Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Кафедра Радіоелектронних та біомедичних комп'ютеризованих   
засобів і технологій (№ 502)

КУРСОВА РОБОТА

**з дисципліни «Основи проектування медичних програмних засобів»**

на тему:

**«Розробка інформаційної системи «DICOM Medical Records»»**

|  |  |
| --- | --- |
| Студент 3 курсу групи 522ст | Керівник курсової роботи |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (посада керівника) |
| Потапова Варвара Олегівна | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (прізвище, ім’я та по-батькові студента) | (прізвище, ім’я та по-батькові керівника) |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (резолюція «До захисту») |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис студента) | (дата) (підпис викладача) |

Харків, 2022 рік

Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Кафедра Радіоелектронних та біомедичних комп'ютеризованих   
засобів і технологій (№ 502)

дисципліна «Основи проектування медичних програмних засобів»

Курс 3 Група 522ст Семестр 4

**ЗАВДАННЯ**

**на курсову роботу**

Потапова Варвара Олегівна

(прізвище, ім’я, по батькові)

1. Тема: **«Розробка інформаційної системи «DICOM Medical Records»»**

2. Строк здачі «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

1. Вихідні дані:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Зміст пояснювальної записки:

Титульний аркуш.

Завдання на курсову.

Зміст.

Вступ.

Розділ 1. Аналіз існуючого програмного забезпечення та аналогів.

1.1 Порівняльний аналіз існуючого програмного забезпечення

1.2 Уточнення постановки завдання.

Розділ 2. Проектні і технічні рішення. Види забезпечення.

2.1 Інформаційне забезпечення.

2.2 Математичне забезпечення

2.3 Програмне забезпечення Висновки.

2 4 Перелік посилань.

Додаток А. Опис програмного забезпечення (програмний код)

Додаток Б. Інструкція користувача та адміністратора

1. Дата видачі завдання «\_\_\_» 2022 р.

Зміст

Contents

[1. Аналіз існуючого програмного забезпечення та аналогів. 6](#_Toc104736139)

[1.1 Порівняльний аналіз існуючого програмного забезпечення 6](#_Toc104736140)

[1.1.1 **Navegatium DICOM Viewer** 7](#_Toc104736141)

[1.1.2 **Sante DICOM Viewer** 8](#_Toc104736142)

[1.1.3 **MicroDicom** 9](#_Toc104736143)

[1.1.4 **Horos DICOM Project** 9](#_Toc104736144)

[1.1.5 **JiveX DICOM Viewer** 10](#_Toc104736145)

[1.1.6 **AMIDE** 11](#_Toc104736146)

[1.1.7 **postDICOM** 12](#_Toc104736147)

[1.1.8 **SMILI DICOM viewer** 12](#_Toc104736148)

[1.1.9 **Aeskulap** 13](#_Toc104736149)

[1.1.10 **Athena DICOM Viewer** 13](#_Toc104736150)

[1.2 Основнi функцiї програм для перегляду DICOM 14](#_Toc104736151)

[1.2 Уточнення постановки завдання. 15](#_Toc104736152)

[1.2.5 Вимоги до функцій, що виконуються системою 23](#_Toc104736153)

[Розділ 2. Проектні і технічні рішення. Види забезпечення. 25](#_Toc104736154)

[2.1 Інформаційне забезпечення 25](#_Toc104736155)

[2.3 Математичне забезпечення 28](#_Toc104736156)

[2.3.1 Дослідження структури зберігання даних у форматі DICOM 28](#_Toc104736157)

[2.3.2 Дослідження математичних моделей подання візуальних даних 31](#_Toc104736158)

[2.3.3 Математическая модель растрового изображения 35](#_Toc104736159)

[Особливості виконання схем алгоритмів 37](#_Toc104736160)

[2.3.5 Розробка програмного продукту 43](#_Toc104736161)

[2.3.6 Текст програми 43](#_Toc104736162)

[2.3.7. Програма і методика випробувань 43](#_Toc104736163)

[2.3.8 Висновки 43](#_Toc104736164)

[2.3.9 Інструкція користувачу 43](#_Toc104736165)

Вступ

Наразі використання DICOM формату для зберігання медичної інформації є актуально, адже він може зберігати медичну інформацію, таку як зображення ультразвуку та магнітно-резонансної карти, а також інформацію пацієнта, все в одному файлі. Формат гарантує, що всі дані залишаться разом, а також забезпечує можливість передачі вказаної інформації між пристроями, що підтримують формат DICOM. Також DICOM спирається на ISO-стандарт OSI, підтримується основними виробниками медичного обладнання та медичного програмного забезпечення.

**Метою** данної курсової роботи є розробка прошрамного продукту, що функціонує як медична картка досліджень пацієнта у медичному закладі із використанням обладнання, що генерує знімки у форматі DICOM для подальшого їх зберігання у системі.

**Актуальність теми дослідження** полягає у обмеженості готових рішень для досягнення поставленої мети.

**Об'єктом дослідження** є процес отримання та обробки знімків медичних досліджень

**Суб'єктом дослідження** є реалізація оптимізованого алгоритму отримання та обробки знімків медичних досліджень.

**Предметом дослідження** є оптимізація процесу отримання та обробки знімків медичних досліджень.

**Мета дослідження:** оптимізація алгоритму отримання та обробки медичні дослідження DICOM для дэсктопних пристроїв.

**Область дослідження:** алгоритми обробки графічних даних.

Гіпотеза: підвищення швидкості отримання та обробки знімків медичних досліджень підвищить якість роботи лікарів, при виконанні наступних завдань:

* проаналізовано існуючі алгоритми обробки зображень формату DICOM;
* визначено основні вимоги до математичної моделі обробки досліджень та, заснованому на ній, технічному рішенню;
* оптимізовано процес отримання медичних досліджень;
* вивчені авторитетні джерела у досліджуваній галузі;
* складено необхідні математичні моделі обробки медичні дослідження;
* обрано оптимальний варіант вирішення проблеми, реалізовано відповідне програмне забезпечення.

Завдання дослідження:

1. провести аналіз існуючих підходів систем обробки інформації у закладах охорони здоров'я;

2. провести аналіз існуючих програмних рішень для формалізації вимог до дослідження;

3. оптимізувати алгоритм отримання медичних досліджень;

4. розробити математичну модель перетворення графічної інформації згідно параметрів центру та ширини вікна денситометричних показників;

5. реалізувати розроблену математичну модель з метою аналізу та тестування у порівнянні з існуючими аналогами.

# Аналіз існуючого програмного забезпечення та аналогів.

## 1.1 Порівняльний аналіз існуючого програмного забезпечення

DICOM (Digital Imaging and Communications in Medicine) — це стандартний формат, який дозволяє медичним працiвникам переглядати, зберiгати та обмiнюватися медичними зображеннями незалежно вiд їх географiчного розташування чи пристроїв, якими вони користуються, якщо цi пристрої пiдтримують даний формат.

Зображення DICOM потрiбно переглядати за допомогою спецiального програмного забезпечення пiд назвою переглядачiв DICOM(DICOM Viewers), яке може читати та вiдображати формат. Зображення разом iз вiдповiдними даними про пацiєнтiв часто зберiгаються у базi даних — Системi архiвацiї та зв’язку зображень (PACS).

Основна функцiя програми для перегляду DICOM-файлу — зберiгати в PACS iнформацiю про результати обстеження (вiзуалiзацiї) разом iз деталями пацiєнта, а потiм, коли це потрiбно, переглядати та iнтерпретувати (та, можливо, редагувати) медичнi зображення, отриманi з PACS. DICOM-зображення унiкальнi тим, що крiм даних про зображення вони мiстять iнформацiю про пацiєнтiв.

Програми для перегляду DICOM-файлiв часто розробляються

з акцентом на однiй або декiлькох з наступних функцiй:

− простий перегляд медичних зображень;

− навчання (медична морфологiя);

− мiнi-сервери PACS;

− дослiдження.

Наприклад, деякi програми призначенi лише для базового перегляду. Тому вони не мають додаткових функцiй, таких як редагування або аналiз зображення. Деякi програми мають можливiсть експортувати данi у формати JPEG або GIF, якi можна використовувати в навчаннi та презентацiях. Програмне забезпечення DICOM для клiнiк може певною мiрою зберiгати зображення на серверах mini-PACS. Деякi програми також пропонують розширенi функцiї, наприклад, анонiмiзацiю, що особливо корисно при проведеннi клiнiчних дослiджень.

### 1.1.1 **Navegatium DICOM Viewer**

Дуже простий у використаннi, Navegatium має абсолютно iнтуїтивний iнтерфейс сенсорного екрану, прекрасно працює на планшетах, ноутбуках та ПК з Windows 10 та Windows 8.1. Може бути встановлений безпосередньо з Windows Store. Програма зарекомендувала себе як один з найкращих безкоштовних та вiдкритих PACSклiєнтiв та робочих DICOM-станцiй через її простоту та сукупнiсть виконуваних функцiй. Програма пiдтримує всi модальностi, реконструкцiю 3D, вiзуалiзацiю об’єму (в тому числi i можливiсть масштабування та обертання), 3D-друк органiв (з навчальною метою), перефарбовавання DICOM-зображень та змiну їх яскравостi/контрастностi.

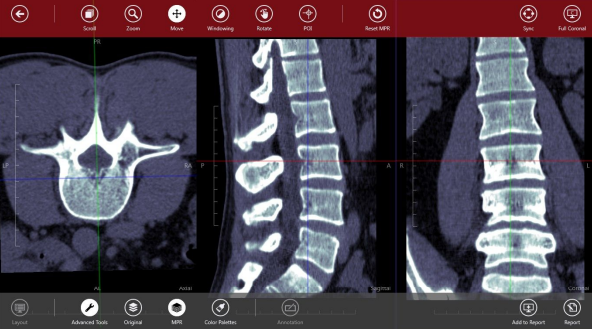
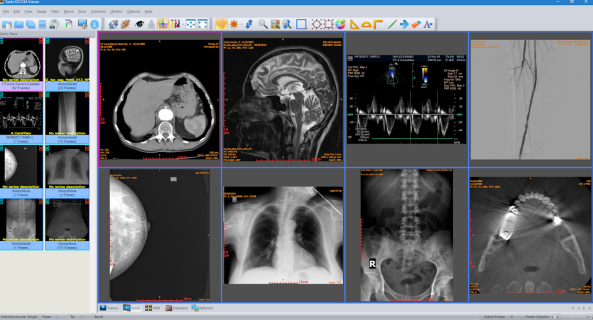


Рис. 1 – Navegatium DICOM viewer

### 1.1.2 **Sante DICOM Viewer**

Sante DICOM Viewer Free можна використовувати як окремий настiльний додаток, але вiн також пiдходить для створення DICOM CD/DVD. Вiн працює з CD/DVD без установки, не має додаткових програмних вимог, таких як .NET або Java-бiблiотеки, i автоматично вiдкриває файл DICOMDIR у коренi CD/DVD. Програма пiдтримує всi модальностi (CT, MR, US, CR, NM, XA, MG, DX тощо), всiх виробникiв та всi типи файлiв DICOM(DICOM 3.0 / NEMA 2). Працює лише на ОС Windows 7 та вище.



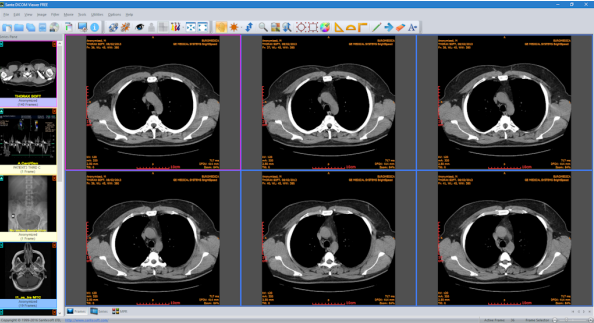


Рис. 2 – Sante DICOM Viewer

### 1.1.3 **MicroDicom**

MicroDicom — це програма для первинної обробки та збереження медичних зображень у форматi DICOM. Вона оснащений найпоширенiшими iнструментами для манiпулювання зображеннями DICOM та має iнтуїтивний iнтерфейс користувача. MicroDicom безкоштовний для некомерцiйного використання. Робоча ОС — Windows. Програма також може бути використана як PACS-клiєнт та/або DICOM-станцiя.

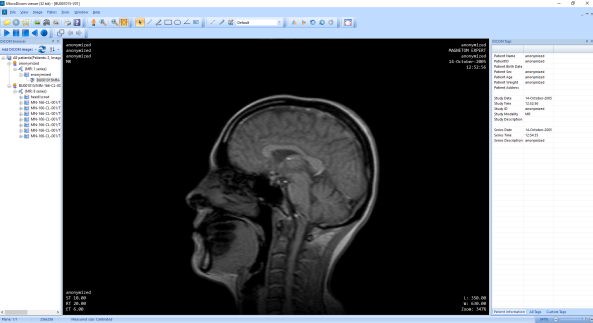


Рис. 3 – MicroDICOM

### 1.1.4 **Horos DICOM Project**

Horos — це безкоштовний переглядач медичних зображень з вiдкритим кодом. Мета проекту Horos — розробити повнiстю функцiональний 64-розрядний переглядач медичних зображень для Mac OS X. Horos базується на OsiriX та iнших бiблiотеках медичних зображень з вiдкритим кодом. Програма поширюється пiд лiцензiєю GNU General General Public License, версiя 3 (LGPL-3.0).

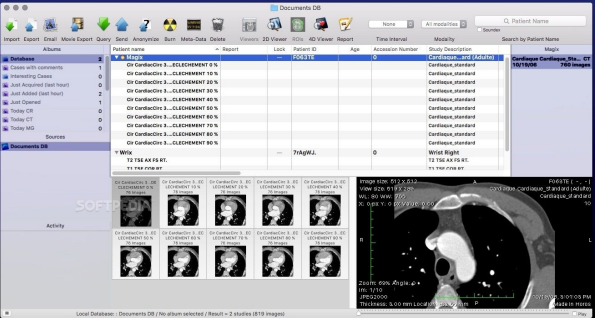


Рис. 4 – Horos DICOM Project

### 1.1.5 **JiveX DICOM Viewer**

Цей безкоштовний переглядач DICOM найкраще адаптований до

перегляду рентгенологiчних знiмкiв, але пiдтримує й iншi види медичних даних, наприклад, ЕКГ або iнформацiю у форматi PDF (у

складi DICOM-файлу).

Данi можна завантажувати на переглядач iз внутрiшньої папки

або iз зовнiшнiх носiїв iнформацiї, таких як CD, DVD або флешка.

На вiдмiну вiд лiцензованого клiєнта огляду, безкоштовний переглядач DICOM не може бути пов’язаний з PACS.



Рис. 5 – JiveX DICOM Viewer

### 1.1.6 **AMIDE**

AMIDE — безкоштовний та вiдкритий iнструмент для перегляду,

аналiзу та реєстрацiї об’ємних наборiв даних медичних зображень.

Це одна з небагатьох програм, що може працювати на Windows,

Linux та Mac OS X.

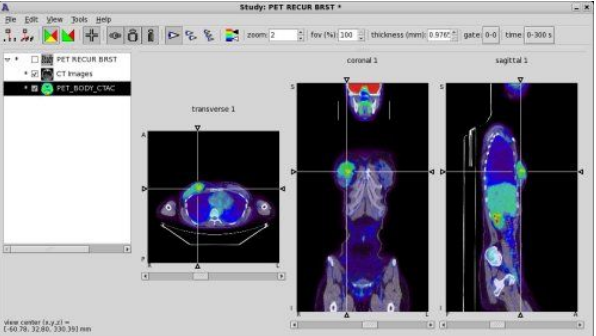


Рис. 6 – AMIDE

### 1.1.7 **postDICOM**

postDICOM — on-line-середовище для пацiєнтiв, лiкарiв, клiнiк та лiкарень. Використовуючи postDICOM, користувачi можуть розмiщувати у хмарi медичнi DICOM-зображення (комп’ютерна томографiя, магнiтно-резонансна томографiя, ультразвукове дослiдження, цифрова рентгенографiя тощо) та клiнiчнi документи (сканованi документи, звiти, iсторiя пацiєнта тощо). Пiзнiше користувачi можуть переглядати i дiлитися цими документами з лiкарями та медичними групами. Завантаженi файли DICOM передаються за допомогою так званого „тонкого клiєнта“ (thin client) — переглядача DICOM, що працює в iнтернет-браузерi.

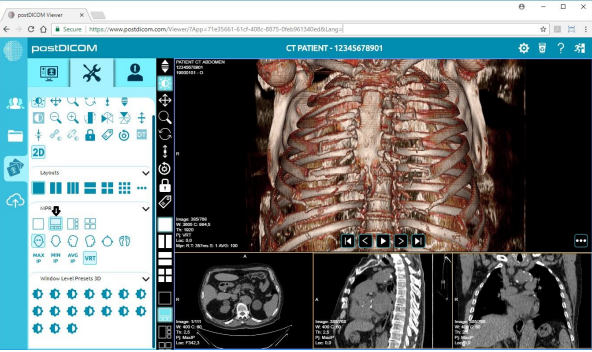


Рис. 7 – postDICOM

### 1.1.8 **SMILI DICOM viewer**

The Simple Medical Imaging Library Interface (SMILI — простий iнтерфейс бiблiотеки медичних зображень — це програма з вiдкритим кодом, легкий i простий у використаннi DICOM-переглядач та бiблiотека медичних зображень для всiх основних операцiйних систем. Основнi функцiї програми SMILI: перегляду 2D- i3D-зображень, векторних зображень, DICOM, анонiмiзацiї, аналiзу форми та моделей/поверхонь з легкими функцiями drag’n’drop. Також програма мiстить ряд стандартних алгоритмiв обробки для згладжування, встановлення порогiв, маскування тощо зображеньта моделей, як з графiчним iнтерфейсом користувача, так i/або за допомогою командного рядка. SMILI знаходиться в активному розвитку своїми розробниками.

Робочi ОС: Windows, Linux, Mac OS X

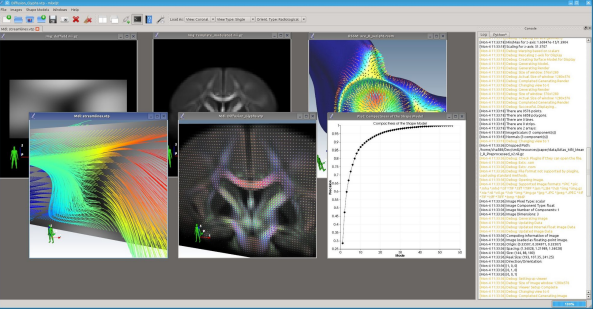


Рис. 8 – SMILI DICOM viewer

### 1.1.9 **Aeskulap**

Найпоширенiший DICOM-переглядач пiд Ubuntu/Debian Linux

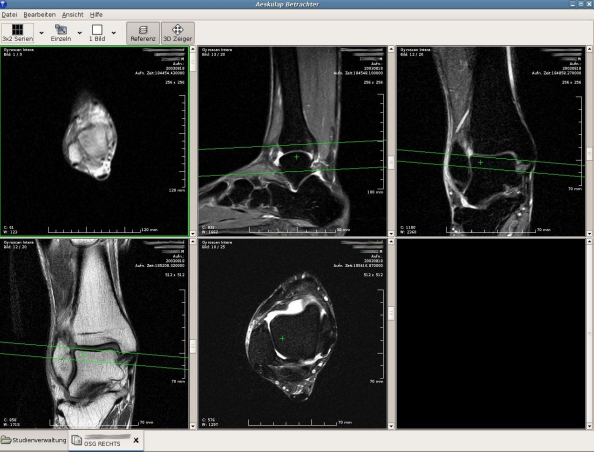


Рис. 9 – Aeskulap

### 1.1.10 **Athena DICOM Viewer**

Athena — це програмне забезпечення DICOM, призначене для вiдображення зображень DICOM на машинах з ОС Windows. Програма доступна у Windows Store i, крiм перегляду DICOM-зображень, має додатковi функцiї. Athena має дуже простий дизайн, спецiально створений для сенсорних екранiв, включаючи планшети Windows та сенсорнi екрани з великою роздiльною здатнiстю. Базова версiя Athena Lite випускається безкоштовно з можливiстю платного апгрейду до версiї Athena Pro.

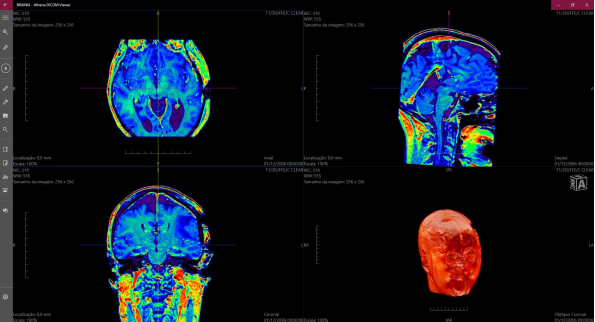


Рис. 10 – Athena DICOM Viewer

## 1.2 Основнi функцiї програм для перегляду DICOM

Важливо вiдзначити, що згаданi десять DICOM-переглядачi далеко не вичерпують перелiк всiх програм, що вмiють працювати з .dcm-файлами. Бiльше того, рокiв через 5-6 цей огляд може виявитися неактуальним. Проте, уважно розглядаючи скрiншоти, можна зробити наступний висновок про основний функцiонал DICOM переглядачiв:

− можливiсть синтезу 3D-зображення з окремих зрiзiв (слайсiв);

− можливiсть вiзуалiзацiї об’єму, тобто манiпуляцiй з 3D-зображеннями при збереженнi їх форми (масштабування та обертання);

− можливiсть змiни яскравостi/контрастностi як зображення в цiлому, так i його частин;

− можливiсть сегментацiї зображення, тобто розбиття його на частини або видiлення окремих частин;

− можливiсть проведення вимiрювання вiдстаней мiж рiзними елементами зображення в натуральних одиницях вимiрювання (мм або дюймах)

— це важливо при плануваннi операцiй;

− можливiсть сумiщення зображень, отриманих рiзними методами (наприклад, КТ та МРТ, або ПЕТ та ОФЕКТ тощо) — iнколи це дає можливiсть помiтити дуже дрiбнi деталi, якi можуть виявитися важливими при постановцi дiагнозу.

## 1.2 Уточнення постановки завдання.

**1.2.1 Загальні відомості**

Повне найменування системи – медична інформаційна система «Dicom Medical Records».

Скорочене найменування системи – МІС «Dicom Medical Records».

Нижче у даному документі під Системою розуміється МІС «Dicom Medical Records».

Підприємство-Замовник – Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут».

*Документи, на базі яких створюється Система*

Система створюється на базі снаступних документів:

- технічне завдання на створення медичної інформаційної системи «Dicom Medical Records».

*Планові терміни початку і завершення робіт*

01/0/2022-01/06/2022

**1.2.2 Призначення і цілі розвитку Системи**

*Призначення системи*

Система призначена для автоматизації опрацювання медичних даних пацієнтів та DICOM файлів у медичних закладах.

Основною ціллю є підвищення єфективності опрацювання медичних даних у медичних закладах.

**1.2.3 Х****арактеристики об’єкта автоматизації**

Об’єктом автоматизації в рамках даного проєкту є маніпулювання даними пацієнтів а також файлами, що супроводжують текстову інформацію про пацієнта.

**1.2.4** **Загальні відомості про систему**

1.2.4.4.1 Вимоги про структуру та функціонування Системи

1.2.4.1.1 Перелік підсистем, їх призначення та основні характеристики, вимоги до рівнів ієрархії та ступеня централізації системи

| **Номер** | **Підсистема** | **Призначення і основні характеристики** |
| --- | --- | --- |
|  | Управління потоком інформації про пацієнтів | Підсистема забезпечує інформаційну підтримку управління потоками інформації про пацієнтів медичного закладу. |
|  | Зберігання лікувально-діагностичної інформації про пацієнта | Підсистема забезпечує зберігання інформації про стан/стани пацієнта, що вносяться в систему протягом всього процесу лікування. |
|  | Адміністрування лікувально-діагностичною інформацією про пацієнта | Підсистема забезпечує адміністрування лікувально-діагностичною інформацією про пацієнта у медичному закладі. |
|  | Підсистема інтеграції з медичним обладнанням | Забезпечує обмін даними між медичною інформаційною системою та діагностичним обладнанням |

Для ефективної взаємодії між собою всі підсистеми МІС повинні функціонувати в єдиному інформаційному просторі та мати єдину базу даних.

Система має бути інтегрована, сумісна і функціонувати у взаємодії з сучасними томографами (МРТ, КТ).

1.2.4.1.2 Вимоги до характеристик взаємозв'язків створюваної Системи із суміжними системами

Система має забезпечувати можливість інформаційного обміну із суміжними системами.

Передача даних із суміжних систем у Систему і назад має проводитися за затвердженим регламентом та структурою обміну. Структура файлів обміну та регламент передачі даних має бути розроблений, узгоджений та затверджений відповідальними особами Замовника та Виконавця.

1.2.4.1.3 Вимоги до режимів функціонування системи

Система призначена для постійної, щоденної роботи операторами системи(медсестра, лікар, адміністратор медичного закладу).

Опреатори системи працюють у of-line режимі при підключеному напряму (через фізичне підключення) апараті томографа або в on-line при підключенні через бездротові технології (Wi-Fi). База даних розгортається локально на працюючому сервері і за необхідності синхронизується із глобальною базою даних.

Система призначена для постійної, безперебійної, щоденної работи операторів системи.

Кількість одночасних сесій системи визначається медичним закладом і задається у конфігурації розробленого програмного продукту.

Сервер бази даних Системи повинен працювати в безперервному цілодобовому режимі, крім періодів проведення регламентних робіт з копіювання даних системи, проведення регламентних ремонтних або відновлювальних робіт.

1.2.4.1.4 Вимоги щодо діагностування Системи

Діагностика здійснюється як внутрішньосистемним механізмом, що забезпечує збір інформації про помилки, що виникають, так і забезпечуваної СУБД згідно з посібником з експлуатації СУБД.

1.2.4.1.5 Допустимі межі модернізації та розвитку Системи

Після введення Системи в промислову експлуатацію модернізація та розвиток Системи, пов'язані з розширенням функціональних можливостей або суттєвою зміною чинного законодавства, повинні виконуватись, залежно від складності, або за окремим договором із залученням розробника компонента Системи, або самостійно попередньо підготовленою спеціалізованою службою Замовника.

Переналаштування Системи, пов'язане зі зміною кількості та складу інформації, має здійснюватись службою супроводу Замовника.

1.2.4.2 Показники призначення

Система має забезпечувати:

- облік особливостей галузевої специфіки підприємства;

- відповідність вимогам галузевих нормативних документів відповідно до вимог цього Технічного завдання;

- можливість налаштування для забезпечення автоматизації бізнес-процесів Замовника у разі їх змін, а також внесення змін до налаштувань структури організації Замовника у Системі;

- стійкість до помилок користувачів.

1.2.4.4.3 Вимоги до надійності технічних засобів, програмного та інформаційного забезпечення

1.2.4.4.3.1 Перелік аварійних ситуацій, за якими мають бути регламентовані вимоги до надійності технічних засобів та програмного забезпечення

Повинні бути регламентовані наступні максимальні показники з відмов компонентів ЛВС ЛПЗ:

- час заміни активного мережного устаткування;

- час відновлення монтажних компонентів;

- час усунення несправностей у роботі мережного устаткування ЛОМ;

- час відновлення роботи ЛОМ після збоїв з енергоживлення;

- час відновлення енергоживлення компонентів ЛВС при аваріях.

Щодо сервера Системи та робочих станцій повинні бути визначені такі максимальні показники, що визначають надійність та час відновлення їх компонентів у разі збоїв або відмов:

- час усунення несправностей у роботі обладнання сервера Системи;

- час усунення несправностей у роботі ПЗ сервера Системи;

- час усунення несправностей у роботі SQL сервера БД Системи;

- час усунення несправностей у роботі устаткування робочих станцій;

- час усунення збоїв у роботі системного та прикладного ПЗ робочих станцій.

1.2.4.4.3.2 Вимоги щодо надійності технічних засобів

Надійність технічних засобів забезпечується використанням сертифікованих засобів обчислювальної техніки, їх комплектуючих та засобів передачі даних.

1.2.4.4.3.3 Вимоги щодо надійності програмного забезпечення

Надійність ПЗ забезпечується використанням сертифікованих операційних систем, загальносистемних програмних засобів та інструментальних програмних систем, що використовуються при розробці прикладного ПЗ. Надійність прикладного ПЗ забезпечується комплексом заходів, що здійснюють управління якістю створення ПЗ на всіх етапах життєвого циклу.

Прикладні програми повинні мати захист від некоректних дій користувачів та помилкових вихідних даних.

Прикладні програми не повинні модифікувати свій код або коди інших програм під час роботи.

1.2.4.4.3.4 Вимоги щодо надійності інформаційного забезпечення

Надійність функціонування компонентів ІЗ забезпечується використанням сертифікованих систем управління базами даних, засобів прийому, обробки та передачі даних.

Повнота та несуперечність даних Системи забезпечується використанням єдиної системи класифікації та кодування, уніфікованої системи документів, концептуальної та логічної моделі даних, що описує єдиний інформаційний простір Системи.

Кошти введення даних у систему повинні забезпечувати контроль правильності даних щодо їх типу.

При модифікації та видаленні даних, засоби ведення ІС повинні вимагати підтвердження правильності виданих команд. Має бути забезпечена можливість дублювання, архівування та збереження будь-якої кількості резервних копій баз даних та їх фрагментів.

Технологічні компоненти ІЗ повинні забезпечувати контроль цілісності структур баз даних, порушення якої можливе після апаратних збоїв.

1.2.4.4.4 Вимоги щодо безпеки

Прикладні програми повинні мати захист від некоректних дій користувачів та помилкових вихідних даних.

Прикладні програми не повинні модифікувати свій код або коди інших програм під час роботи.

При виникненні виключних ситуацій Система має документувати причини/логи у СУБД.

1.2.4.4.5 Вимоги щодо ергономіки та технічної естетики

Система повинна враховувати сучасні вимоги щодо ергономіки та технічної естетики. До таких вимог належать такі:

- Забезпечення комфорту та зручності роботи користувача;

- інтуїтивно зрозумілий інтерфейс;

- виконання подібних функцій подібними способами;

- Використання графічного інтерфейсу користувача.

Для реалізації цих вимог Система має бути реалізована з урахуванням принципів та положень, викладених нижче.

Діалоговий інтерфейс програм Системи має будуватися на основі віконного інтерфейсу операційної системи Microsoft Windows. При цьому програми Системи максимально використовують термінологію, способи організації діалогу, призначення клавіш клавіатури і кнопок миші, прийняті в Microsoft Windows.

Програми Системи повинні переважно використовувати колірну гаму, що забезпечує можливість багатогодинної роботи користувача з квітами, що мінімально втомлюють зір.

Вимоги щодо ергономіки та технічної естетики мають бути уточнені на етапі технічного проектування.

1.2.4.4.6 Вимоги щодо захисту інформації від несанкціонованого доступу

Захист інформації від несанкціонованого доступу в Системі здійснюється засобами операційної системи, СУБД та прикладного програмного забезпечення.

На рівні користувача в Системі повинні бути передбачені такі засоби захисту інформації:

- авторизований вхід до системи – доступ користувача до системи повинен здійснюватися після введення ним коректних ідентифікаторів та паролів. Ідентифікатор та пароль користувачеві призначається адміністратором системи;

- мають бути реалізовані засоби розмежування доступу користувача до інформації;

- мають бути реалізовані засоби розмежування доступу користувача до функцій системи;

- мають бути реалізовані засоби протоколювання дій користувачів щодо зміни в електронних медичних записах пацієнтів;

1.2.4.4.7 Вимоги щодо збереження інформації при аваріях

Система повинна забезпечувати збереження інформації при збоях технічних засобів або помилках користувачів.

У Системі має бути забезпечена підтримка логічної цілісності та узгодженості даних.

1.2.4.4.7.1 Перелік можливих аварійних ситуацій із зазначенням вимог до засобів відновлення працездатності Системи

Збій загального або спеціального програмного забезпечення (окремого АРМ або сервера).

Після збою серверної, клієнтської операційних систем або СУБД у процесі виконання завдань користувача має бути забезпечене відновлення даних у базі даних до стану, що існував на момент закінчення останньої нормально завершеної перед збоєм транзакції.

1.2.4.4.8 Вимоги щодо захисту від впливу зовнішніх впливів

Для функціонування програмного комплексу необхідно дотримання всіх вимог та правил експлуатації комп'ютерної техніки. Додаткових вимог та обмежень не вводиться.

1.2.4.4.9 Вимоги до патентного захисту

Програмні та технічні засоби, що купуються у сторонніх фірм та підприємств, повинні супроводжуватися документацією, що підтверджує правомочність цих організацій постачати цю продукцію та супроводжуватися ліцензійною угодою.

1.2.4.4.10 Вимоги щодо стандартизації та уніфікації

1.2.4.4.10.1 Використання стандартних, уніфікованих методів реалізації функції системи

Система, побудована в архітектурі "клієнт-сервер", повинна використовувати засоби цієї СУБД для реалізації функцій системи, що виконуються на сервері БД.

У клієнтській частині Система використовує операційну систему MS Windows 10. При реалізації функцій Системи максимально повинні використовуватися засоби цих систем.

Програмні засоби Системи повинні відповідати угодам і стандартам, що є у використовуваній СУБД, та операційній системі в частині роботи з клавіатурою, відображення інформації на екрані, виклику довідкової інформації, організації інтерфейсу користувача тощо.

Для виключення надмірності технологічних процедур під час виконання функцій системи слід однаково реалізувати загальні всім функцій процедури.

1.2.4.4.10.2 Використання типових проектних рішень

Проектні рішення Системи під час виконання різних функцій повинні забезпечувати:

- дотримання єдиних правил організації інтерфейсу користувача;

- однакову реакцію системи на неправильні дії користувачів;

- використання фіксованого переліку термінів та визначень системи при організації діалогу та формуванні екранів;

- типовий підхід до розмежування доступу користувачів до інформації;

- максимальне використання засобів, що є в інструментальних засобах системи (базові бібліотеки процедур та функцій, динамічні бібліотеки, елементи інтерфейсу тощо).

1.2.4.4.10.3 Вимоги до використання типових компонент та комплексів

ПЗ Системи повинні використовувати об'єктно-орієнтований і модульний принцип побудови програмної системи з використанням типових програмних компонент, що реалізують одні й ті самі функції (фрагменти функцій) системи.

Вимоги до стандартизації та уніфікації мають бути уточнені на етапі технічного проектування.

1.2.4.4.11 Додаткові вимоги

Програмний комплекс повинен забезпечувати стійке функціонування в мережі з одночасною роботою з інформаційною системою до 50, 100, 200 користувачів залежно від характеристик сервера.

### 1.2.5 Вимоги до функцій, що виконуються системою

У розділі описано ключові вимоги до можливостей конфігурації системи.

1.2.5.4.1 Функція «Реєстрація пацієнтів»:

- введення даних про пацієнта (ім’я та прізвище, дата народження, стать, скарги на момент реєстрації)

1.2.5.4.2 Функция «Створення запису (дослідження)»:

- відображення даних про пацієнта;

- завантаження DICOM – файлів у режимі перегляду;

- створення запису про дослідження пацієнта. Данні для введення: 1) дата запису 2) частина тіла, що задокументовано на зображені 3) нотатки оператора системи 3) файл, що буде прикладено до запису

1.2.5.4.3 Функция «Перегляд пацієнтів»:

- перегляд таблиці пацієнтів медичного закладу;

- забезпечити навігацію до вікна «Створення запису (дослідження)»

1.2.5.4.4Функція «Імпорт/експорт зображень»:

- функція має забезпечити імпорт/експорт зображень у форматі DICOM;

- функція має забезпечити імпорт/експорт зображень у форматах JPEG, BMP, TIFF, GIFF.

1.2.5.4.5 Функція «Робота із зображеннями»:

- має забезпечити перегляд на екрані одиничних зображень;

- має забезпечити проведення вимірювань;

- має надавати інструмент «Point» для роботи із зображеннями;

- має надавати інструмент «Circle» для роботи із зображеннями;

- має надавати інструмент «Polyline» для роботи із зображеннями;

- має надавати інструмент «Interpolated» для роботи із зображеннями;

- має надавати інструмент «Rectangle» для роботи із зображеннями;

- має надавати інструмент «Ellipse» для роботи із зображеннями;

- має надавати інструмент «Multiline» для роботи із зображеннями;

- має надавати інструмент «Rangaline» для роботи із зображеннями;

- має надавати інструмент «Infiniteline» для роботи із зображеннями;

- має надавати інструмент «Arrow» для роботи із зображеннями;

- має надавати інструмент «Axis» для роботи із зображеннями;

- має надавати інструмент «Ruler» для роботи із зображеннями;

- має надавати інструмент «Crosshair» для роботи із зображеннями;

- має надавати інструмент «Text» для роботи із зображеннями;

- має надавати інструмент «Збільшити» для роботи із зображеннями;

- має надавати інструмент «Зменшити» для роботи із зображеннями;

- повинна забезпечити відображення координат курсору на зображенні та значення інтенсивності у точці курсору;

- має забезпечити перетворення інтенсивності зображення;

- має забезпечити виділення областей інтересу на зображенні;

- повинна забезпечити накладання шкал та масштабних сіток.

# Розділ 2. Проектні і технічні рішення. Види забезпечення.

## 2.1 Інформаційне забезпечення

**Діаграма прецедентів.**

Діаграма прецедентів дозволить визначити функціональну структуру системи, не заглиблюючись у деталі її реалізації.

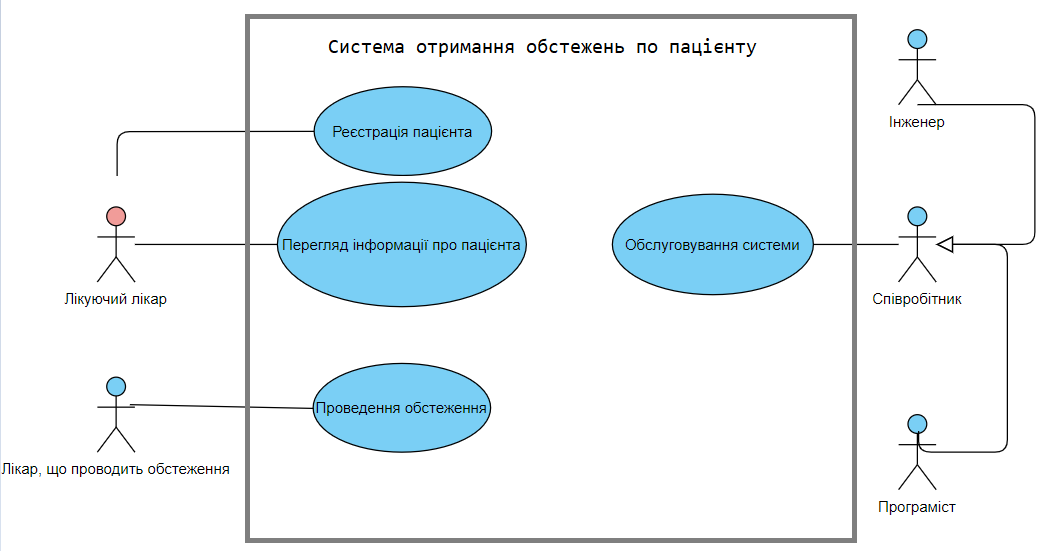


Рисунок 2.1.1. Діаграма прецендентів

Таблица 1.2.1 – Прецедент: Проведення обстеження

|  |
| --- |
| Прецедент: проведення обстеження |
| ID:1 |
| Опис: Проведення обстеження пацієнта з використанням медичного обладнання |
| Головні актори: Лікар, що проводить обстеження (лікуючий лікар) |
| Другорядні актори: Ні |
| Передумови: Прецидент розпочинається за ініціативи користувача |
| Основний поток:   1. Користувач проводить обстеження пацієнта за допомогою медичного обладнання. 2. Данні про обстеження передаються з застосунку в базу даних інформаційної системи. |
| Постумова: обстеження проведено. |
| Альтернативні потоки: Ні |
|  |

Таблица 1.2.2 – Прецедент: Реєстрація користувача

|  |
| --- |
| Прецедент: проведення обстеження |
| ID:2 |
| Опис: Реєстрація нового користувача у системі |
| Головні актори: Лікуючий лікар |
| Другорядні актори: Ні |
| Передумови: Прецидент розпочинається за ініціативи користувача |
| Основний поток:   1. Користувач вводить данні про нового пацієнта. 2. Данні про пацієнта передаються з застосунку в базу даних інформаційної системи. |
| Постумова: пацієнта зареєстровано. |
| Альтернативні потоки: Ні |
|  |

Таблица 1.2.3 – Прецедент: Просмотр сведений по пациенту

|  |
| --- |
| Прецедент: проведення обстеження |
| ID:3 |
| Опис: Перегляд інформації про пацієнта |
| Головні актори: Лікар, що проводить обстеження (лікуючий лікар) |
| Другорядні актори: Ні |
| Передумови: Прецидент розпочинається за ініціативи користувача |
| Основний поток:   1. Пошук необхідного пацієнта. 2. Пошук необхідного обстеження. |
| Постумова: Лікар отримав данні про обстеження пацієнта. |
| Альтернативні потоки: Ні |
|  |

**Діаграма послідовності**

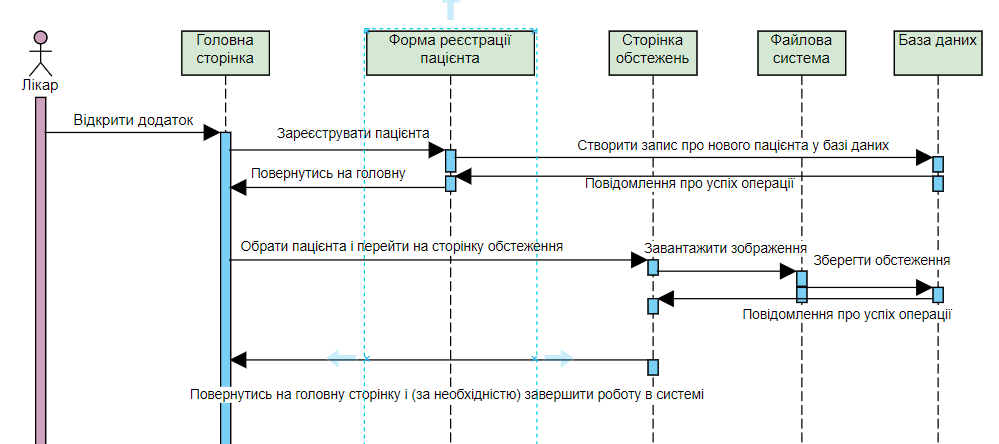


Рисунок 2.1.2. Діаграма послідовностей (Sequence diagram)

В результаті на даному етапі розробки було виділено основні етапи роботи співробітників з проведення та отримання доступу до досліджень з використання медичної автоматизованої інформаційної системи. Етапи розглядалися за принципом «чорної скриньки», це дозволило визначити послідовність роботи працівників із системою. В цьому випадку можна виділити найважливіші етапи – проведення дослідження, перегляд інформації про пацієнта та обслуговування системи.

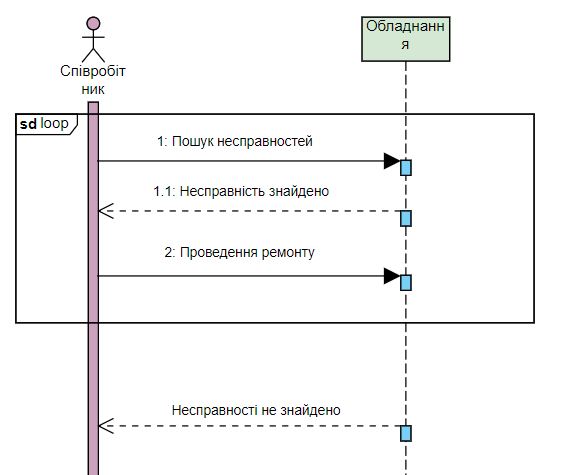


Рисунок 1.5.3 – Диаграмма по прецеденту – Обслуживание системы

Діаграмма классов

## 2.3 Математичне забезпечення

### 2.3.1 Дослідження структури зберігання даних у форматі DICOM

Для реалізації можливості керування параметрами регулювання атрибутів, що відповідають за відображення щільності тканин, необхідно проаналізувати структуру формату зберігання даних медичних досліджень формату DICOM. DICOM, як протокол, дозволяє зберігати результати передачі в однойменному форматі. Стандартна структура такого файлу представлена на малюнку 2.3.1.1

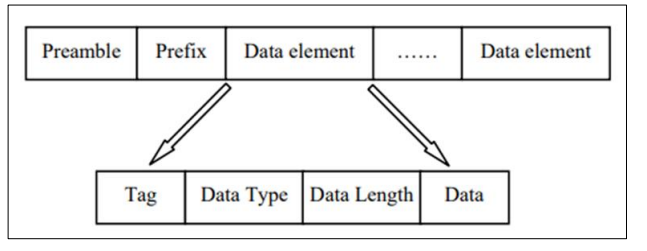


Рисунок 2.3.1.1 – Файлова структура формату DICOM

1. Заголовок файлу - складається з вступної частини та префіксу;

* Вступна частина (англ. preamble) - 128 байт завдовжки(використовується для доступу до зображень та інших даних у файлі DICOM, забезпечуючи сумісність із зазвичай використовуваними форматами файлів зображень);
* Префікс (англ. prefix) - 4 байти в довжину, містить рядок "DICM" верхньому регістрі;

2. Дані (англ. data element) - «зазвичай має кілька наборів елементів даних (кожен елемент даних, що відповідає атрибуту IOD, має чотири домени, а саме: тег, уявлення значення, довжина значення та поле значення, в якому представлення значення є необов'язковим)».

Дані пікселя (7FE0, 0010) містять «необхідні дані для медичного монітора». Інші елементи даних, тісно пов'язані з елементом піксельних даних представлені нижче.

(0028, 0008): кількість кадрів;

(0028, 0010): кількість ліній зображення;

(0028, 0011): кількість стовпців зображення;

(0028, 0100): номер біта розподілу;

(0028, 0101): номер біта зберігання;

(0028, 0102): старший біт числа.

Код піксельних даних визначається номером біта розподілу, номером біта пам'яті та найвищим номером біта, а номер біта розподілу повинен бути більшим, ніж номер біта пам'яті. Дані пікселя DICOM зображення часто є 16-розрядними або 12-розрядними. Якщо обрано16-бітний формат, кожен піксель має два байти; якщо 12-біт, розподіл байтів на піксель складніший, і можна визначити номер біта розподілу, номер біта пам'яті та найвища кількість біт, розрізняючи значення елементів (0028,0100), (0028,0101) та (0028,0102).

Коли 16 розподілених біт, 12 біт пам'яті та старший біт складають по 12 у кожному пікселі, то піксель займає всього 2 байти і використовує молодший 12 біт. Коли найстарший біт – 12-й, то кожен піксел використовує старші 12 біт із 2-х байт. Коли 12 розподілених біт, 12 біт зберігання та найстарший біт – 11-й у кожному пікселі, піксель займає тільки 3 байти, зміст середнього байта розділено на 2 частини, відповідно, належать до першого та останнього байта, після цього кожен піксель має 12 біт. «Дані зображення можуть бути стиснуті або залишені без стиску. Коли дані передаються у стислому форматі, Value Representation приймає значення OB, інакше значення OW. Для стиснутих даних, послідовність зазвичай записується зверху вниз, зліва направо, дані кодуються та зберігаються як безперервний бітовий потік. Стислі дані можуть зберігатися сегментами та розділятися серією пов'язаних елементів підтримки процесу стиснення зображення невизначеної довжини».

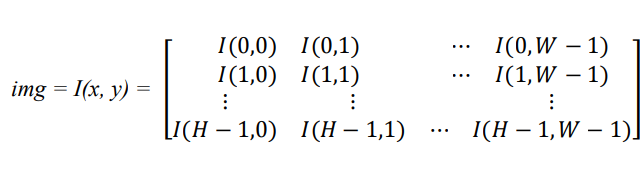
Dicom Standard Image – це спеціальний формат, який має складні типи та різні комбіновані формати». Для його відображення та обробки необхідний спеціально розроблений процесор обробки зображень, однак більшість сучасних додатків його не підтримують. «Випробування показали, що лікарі зазвичай обирають специфічні частини зображення як основу для висновків у діагнозі захворювання. Складові характеристики даних зображення DICOM мають багато спільного з поширеними форматами зображень, такими як BMP. Формат BMP, що використовує стиснення без втрат, підходить для аналізу зображень, виявлення ознак захворювання та інший необхідної інформації».

Таким чином, проведено аналіз структури даних формату DICOM, результаті якого визначено особливості структури файлу, вивчено атрибути IOD, а також атрибути при передачі та обробці зображень. Це дозволило зробити висновок про можливість використання растрового формату зображень для виявлення ознак захворювання та іншої необхідної інформації.

### 2.3.2 Дослідження математичних моделей подання візуальних даних

З математичної точки зору зображення може бути представлене в вигляді речовинної функції img від двох змінних x та y, I (x, y). У статті Штанчаєва Х.Б. зображення представлене у матричному вигляді. «Функція 𝑖𝑚𝑔 = 𝐼 (𝑥, 𝑦), зазвичай визначається у прямокутній формі, але для зручності можна уявити квадратними областями, тобто. 𝑥 ∈ [0; 𝑊], а 𝑦 ∈ [0; 𝐻] де W - ширина зображення, H - висота зображення і W = H, тобто. зображення має розмір W × H. Значення, яке стоїть на перетині x та y, називається пікселем. Не важко помітити, що зображення схоже на прямокутною системою координат. Початком координат (0, 0) стосовно зображення, треба рахувати (x, y) = (1, 1). Пара (x, y) = (1, 2) відноситься до другому пікселю».

Також зображення можна представити у матричному вигляді:

 (2.3.21)

де H-висота зображення, W-ширина зображення.

Особливістю цього представлення цифрового зображення є тому, що перший елемент у виразі (1) має координати (x, y) = (0,0), а не(x, y) = (1, 1) як зазначалося вище.

В роботі Грузмана І.С., Киричука В.С., Косих В.П., Перетягіна Г.І., Спектора А.А. зображення представляється як дискретного представлення безперервного зображення. Рідко, зображення, отримані в інформаційних системах, що мають цифрову форму, тому їх перетворення до цифрової форми – «обов'язкова операція, якщо це передбачається у поставленому завданні». Таке перетворення включає в себе дві процедури. «Перша процедура полягає у заміні безперервного кадру на дискретний (дискретизація), а друга замінює безперервний набір значень яскравості на набір квантованих значень (квантування)». «У цифровому поданні кожне з квантованих значень яскравості пов'язане з двійковим числом, чим і досягається можливість передачі зображення електронно-обчислювальну машину».

«Існують різні способи заміни безперервного зображення на дискретний. Наприклад, ви можете вибрати будь-яку систему ортогональних функцій і, обчисливши коефіцієнти представлення зображення за цим системі (базису), замінити ними зображення. Великий вибір базисів дає можливість утворення різних дискретних уявлень безперервного зображення. Однак найбільш використовуваною є періодична дискретизація, зокрема, дискретизація із прямокутним растром. Даний спосіб дискретизації можна розглядати як варіант застосування ортогонального базису, в якому як елементи використовуються зсунуті δ-функції». Далі розглянемо основні особливості прямокутної дискретизації.

Нехай 𝑥(t1,t2) – безперервне зображення, а 𝑥(𝑖1 𝑖2) відповідне йому дискретне, отримане з безперервного зображення шляхом прямокутної дискретизації. Таким чином, зв'язок між ними визначається виразом:

 (2.3.2.2)

де Δt1, Δt2 – вертикальний та горизонтальний інтервали дискретизації, відповідно. Рисунок 2.3.2.1 ілюструє розташування відліків на площині t1,t2 за прямокутної дискретизації.

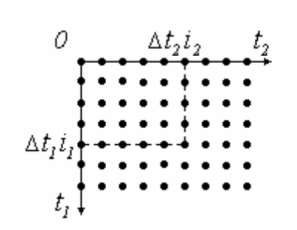


Рисунок 2.3.2.1 - Розташування відліків під час прямокутної

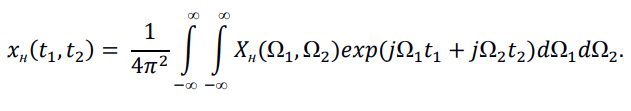
дискретизації

Основне питання при заміні безперервного зображення на дискретне полягає в тому, щоб визначити умови, за яких ця заміна є повноцінної, тобто без втрати даних, що містяться в безперервному сигнал. «Втрати відсутні, якщо можлива зворотна операція – відновлення безперервного сигналу із дискретного. З точки зору математики, питання полягає в тому, щоб провести реконструкцію безперервного сигналу у двовимірних проміжках між вузлами, у яких його значення відомі або, іншими словами, у здійсненні двовимірної інтерполяції». Відповісти на це питання можна, провівши аналіз спектральних властивостей безперервного та дискретного зображень.

Двовимірний безперервний частотний спектр Хн(Ω1, Ω2) безперервного сигналу 𝑥н(t1,t2) визначається двомірним прямим перетворенням Фур'є:

(2.3.2.3)

якому відповідає двовимірне зворотне безперервне перетворення Фур'є:

(2.3.2.4)

«Двовимірний характер зображення порівняно із звичайними сигналами включає додаткові можливості оптимізації цифрового уявлення зменшення обсягу одержуваних цифрових даних. У зв'язку з цим вивчалося питання про найкраще розміщення рівнів квантування та про використання різних растрів.

На практиці, в більшості випадків, використовують дискретизацію, засновану на застосуванні прямокутного растру, та рівномірне квантування яскравості. Це пов'язано з невеликими перевагами від використання оптимальних перетворень та простотою виконання відповідних операцій»

«При використанні прямокутного растру в остаточному вигляді цифрове зображення зазвичай являє собою матрицю, рядки та стовпці якій відповідають рядкам та стовпцям зображення».

«При цифровій обробці зображень безперервний динамічний Діапазон значень яскравості ділиться на ряд дискретних рівнів. Ця процедура називається квантуванням». Квантувач перетворить безперервну змінну x дискретну змінну xкв, що приймає кінцеве безліч значень {r1…rL} . Ці значення називаються «рівнями квантування». У загальному випадку перетворення виражається ступінчастою функцією, зображеною малюнку 2.3.2.2. Якщо яскравість x відліку зображення належить інтервалу(dj…dj+1) (тобто, коли dj < 𝑥≤ dj), то вихідний відлік замінюється на рівень квантування rj, де dj,J =

При цьому вважається, що динамічний діапазон значень яскравості обмежений і дорівнює

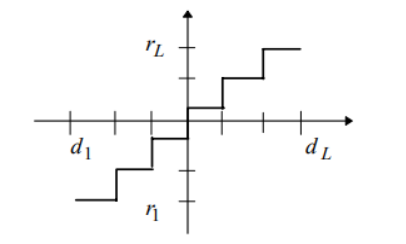
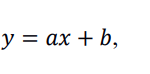


Рисунок 2.3.2.2 - Ступінчаста функція квантування

Красильников Н.М. описує растрову та векторну моделі двовимірного зображення наступним чином. «В основі растрової моделі лежить растр – матриця пікселів, які становлять інтенсивність відповідних ділянок зображення. Растрове зображення характеризується роздільною здатністю, яка визначається кількістю пікселів на одиницю довжини. Чим більше пікселів посідає одиницю довжини, тим вище Розширення". Для обробки зображення за допомогою ЕОМ, його представляють у цифровій формі. «У разі чорно-білого зображення це означає, що інтенсивність (яскравість) кожного пікселя представляється числом від 0 до 255, тобто зображення представляється у вигляді двомірної матриці, що складається зазвичай із 8-розрядних двійкових чисел». Для представлення кольорового зображення у цифровій формі використовують три матриці, кожна з яких складається з 8-розрядних двійкових чисел, рідше з 16-розрядних. При цьому елементи кожної з цих матриць представляють інтенсивність червоного, зеленого та синього компонентів кольору пікселя (модель RGB). В разі векторної моделі, на відміну від растрового зображення, основою яких є пікселі, основою векторних зображень є контури, представлені кривими, званими векторами. «Кожен контур векторного зображення є окремим об'єктом, який можна незалежно від інших редагувати. Відповідно до цього векторну графіку іноді називають об'єктно-орієнтованою графікою». Для опису відрізка прямої у векторній графіці використовують рівняння

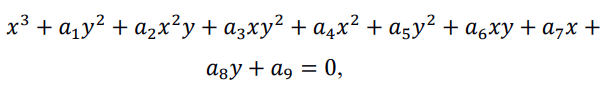
(2.3.2.5)

де x та y – декартові координати, а a і b – постійні коефіцієнти. початок і кінець відрізка задаються координатами x1 та x2. Для опису кривих другого порядку – кіл, еліпсів, парабол та гіпербол – використовують криві другого порядку

(2.3.2.6)

де a1,a2,a3,a4,a5 - Постійні коефіцієнти. Також для даної кривої необхідно задати координати, що визначають початок та кінець - x1 та x2.

На відміну від кривих другого порядку, криві третього порядку можуть мати точки перегину, що дає можливість їх використовувати для подання у векторній формі різних природних об'єктів. Рівняння для опису кривих третього порядку має такий вигляд

(2.3.2.7)

де a1,a2,a3,a4,a5,a6,a7,a8,a9 - Постійні коефіцієнти. x1 та x2 – координати, що задають початок та кінець.

Широке застосування у векторній графіці отримали криві Без’є третього порядку. «Особливістю цих кривих є те, що вони дозволяють зручно регулювати як положення вузлів на площині зображення, але також величини першої похідної лінії (кута нахилу) та її другий похідною (кривизну) у цих точках». Це «забезпечує можливість з'єднувати окремі сегменти без зламів у точках з'єднання і цим апроксимувати відрізками кривих Без'є контури будь-якої складності».

### 2.3.3 Математическая модель растрового изображения

Растрові зображення визначаються як звичайна прямокутна сітка розмірність M на N, що складається з осередків, званих пікселями, кожен піксель містить значення кольору. Вони характеризуються лише двома параметрами: кількістю пікселів та змістом інформації (глибиною кольори) на піксель. Є інші атрибути, які застосовуються до растрових зображень, але вони є похідними від цих двох фундаментальних параметрів.

Нижче на малюнку 2.3.3.1 представлена схема матриці пікселів у растровом зображення.

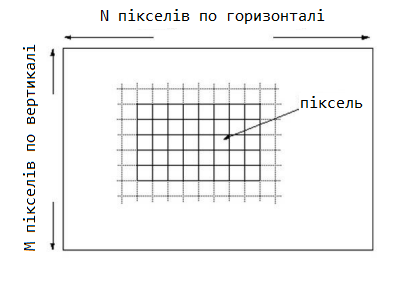


Рисунок 2.3.3.1 – Матриця пікселів растрового зображення

При колірній палітрі RGBA колір – це набір компонентів:

R – червоний колір;

G – зелений колір;

B – синій колір;

A – альфа-канал, що використовується для кодування рівня прозорості. При 32 бітній глибині з чотирма каналами кольору кожен канал кодується 8 бітами. Модель представлення пікселя у двійковому вигляді зображено малюнку 2.3.3.2

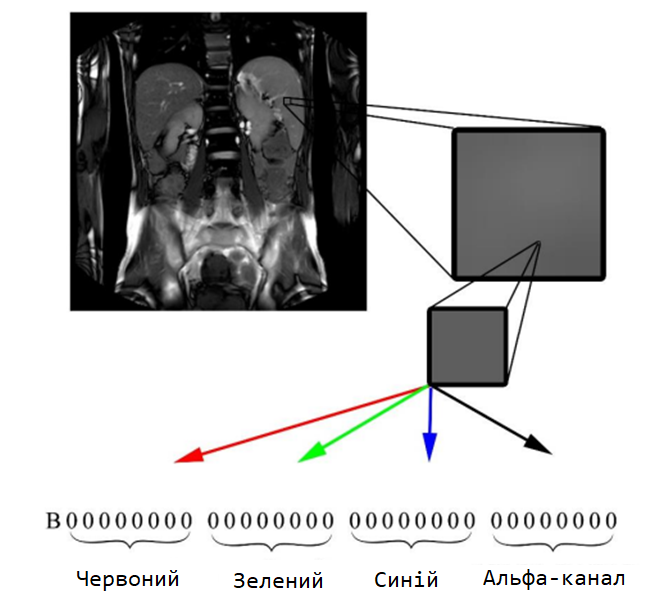


Рисунок 2.3.3.2 – Модель пікселя

Таким чином, один колір може бути закодований діапазоном значень від 0 до 255 (при 8-бітному кодуванні). Формат DICOM часто використовує кодування у 10, 12 або 16-бітному форматі. При кодуванні з 16 бітною глибиною кольору максимальна кількість значень дорівнюватиме 2^16 = 65536.

### 2.3.4 Особливості виконання схем алгоритмів

**Алгоритм процесу знаходження рішення задачі**

Diagram, text

Description automatically generated

Рисунок 2.3.4.1 Алгоритм функції HasReadTags

Diagram

Description automatically generated

Рисунок 2.3.4.2 Алгоритм функції ReadTags

Diagram

Description automatically generated

Рисунок 2.3.4.3 Алгоритм функції ReadToBitmap (частина перша)

Diagram

Description automatically generated

Рисунок 2.3.4.4 Алгоритм функції ReadToBitmap (частина друга)

Diagram

Description automatically generated

Рисунок 2.3.4.5. Алгоритм функції Read

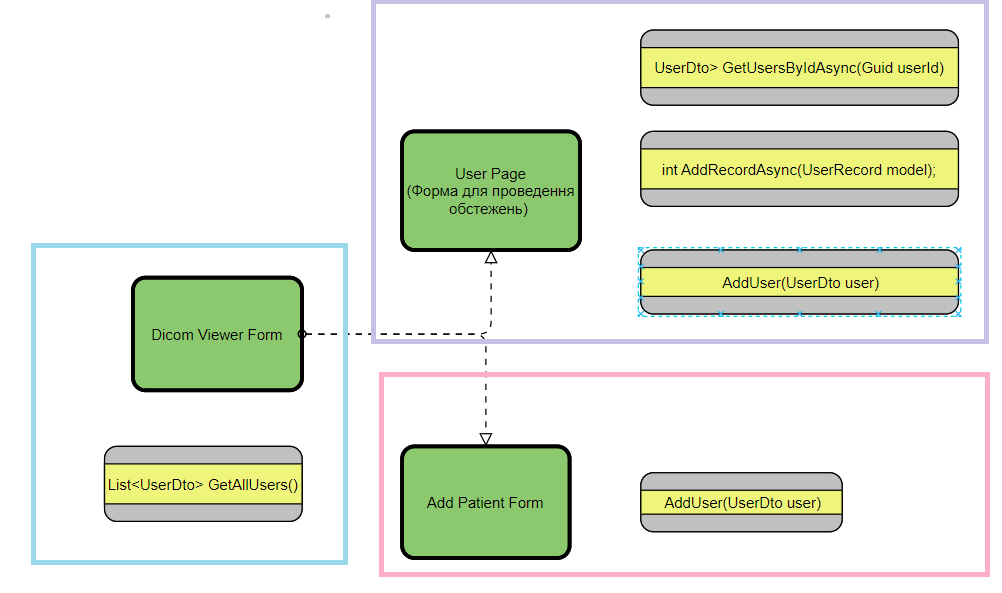


Рисунок 2.3.4.6. Схема ієрархія компонентів програми

### Розробка програмного продукту

Інструменти розробки програмного продукту:

|  |  |
| --- | --- |
| Мова програмування | С# версія 9.0 |
| Framework розробки | net5.0-windows |
| База даних | Реляційна, SQL Server |
| Тип програмного продукту | Дэсктопний, Windows Forms |
| Пакет для логування подій | Serilog, версія 20.11.0 |

Таблиця 2.3.5.1 Структура таблиці User

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | nvarchar(50) | Індетифікатор пацієнта |
| Name | nvarchar(50) | Ім’я пацієнта |
| DOB | date | Дата народження пацієнта |
| Sex | bit | Стать пацієнта |
| Complaints | nvarchar(MAX) | Нотатки про скарги пацієнта на момент реєстрації |

Таблиця 2.3.5.1 Структура таблиці Record

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | nvarchar(50) | Індетифікатор обстеження |
| Date | date | Дата обстеження |
| BodyPart | nvarchar(MAX) | Частина тіла, що обстежується |
| UserId | nvarchar(50) | Ідентифікатор пацієнта |
| Note | nvarchar(MAX) | Нотатки до обстеження |
| FileName | nvarchar(MAX) | Ім’я DICOM файла обстеження |

### 2.3.6 Текст програми

Текст классу **RepositoryBase.cs**, що є базовим для всіх репозіторїв і містить їх базову функціональність.

using System;

using System.Data.SqlClient;

using System.Threading.Tasks;

namespace DAL.Models

{

public abstract class RepositoryBase

{

protected abstract string TableName {get; }

protected abstract string CreateTableQuery {get; }

private readonly IDbRequestExecutor \_dbRequestExecutor;

private readonly IDbInfrastructurer \_dbInfrastructurer;

protected RepositoryBase(

IDbRequestExecutor dbRequestExecutor,

IDbInfrastructurer dbInfrastructurer)

{

\_dbRequestExecutor = dbRequestExecutor;

\_dbInfrastructurer = dbInfrastructurer;

}

protected async Task EnsureTableExists()

{

if (!DatabaseExists())

{

\_dbInfrastructurer.CreateDatabase();

}

if (await TableExistsAsync())

{

return;

}

var dbConnection = DbManager.GetConnectionWithDb();

SqlCommand cmd = new SqlCommand(CreateTableQuery, dbConnection);

cmd.Parameters.Add(new SqlParameter("@tableName", TableName));

try

{

dbConnection.Open();

cmd.ExecuteNonQuery();

Console.WriteLine("Table Created Successfully");

}

catch(SqlException e)

{

Console.WriteLine("Error Generated. Details: " + e.ToString());

}

finally

{

dbConnection.Close();

}

}

private async Task<bool> TableExistsAsync()

{

var dbConnection = DbManager.GetConnectionWithDb();

var isTableExists = "SELECT \* FROM DicomViewerDb.sys.tables WHERE name=@tableName";

var isDbExistsCommand = new SqlCommand(isTableExists, dbConnection);

isDbExistsCommand.Parameters.Add(new SqlParameter("@tableName", TableName));

return (await \_dbRequestExecutor.ExecuteScalarAsync(isDbExistsCommand, dbConnection)) != null;

}

private bool DatabaseExists()

{

string sqlCreateDBQuery;

bool result = false;

try

{

var tmpConn = DbManager.GetConnection();

sqlCreateDBQuery = string.Format("SELECT database\_id FROM sys.databases WHERE Name = '{0}'", DbManager.GetDbName());

using (tmpConn)

{

using (SqlCommand sqlCmd = new SqlCommand(sqlCreateDBQuery, tmpConn))

{

tmpConn.Open();

object resultObj = sqlCmd.ExecuteScalar();

int databaseID = 0;

if (resultObj != null)

{

int.TryParse(resultObj.ToString(), out databaseID);

}

tmpConn.Close();

result = (databaseID > 0);

}

}

}

catch (Exception ex)

{

result = false;

}

return result;

}

}

}

Текст классу **RecirdRepository.cs**, що забезпечує взаємодію програми із таблицею “Record”

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Threading.Tasks;

namespace DAL.Models

{

public interface IRecordRepository {

Task<UserRecord> GetByIdAsync(Guid userId);

Task<List<UserRecord>> GetAllByUserAsync(Guid userId);

Task<int> AddRecordAsync(UserRecord model);

}

public class RecordRepository : RepositoryBase, IRecordRepository

{

private readonly IDbRequestExecutor \_dbRequestExecutor;

public RecordRepository(

IDbRequestExecutor dbRequestExecutor,

IDbInfrastructurer dbInfrastructurer)

: base(dbRequestExecutor, dbInfrastructurer)

{

\_dbRequestExecutor = dbRequestExecutor;

}

public async Task<int> AddRecordAsync(UserRecord model)

{

await EnsureTableExists();

var dbConnection = DbManager.GetConnectionWithDb();

string addUserExpression = $"INSERT INTO [{TableName}] (Id,Date,BodyPart,Note,UserId,FileName)"

+ "VALUES (@Id,@Date,@BodyPart,@Note,@UserId,@FileName)";

var addUserCommand = new SqlCommand(addUserExpression, dbConnection);

addUserCommand.Parameters.Add(new SqlParameter("@Id", model.Id));

addUserCommand.Parameters.Add(new SqlParameter("@Date", model.RecordDate));

addUserCommand.Parameters.Add(new SqlParameter("@BodyPart", model.BodyPart));

addUserCommand.Parameters.Add(new SqlParameter("@Note", model.Note));

addUserCommand.Parameters.Add(new SqlParameter("@FileName", model.FileName));

addUserCommand.Parameters.Add(new SqlParameter("@UserId", model.UserId));

return await \_dbRequestExecutor.ExecuteNonQueryAsync(addUserCommand, dbConnection);

}

public async Task<List<UserRecord>> GetAllByUserAsync(Guid userId)

{

await EnsureTableExists();

var resultList = new List<UserRecord>();

var sqlExpression = $"SELECT Id, BodyPart, Date, FileName, Note from [{TableName}] where UserId = @id";

await using var connection = DbManager.GetConnectionWithDb();

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@id", userId));

var reader = await command.ExecuteReaderAsync();

if (reader.HasRows)

{

while (reader.Read())

{

resultList.Add(new UserRecord

{

Id = Guid.Parse(reader.GetValue(0).ToString() ?? string.Empty),

BodyPart = reader.GetValue(1).ToString(),

RecordDate= DateTime.Parse(reader.GetValue(2).ToString() ?? string.Empty),

FileName = reader.GetValue(3).ToString(),

Note = reader.GetValue(4).ToString()

});

}

}

await reader.CloseAsync();

return resultList;

}

public async Task<UserRecord> GetByIdAsync(Guid userId)

{

await EnsureTableExists();

UserRecord model = null;

var sqlExpression = $"Select Id, BodyPart, Date, FileName, Note from[{TableName}] where Id = @id";

await using var connection = DbManager.GetConnectionWithDb();

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@id", userId));

var reader = await command.ExecuteReaderAsync();

if (reader.HasRows)

{

while (reader.Read())

{

model = new UserRecord

{

Id = Guid.Parse(reader.GetValue(0).ToString() ?? string.Empty),

BodyPart = reader.GetValue(1).ToString(),

RecordDate = DateTime.Parse(reader.GetValue(2).ToString() ?? string.Empty),

FileName = reader.GetValue(3).ToString(),

Note = reader.GetValue(4).ToString()

};

}

}

await reader.CloseAsync();

return model;

}

protected override string TableName => "Record";

protected override string CreateTableQuery =>

"CREATE TABLE [dbo].[Record](" +

"[Id][nvarchar](50) NOT NULL PRIMARY KEY CLUSTERED ," +

"[Date] [date] NOT NULL," +

"[BodyPart] [nvarchar] (max) NULL," +

"[Note] [nvarchar] (max) NULL," +

"[UserId] [nvarchar] (50) NOT NULL," +

" [FileName] [nvarchar] (max) NOT NULL," +

"CONSTRAINT fk\_user\_record FOREIGN KEY(UserId) REFERENCES[dbo].[Users] (Id))";

}

}

Текст класу Program.cs, що є точкою входу в програму

using System;

using System.Windows.Forms;

using BLL;

using DAL.Models;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Serilog;

namespace DicomViewerProj

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

\

var host = new HostBuilder()

.ConfigureServices((hostContext, services) =>

{

services.AddSingleton<DicomViewer>();

services.AddLogging(configure => configure.AddConsole());

services.AddScoped<IDbInfrastructurer, DbInfrastructurer>();

services.AddScoped<IUserRepository, UserRepository>();

services.AddScoped<IUserService, UserService>();

services.AddScoped<IDbRequestExecutor, DbRequestExecutor>();

services.AddScoped<IRecordRepository, RecordRepository>();

}).Build();

ConfigureSerilog();

using var serviceScope = host.Services.CreateScope();

{

var services = serviceScope.ServiceProvider;

try

{

var form1 = services.GetRequiredService<DicomViewer>();

Application.Run(form1);

Console.WriteLine("Run Form1 is successful");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

private static void ConfigureSerilog()

{

Log.Logger = new LoggerConfiguration()

.MinimumLevel.Debug()

.WriteTo

.MSSqlServer(DbManager.GetConnectionWithDb().ConnectionString, "Logs")

.CreateLogger();

}

}

}

### 2.3.7. Програма і методика випробувань

### 2.3.8 Висновки

### 2.3.9 Інструкція користувачу