути чітко задокументовані, а система повинна підтримувати розділення конфігурацій залежно від середовища.

## 3.3.6 Продуктивність

Система MedVision повинна забезпечувати високу швидкодію в обробці запитів та аналізі зображень. Критичні операції, такі як завантаження медичних знімків, їх класифікація за допомогою методу опорних векторів та генерація теплових карт, мають виконуватись із мінімальною затримкою. У стандартних умовах час обробки одного зображення не повинен перевищувати 5 секунд за наявності серверного GPU-прискорення.

Інтерфейс користувача має реагувати на дії користувача протягом 1 секунди у 95% випадків. Завантаження основних сторінок особистого кабінету пацієнта або лікаря не повинно перевищувати 2 секунд навіть при середньому рівні навантаження.

Система повинна підтримувати одночасну роботу щонайменше 100 активних користувачів без значного зниження швидкодії. Архітектура повинна дозволяти масштабування продуктивності за рахунок горизонтального масштабування компонентів у хмарному середовищі.

Архітектура системи побудована таким чином, щоб забезпечити горизонтальну масштабованість. Кожен з компонентів – веб-клієнт, API-сервер, база даних, сервіси обробки зображень – може бути масштабований незалежно в хмарному середовищі. Це дозволяє гнучко адаптувати систему до зростання кількості користувачів і обсягу оброблюваних даних без втрати продуктивності.

## 3.4 Вимоги до бази даних

Система MedVision використовує реляційну базу даних MySQL для зберігання структурованої інформації, пов’язаної з користувачами, медичними зображеннями, результатами аналізів, історією діагнозів, замітками лікарів та іншими супровідними даними. Структура бази повинна бути розроблена з дотриманням принципів нормалізації, що забезпечить мінімізацію надлишковості даних, збереження цілісності та ефективність при виконанні запитів.

База даних повинна підтримувати ролі користувачів, що визначають їх права доступу до функцій системи, та зберігати пов’язану інформацію з прив’язкою до конкретного пацієнта чи лікаря. Особлива увага приділяється безпеці – персональні дані повинні зберігатися у захищеному вигляді, паролі мають бути хешовані, а доступ до медичних записів обмежений через авторизований API.

Система повинна вести журнал змін, пов’язаних із діагнозами, замітками та результатами аналізів, зберігаючи інформацію про час змін і відповідального користувача. Всі зв’язки між сутностями — користувачами, аналізами, файлами зображень – повинні бути реалізовані у вигляді чіткої схеми з використанням зовнішніх ключів.

З метою масштабованості структура бази має підтримувати роботу з великим обсягом даних без суттєвого зниження продуктивності. Доступ до бази даних повинен здійснюватися виключно через серверну логіку, реалізовану у Spring Boot, з використанням захищених з’єднань та ізоляції конфігураційних параметрів через змінні середовища.

## 3.5 Інші вимоги

Система MedVision повинна забезпечувати додаткові функціональні та технічні характеристики, які не були розглянуті в основних розділах специфікації. Зокрема, передбачається впровадження механізму логування подій, що дасть змогу фіксувати ключові дії користувачів, зміни у даних та доступ до медичної інформації. Це важливо для забезпечення прозорості, аудиту та захисту від несанкціонованих втручань.

Інтерфейс системи повинен бути адаптивним, з можливістю коректного відображення на пристроях із різними розмірами екранів, включно з мобільними телефонами та планшетами. Також передбачається потенційна сумісність системи з іншими медичними інформаційними платформами через REST API, що дозволить розширити її функціональність. Для цього розробники мають надати повну документацію до API у форматі Swagger або OpenAPI, що спростить інтеграцію з зовнішніми сервісами та підтримку системи в майбутньому.

## 4. Додаткові матеріали

## 4.1 Аналітичні моделі

У цьому розділі наведено графічні моделі, що були використані під час розробки програмної системи MedVision. Вони ілюструють функціональні, структурні, динамічні та поведінкові аспекти роботи клієнтської частини, що забезпечує взаємодію з користувачами, обробку медичних зображень і відображення результатів аналізу.

## 4.4.1 Use Case Diagram

Діаграма варіантів використання (Use Case Diagram) відображає основні функціональні можливості системи MedVision, які доступні для різних категорій користувачів. Діаграма дозволяє візуалізувати сценарії взаємодії акторів із системою, встановлюючи межі відповідальності та функціональні ролі (див. рис. 4.1).

Актори системи:

* Patient (Пацієнт) – кінцевий користувач, який може переглядати результати аналізів, історію, отримувати повідомлення та завантажувати звіти;
* Doctor (Лікар) – медичний фахівець, що здійснює завантаження зображень, аналіз, додавання діагнозів, ведення історії пацієнтів;
* Administrator (Адміністратор) – технічний адміністратор системи, відповідальний за управління обліковими записами та ролями користувачів.

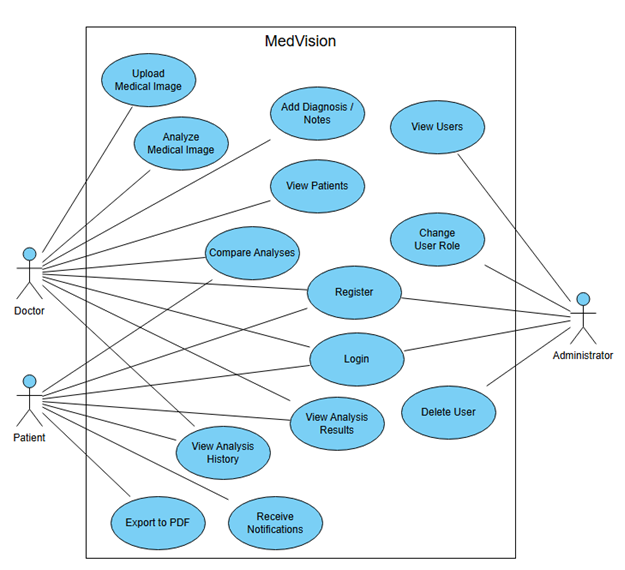


Рисунок 4.1 – MedVision Use Case Diagram

Основні варіанти використання.

Register – створення нового облікового запису (для всіх акторів);

Login – автентифікація користувача з наданням доступу до персоналізованого інтерфейсу;

Upload Medical Image – передача томографічного зображення для подальшої обробки (доступно лише лікарю);

Analyze Image – запуск алгоритму SVM для класифікації зображення та отримання результатів (доступно лікарю);

Add Diagnosis / Notes – внесення текстового діагнозу або позначення області інтересу на зображенні (тільки для лікаря);

View Analysis Results – перегляд результатів аналізу (для пацієнта і лікаря);

View Analysis History – доступ до попередніх аналізів (для пацієнта і лікаря);

Receive Notifications – отримання системних повідомлень про нові результати чи зміни (пацієнт);

Export to PDF – можливість зберегти результат у вигляді документа PDF (пацієнт);

View Patients – перегляд списку пацієнтів і результатів їх аналізів (доступно лікарю);

Compare Analyses – аналіз змін між результатами в динаміці (для лікаря та паціента);

View Users – перегляд усіх зареєстрованих користувачів (доступно адміністратору);

Change User Role – зміна ролі користувача (доступно лише адміністратору);

Delete User – видалення облікового запису користувача (адміністратор).

Кожен актор пов’язаний лініями асоціації з відповідними варіантами використання, що демонструє, які саме дії він може ініціювати в рамках системи. Діаграма допомагає формалізувати функціональну модель MedVision з точки зору користувацького досвіду.

## 4.4.2 Activity Diagram

Діаграма активностей (Activity Diagram) моделює послідовність дій користувача під час виконання одного з ключових процесів системи MedVision – аналізу медичного зображення (див. рис. 4.2). Цей сценарій охоплює етапи, які проходить лікар від моменту авторизації до формування висновку за результатами класифікації зображення.

Лікар входить до системи, переходить до розділу завантаження зображень, обирає файл томографії та ініціює його обробку. Система перевіряє валідність файлу, після чого виконує попередню обробку зображення (масштабування, фільтрація, нормалізація) та передає його на класифікацію алгоритмом Support Vector Machine (SVM). У результаті формуються аналітичні метрики та теплова карта. Після перегляду результатів лікар може додати діагноз та коментарі.

A diagram of a system

AI-generated content may be incorrect.

Рисунок 4.2 – MedVision Activity Diagram

Основні етапи процесу:

Login to system – автентифікація лікаря в системі;

Navigate to upload section – перехід до розділу завантаження зображень;

Select medical image file – вибір зображення для аналізу;

Is image valid? – перевірка формату й розміру зображення;

Якщо зображення є невалідним → виконується дія Show error message, після чого процес завершується;

Якщо зображення валідне → відбувається подальша обробка;

System validates image format and size – системна валідація файлу;

System preprocesses image – застосування фільтрів та нормалізація;

System performs SVM classification – класифікація зображення алгоритмом SVM;

System generates analysis results and heatmap – створення метрик класифікації та теплової карти;

Doctor reviews results – перегляд результатів лікарем;

Doctor adds diagnosis and notes – формування висновку та приміток.

Особливості моделі:

* початкова точка позначена стандартним символом Start Node (заповнене чорне коло);
* гілкування реалізовано за допомогою умовного вузла (Decision Node), що моделює перевірку валідності зображення;
* завершення процесу позначено End Node (жирна крапка з контуром);
* всі дії виконуються послідовно, крім випадку невалідного файлу, що завершує процес раніше.

Ця діаграма допомагає наочно представити логіку поведінки системи під час аналізу медичних знімків і може використовуватись як основа для реалізації відповідного модуля.

## 4.4.3 Class Diagram

Діаграма класів (Class Diagram) демонструє статичну структуру програмної системи MedVision, зосереджену на основних об’єктах, їх атрибутах, методах та взаємозв’язках (рис. 4.3). Вона дозволяє візуалізувати логіку побудови об’єктно-орієнтованої моделі, яка лежить в основі процесів збереження, обробки та аналізу медичних даних.

Основні класи:

User – базовий клас, який представляє загальну інформацію про користувача (ім’я, email, пароль, роль тощо). Використовується як основа для класів Doctor та Patient.

Doctor – розширення класу User, яке містить медичні дані (посада, відділення, ліцензія) та методи для перегляду пацієнтів, завантаження зображень і додавання діагнозу.

Patient – також наслідує User, зберігає інформацію про пацієнта (дата народження, стать) та дозволяє переглядати історію аналізів, отримувати сповіщення й експортувати результати.

ImageAnalysis – клас, що відповідає за результат обробки зображення: метрики класифікації (accuracy, precision, recall), теплову карту та діагноз. Має зв’язки з Doctor і Patient.

AnalysisNote – представляє текстову примітку або коментар лікаря, що пов’язується з конкретним зображенням (ImageAnalysis) та може включати координати області інтересу.

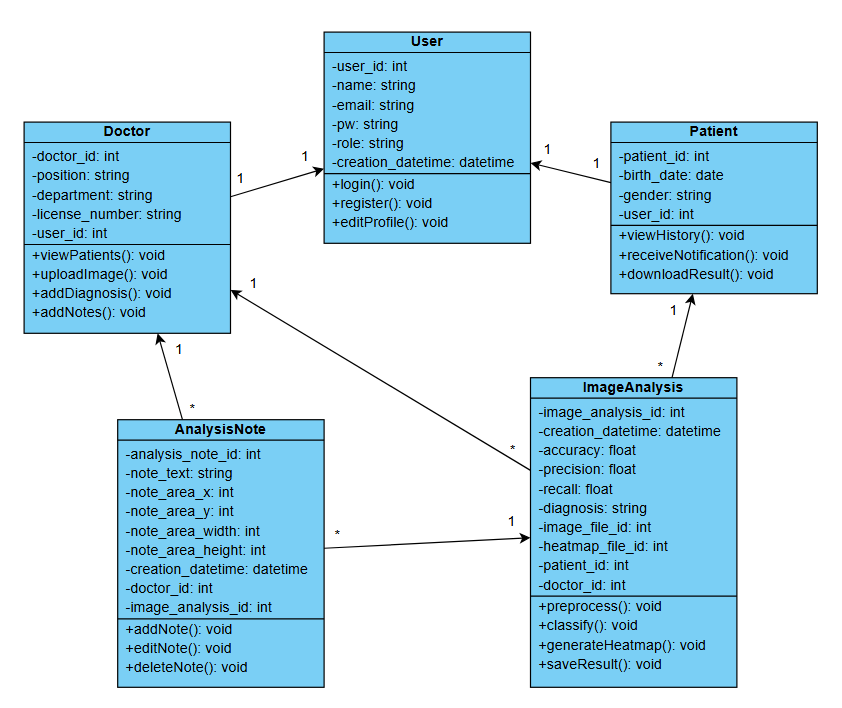
****

Рисунок 4.3 – MedVision Class Diagram

Зв’язки між класами:

Doctor та Patient мають зв’язок 1:1 з User, що забезпечує уніфіковане керування обліковими записами.

Один Doctor може мати багато ImageAnalysis, але кожен аналіз прив’язаний до одного лікаря (1:\*).

Аналогічно, один Patient може мати багато ImageAnalysis, але кожен аналіз стосується лише одного пацієнта (1:\*).

Клас AnalysisNote пов’язаний з ImageAnalysis та Doctor у співвідношенні 1:\*, тобто одне зображення чи лікар може мати кілька нотаток.

Кожен клас містить набір атрибутів та публічних методів, які відповідають за логіку роботи з відповідною сутністю, з дотриманням правил інкапсуляції.

## 4.4.4 Data Flow Diagram

Діаграма потоків даних (Data Flow Diagram) відображає ключові інформаційні потоки в системі MedVision, моделюючи взаємодію між користувачами, підсистемами, базами даних і модулями обробки. DFD дозволяє побачити, як дані переміщуються в межах системи, починаючи від введення користувачем і завершуючи збереженням результатів аналізу або виведенням звітів (див. рис. 4.4).

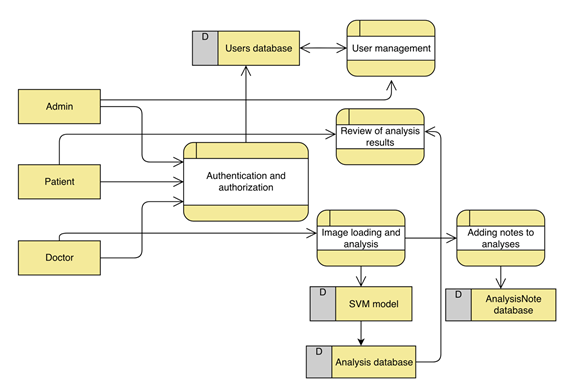


Рисунок 4.4 – MedVision Data Flow Diagram

Система включає п’ять основних підсистем обробки даних.

Перша – Authentication and Authorization – відповідає за обробку облікових даних, які вводять Пацієнт, Лікар або Адміністратор. Введені дані надсилаються до модуля перевірки, який здійснює валідацію через базу даних користувачів (Users database). Після успішної перевірки користувач отримує доступ до відповідної частини системи.

Наступний модуль – User Management – доступний виключно адміністратору. Він забезпечує повний контроль над обліковими записами, включаючи створення, редагування та видалення. Усі дії безпосередньо впливають на Users database, яка є основним джерелом інформації про облікові записи.

Модуль Image Loading and Analysis активується лікарем для завантаження томографічних знімків. Завантажене зображення передається до класифікаційного модуля SVM Model, де проходить автоматичний аналіз. Після завершення обробки результати класифікації (включаючи heatmap) записуються до бази даних аналізів (Analysis database).

Далі – Review of Analysis Results – функціональність, доступна лікарям та пацієнтам. Вона дозволяє переглядати результати аналізів, які отримуються з Analysis database. Також забезпечується доступ до історії попередніх аналізів.

Останній модуль – Adding Notes to Analyses – доступний лише лікарям і дає змогу додавати текстові та візуальні коментарі до результатів аналізів. Ці нотатки зберігаються окремо в AnalysisNote database та пов’язані з відповідними зображеннями.

Загалом, DFD MedVision демонструє, як дані проходять усі етапи обробки – від автентифікації та завантаження знімків до аналізу, збереження результатів і їх подальшої інтерпретації лікарем чи пацієнтом.

## 4.4.5 Sequence Diagram

Діаграма послідовності (Sequence Diagram) демонструє послідовну взаємодію основних учасників програмної системи MedVision із модулями інтерфейсу, бекенду, обробки зображень та базою даних (див. рис. 4.5). Вона відображає ключові функціональні сценарії, які реалізуються на платформі під час виконання типових дій користувачів.

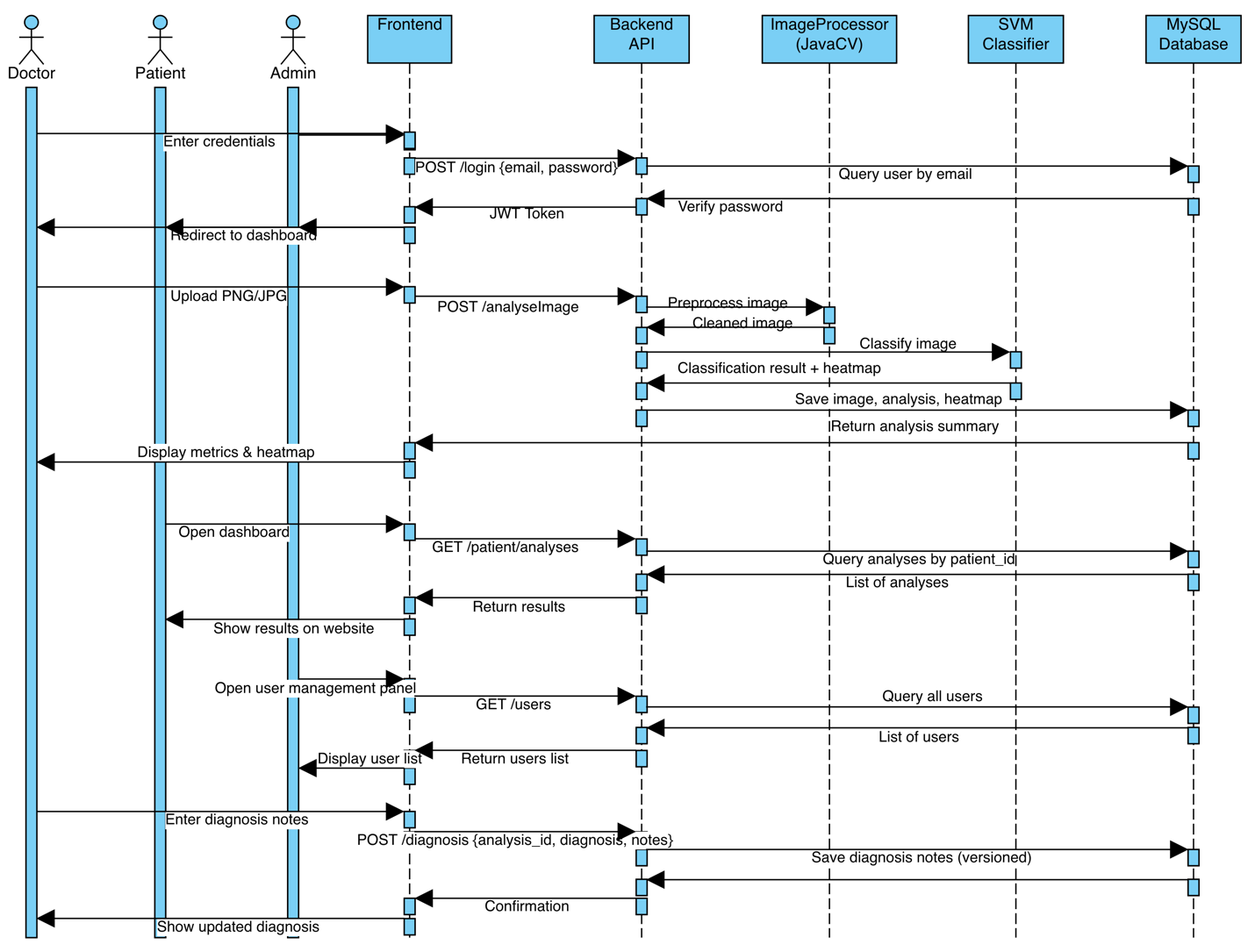


Рисунок 4.5 – MedVision Sequence Diagram

Учасники системи:

* Doctor (Лікар) – здійснює завантаження зображень, формування діагнозу, перегляд результатів аналізу;
* Patient (Пацієнт) – переглядає результати обстежень;
* Admin (Адміністратор) – управляє обліковими записами користувачів;
* Frontend – клієнтська частина, реалізована як SPA на Angular;
* Backend API – серверна логіка, реалізована на Spring Boot;
* ImageProcessor – модуль попередньої обробки зображень (JavaCV);
* SVM Classifier – класифікаційний алгоритм на основі Support Vector Machine;
* MySQL Database – централізоване сховище даних.

Опис основних сценаріїв:

* Авторизація користувача. Користувач (лікар, пацієнт або адміністратор) вводить облікові дані у формі входу. Дані передаються через запит POST /login до Backend API. Сервер здійснює перевірку логіна і пароля через MySQL Database. У випадку успішної автентифікації видається JWT-токен і виконується перехід до основної сторінки.
* Завантаження та аналіз медичного зображення (лікар). Лікар завантажує зображення (у форматі PNG/JPG). Frontend надсилає його у запиті POST /analyseImage. Зображення надходить до ImageProcessor, де проходить попередню обробку (фільтрацію, масштабування тощо). Далі воно аналізується класифікатором SVM. Результати у вигляді метрик та теплової карти зберігаються у MySQL Database і повертаються лікарю через UI.
* Перегляд результатів аналізів (пацієнт). Пацієнт переходить до сторінки перегляду результатів. Frontend виконує запит GET /patient/analyses, який обробляється Backend API. Після звернення до бази за patient\_id, пацієнт отримує структурований список аналізів.
* Управління користувачами (адміністратор). Адміністратор відкриває інтерфейс керування користувачами. Frontend ініціює запит GET /users, після чого Backend отримує дані з MySQL Database і повертає їх для відображення у табличному вигляді.
* Додавання діагнозу (лікар). Лікар формує діагноз, додає примітки до конкретного аналізу, після чого відправляє запит POST /diagnosis із параметрами analysis\_id, текстом діагнозу та коментарем. Сервер зберігає оновлені дані до бази із можливістю версіонування та повертає підтвердження для оновлення інтерфейсу.

## 4.4.6 Component Diagram

Діаграма компонентів (Component Diagram) демонструє логічну архітектуру програмної системи MedVision, відображаючи основні структурні модулі, способи їхньої взаємодії та реалізаційні залежності (див. рис. 4.6). Вона слугує для візуалізації побудови системи на рівні компонентів та технологічних взаємозв’язків між ними.

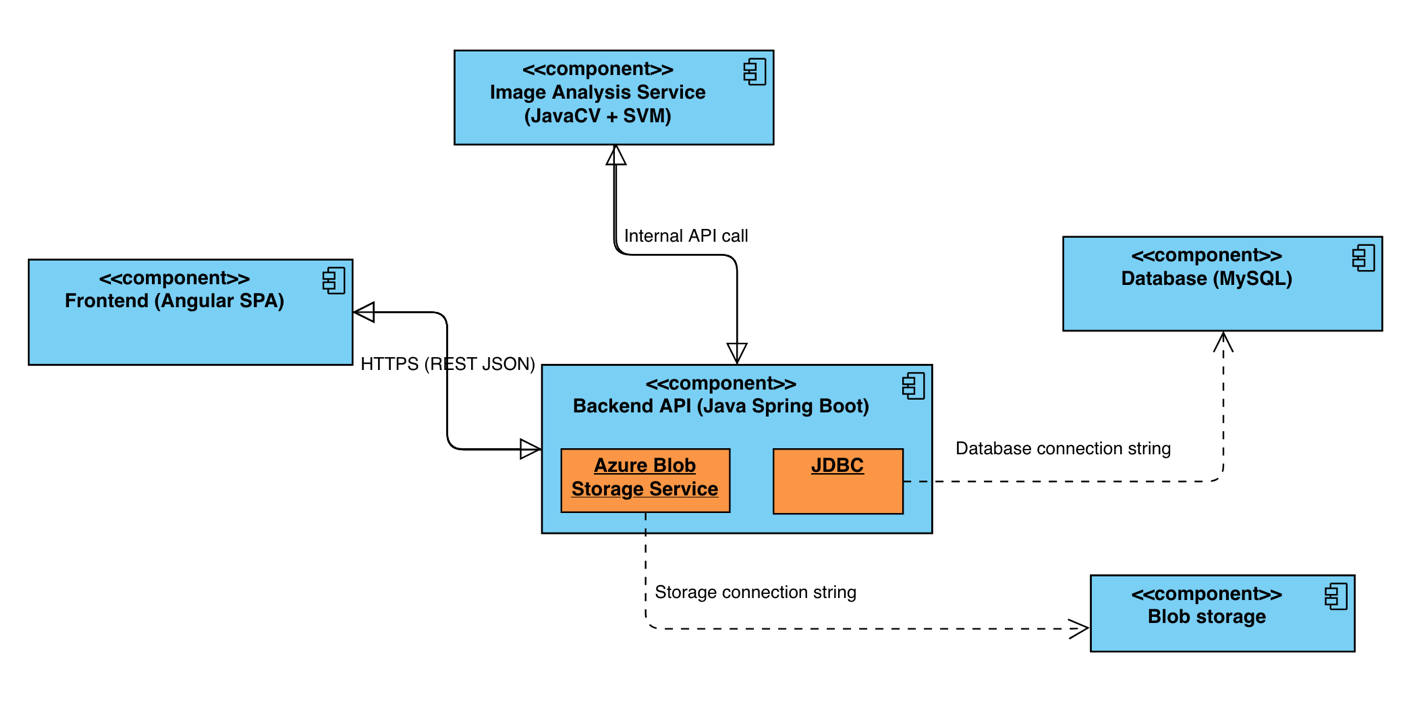


Рисунок 4.6 – MedVision Component Diagram

Основні компоненти:

* Frontend (Angular SPA). Односторінковий веб-додаток, що реалізує графічний інтерфейс користувача для лікарів, пацієнтів та адміністратора. Забезпечує формування HTTP-запитів до серверної частини, надсилаючи дані у форматі JSON. Реалізовано з використанням Angular.
* Backend API (Spring Boot). Серверна частина, відповідальна за бізнес-логіку, авторизацію, обробку запитів та взаємодію з базами даних і модулями обробки зображень. Включає: JDBC-модуль для доступу до MySQL бази даних; Azure Blob Storage Service – інтеграційний підкомпонент для збереження файлів зображень і heatmap'ів у хмарному середовищі. Комунікація з фронтендом відбувається через REST API.
* Image Processing Module (JavaCV + SVM). Компонент обробки медичних зображень. Виконує попередню фільтрацію, масштабування та класифікацію за допомогою алгоритму опорних векторів (SVM). Розміщується окремо від основного API та взаємодіє з ним через внутрішній API-виклик (Internal API call). Результати передаються назад до Backend API.
* Database (MySQL). Реляційна база даних, що зберігає інформацію про користувачів, ролі, аналізи, діагнози, історію змін та коментарі. Взаємодія з сервером відбувається через стандартний JDBC-драйвер.
* Blob Storage (Azure). Компонент хмарного зберігання, який використовується для збереження великих медичних зображень, а також згенерованих теплових карт (heatmaps). Підключення реалізовано через Azure Storage Access (SA) connection string, з доступом лише зі сторони бекенду.

## 4.4.7 Deployment Diagram

Діаграма розгортання (Deployment Diagram) демонструє інфраструктурну архітектуру програмної системи MedVision, реалізовану на базі хмарної платформи Microsoft Azure (див. рис. 4.7). Frontend і Backend розгорнуті на окремих Azure App Service, що масштабуються горизонтально через кілька екземплярів додатку. Для зберігання зображень використовується Azure Cloud Storage, а для реляційних даних – Azure Database for MySQL. Всі сервіси взаємодіють через захищені протоколи (HTTPS, JDBC, File Access). Така архітектура забезпечує високу доступність, масштабованість та розділення відповідальності компонентів системи.

Розгортання реалізовано у вигляді взаємопов’язаних процесорів (processors) та артефактів (artifacts), що відображають фізичні вузли розміщення програмних модулів.

* Processor 1 – Azure App Service (Frontend). Компонент: <<artifact>> MedVision Frontend App. Цей процесор розгортає односторінковий клієнтський додаток на базі Angular. Він відповідає за взаємодію з кінцевими користувачами (пацієнтами, лікарями, адміністраторами), формування HTTP-запитів та візуалізацію результатів аналізів. Реалізовано як два екземпляри на Web Server 3 та Web Server 4 (Instance 1 і Instance 2). Комунікація з Backend здійснюється через HTTPS у форматі REST API (JSON).
* Processor 2 – Azure App Service (Backend). Компонент: <<artifact>> MedVision Backend App. Основна бізнес-логіка системи, розроблена з використанням Spring Boot. Обробляє запити з Frontend, взаємодіє з базою даних, класифікатором зображень і файловим сховищем. Запущено на Web Server 1 та Web Server 2 (Instance 1 і 2). Виконує: обробку запитів через HTTPS, SQL-запити до бази даних через JDBC, доступ до зображень через File Access.
* Processor 3 – Azure Database for MySQL (Flexible Server). Компонент: <<artifact>> MedVision MySQL DB instance. Централізоване сховище структурованих даних: облікові записи користувачів та ролі, історія аналізів і результати класифікації, діагнози, коментарі, метадані. Доступ з Backend API забезпечується через JDBC-з’єднання.
* Processor 4 – Azure Cloud Storage Account. Компонент: <<artifact>> MedVision Storage container. Файлове сховище великих об’ємів, де зберігаються: завантажені медичні зображення (JPG, PNG), теплові карти (heatmaps), згенеровані класифікатором. Доступ до контейнера здійснюють: Backend API, ImageProcessor (JavaCV) – для завантаження й зчитування зображень. Комунікація виконується через File Access (Azure SDK або HTTPS).

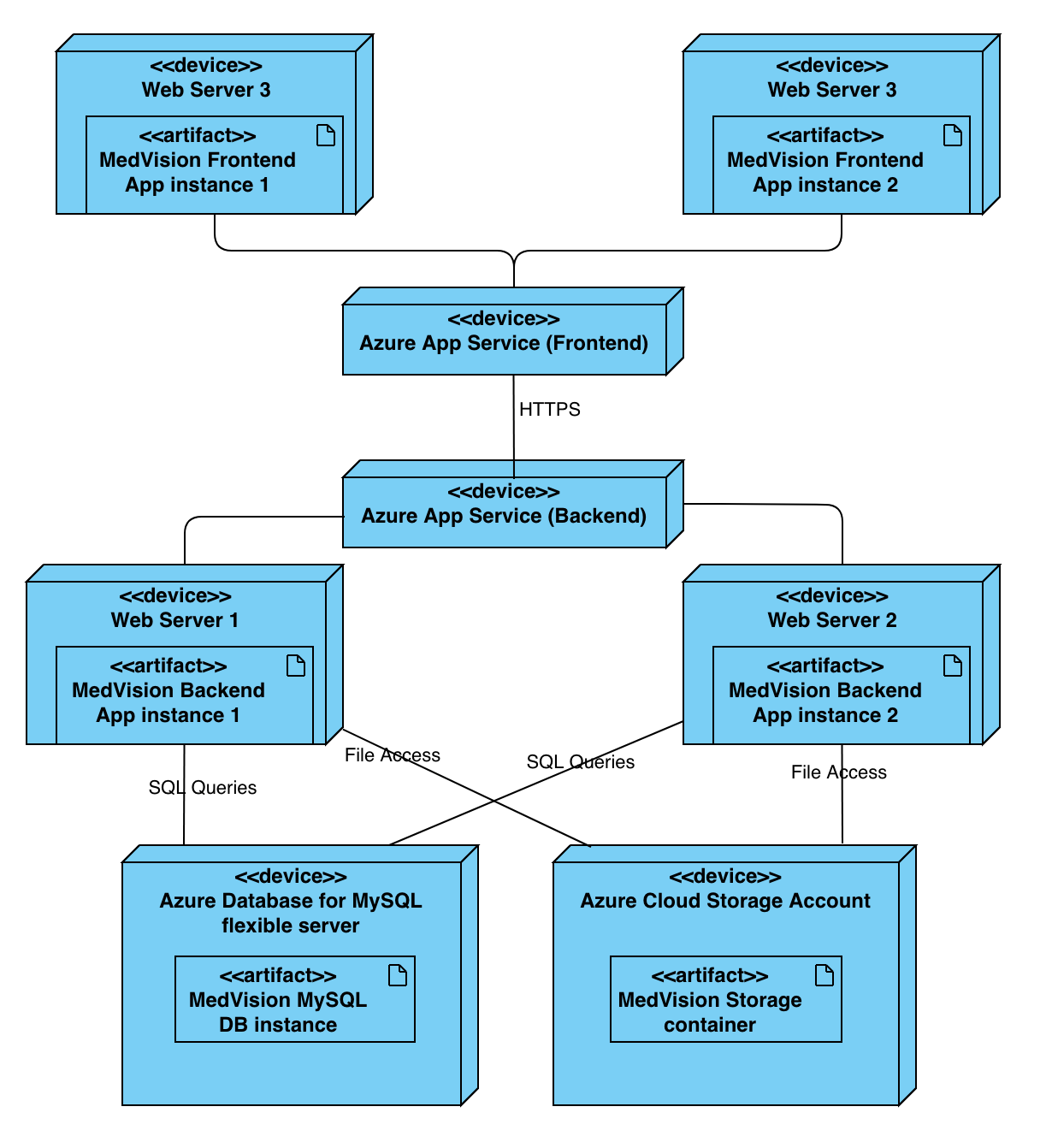


Рисунок 4.7 – MedVision Deployment Diagram

## 4.2. Процес управління змінами

Усі зміни в документі SRS та в проєкті MedVision регламентуються централізованою політикою змін, яка передбачає формалізовану процедуру їх ініціювання, розгляду, впровадження та контролю. Зміни можуть ініціюватися будь-якими учасниками проєкту, зокрема розробниками, архітекторами, тестувальниками, замовниками або іншими зацікавленими сторонами. Кожна пропозиція щодо змін проходить попередній аналіз, де оцінюється її вплив на функціональні, нефункціональні та архітектурні аспекти системи. Після цього вона розглядається на відповідній технічній нараді, де приймається рішення про її доцільність та подальше впровадження.

Якщо зміну затверджено, вона включається до плану розробки, після чого відбувається її реалізація у вигляді оновлення коду, модифікації бази даних або актуалізації документації. Усі дії супроводжуються обов’язковим тестуванням на рівні модулів та інтеграції, що гарантує стабільність системи після внесення змін. Після реалізації оновлень проводиться відповідне документування, включаючи оновлення специфікації вимог (SRS), журналу змін (Revision History) та створення або коригування тестових сценаріїв. Зміни інтегруються в основну гілку системи контролю версій Git із присвоєнням відповідного номера версії та зазначенням усіх модифікованих компонентів.

Контроль версій є ключовим аспектом управління змінами. Усі редакції документації та коду фіксуються із зазначенням дати, автора змін та опису внесених правок. Це дозволяє забезпечити зворотну сумісність, відстеження еволюції вимог та, у разі потреби, можливість відкату до попередніх стабільних станів. Для забезпечення якості процесу щомісяця проводиться аудит змін, під час якого відповідальні особи аналізують повноту реалізації, відповідність затвердженим специфікаціям та актуальність внесених правок. Результати аудитів включаються до технічної документації як підтвердження цілісності та контрольованості життєвого циклу продукту.