De-identifying Medical Images – Made Easy

Project Report

Student Details

Name: Likhith Ravula

NUID: 002293027

Course Details

Course ID: IE 7945

Faculty: Prof Kirankumar Trivedi

Submission Details

Date: June 30th, 2024

By: Likhith Ravula

Contribution: 100%

Contents:

Project repo: GitHub

1. Introduction

To protect people's privacy, data must be anonymized by taking out or changing any details that could be used to identify them. This makes it safe to share the data with others for various purposes across the industry/ organization.

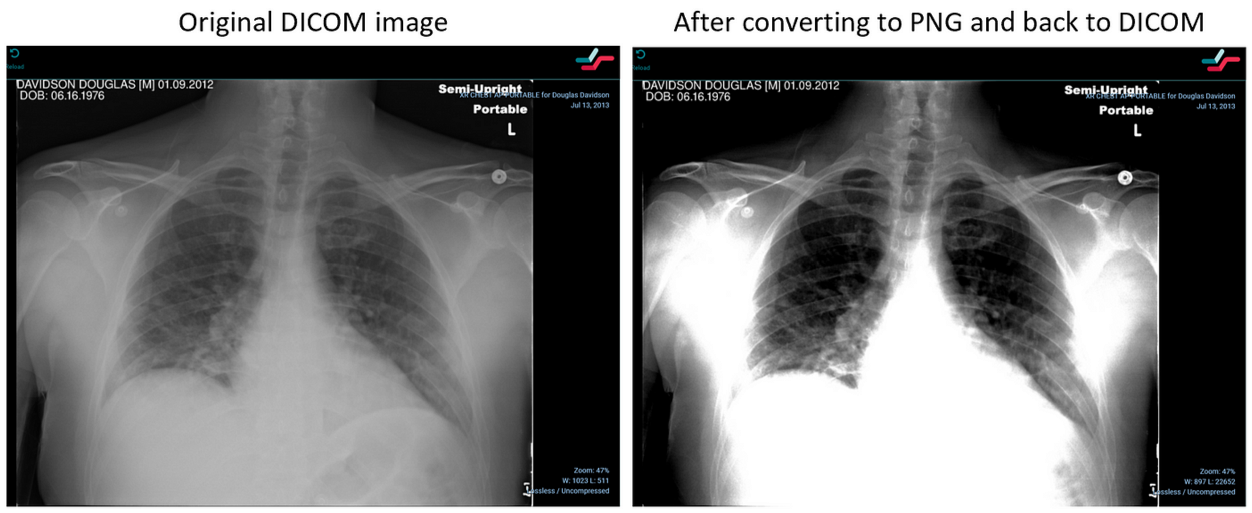
Regulations like HIPAA exist to anonymize data, especially in healthcare. This anonymization process ensures information can't be linked back to specific individuals. For instance, the human subject research data needs to be analyzed but privacy for the participants must be a top priority. De-identification helps achieve this balance.

Direct identifiers and demographics, also known as Protected Health Information (PHI), includes a patient's name, address, gender, etc., and convey a patient's physical or mental health condition, or diagnosis related to that individual, as well as financial data related to healthcare like, medical records, bills, and lab results. These must be de-identified before the data is stored/ shared across serves.

2. Existing System

The existing methods to de-identify images have a lot of limitations including file formats, etc. The major limitation is that the existing methods are built for traditional file formats such as PNG, JPG, etc. Medical images (MR, CT, and others) are transmitted and stored and processed in a file type called DICOM (Digital Imaging and Communication in Medicine), which is a standard built to adhere to the HIPAA norms.

Converting medical images (DICOM) to common formats (e.g., JPEG) works for feeding data to ML models (Computer Vision & NLP) that remove sensitive text. However, this conversion can degrade image quality. Medical images use specific pixel data formats (photometric interpretations) that differ from common image formats. While using the converted images in ML models might be possible, converting the anonymized results back to DICOM format becomes difficult as the conversion process compresses the image and loses crucial data embedded in the original DICOM file metadata.


Original DICOM image
After converting to PNG and back to DICOM
DAVIDSON DOUGLAS [M] 01.09.2012
DOB: 06.16.1976
Semi-Upright
Portable
L
DAVIDSON DOUGLAS [M] 01.09.2012
DOB: 06.16.1976
Semi-Upright
Portable
L

source

3. Architecture

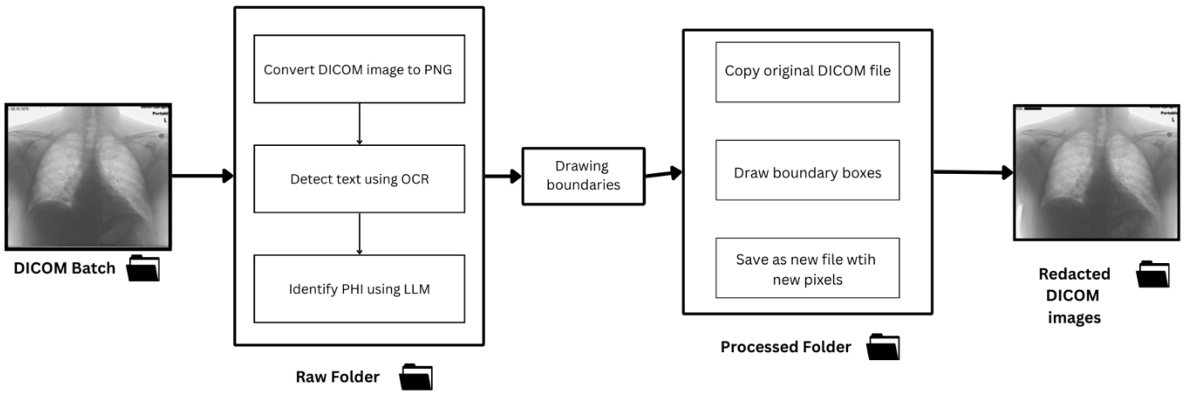
3.1. Core Architecture

Using open-source python modules and services, this architecture avoids the problem of losing the image quality. The architecture is as follows:

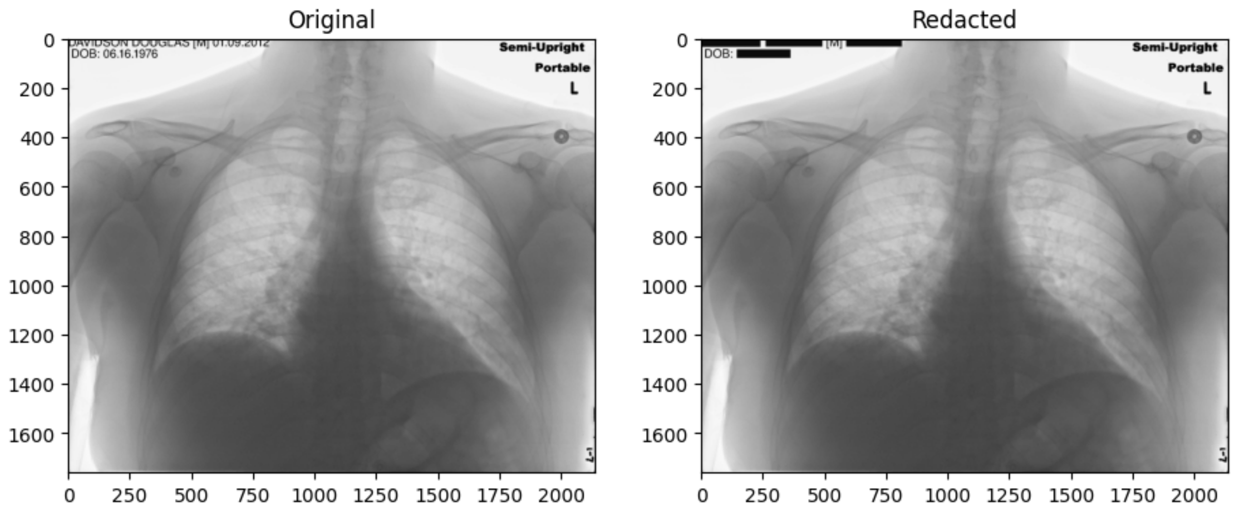
- Given images are converted into PNG for the image and text identification.
- OCR (Optical Character Recognition) using Tesseract detects all text in the image.
- A fine-tuned LLM (Large Language Model), llama2 then pinpoints only the sensitive text containing personal health information (PHI).
- Boundary boxes are then drawn around the pixels containing the PHI.

- The same pixels' coordinates are then saved.
- The original DICOM image is then copied and referencing the saved pixels' coordinates, the PHI pixels in the original DICOM file are redacted.

Only the pixels corresponding to this identified PHI in the original DICOM file are then modified so that the de-identification is performed without any compression or loss.



Below is an example of an original DICOM image and a redacted DICOM image.



source

3.2. Extended Architecture

A front-end application, developed using Flutter, is supported by a MySQL backend database. Flask APIs developed in python establishes the communication between the frontend and backend.

The architecture can be described as comprising the following components:

1. Front-end App:

   - Function: User interface where users interact with the application.

   - Interaction: Sends user input and requests to the Flask App Service, for login, triggering the services.

2. Flask App Service:

   - Function: Acts as the backend server, processing requests from the front-end app.

   - Interactions:

     - Front-end App: Receives requests from and sends responses back to the front-end app

     - MySQL Database: Manages sign-in credentials and logs the pipeline's performance

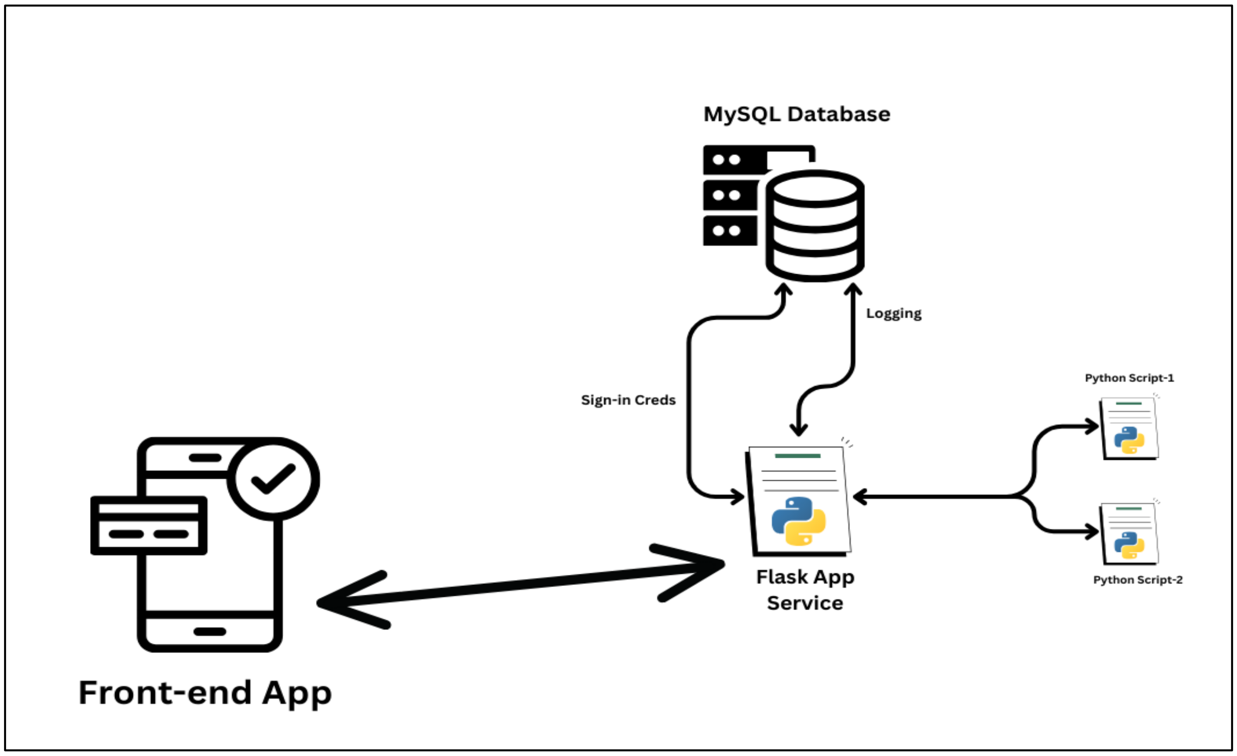
3. Python Scripts:

   - Function: Performs tasks from reading the images, converting them, identifying the PHI, and redacting the DICOM images.

4. MySQL Database:

   - Function: Supports the entire application by storing the credentials and storing the logs and metadata of the images in the pipeline.

These components together create a cohesive system where the front-end app handles user interactions, the Flask App Service processes these interactions, executes necessary scripts, and manages data through the MySQL database.



4. Data Source

The DICOM images are extracted from The Cancer Imaging Archive (TCIA) for the development and evaluation of medical image de-identification. Data citation: Rutherford, M., Mun, S.K., Levine, B., Bennett, W.C., Smith, K., Farmer, P., Jarosz, J., Wagner, U., Farahani, K., Prior, F. (2021). A DICOM dataset for evaluation of medical image de-identification (Pseudo-PHI-DICOM-Data) [Data set]. The Cancer Imaging Archive. DOI: https://doi.org/10.7937/s17z-r072

5. Technology Stack

5.1. Converting DICOM to PNG

DICOM image is converted to a PNG image when the image is loaded into the pipeline. The conversion is performed to extract the text on the image. Tesseract model is employed to identify the text, and as the model cannot take a '.dcm' file, I am are converting and ingesting a '.png' file. Python package "dicom2png" is used to convert the DICOM images to PNG imges.

5.2. OCR using Tesseract

Tesseract is a widely used open-source optical character recognition (OCR) engine that converts printed or handwritten text within images into machine-readable text. Developed by HP and maintained by Google, Tesseract supports a wide range of languages and is highly adaptable for various OCR tasks. Its robust accuracy and ease of integration make it a popular choice for applications requiring text extraction from scanned documents, photographs, and other image sources. Tesseract operates by analyzing pixel patterns and translating them into text characters, making it a powerful tool for digitizing physical documents, enabling search and edit functionalities, and automating data entry processes.

5.3. Identifying PHI using LLM

Large Language Models (LLMs) are advanced artificial intelligence systems designed to understand and generate human-like text based on vast amounts of training data. These models utilize deep learning techniques to process natural language, making them capable of tasks such as translation, summarization, question answering, and text generation. By leveraging extensive datasets, LLMs can predict and generate coherent text, mimicking human language patterns and contextual understanding.

LLaMA 2 is a specific instance of a large language model developed by Meta (formerly Facebook). Building on the foundation of its predecessor, LLaMA, LLaMA 2 offers improved performance and enhanced capabilities in natural language understanding and generation. It is designed to be more efficient, requiring less computational power while maintaining high accuracy and fluency in text generation.

Technicalities of LLaMA 2

LLaMA 2 employs a transformer architecture, which is the backbone of many modern LLMs. It utilizes a multi-head attention mechanism to process input text, allowing the model to weigh the importance of different words and phrases in context. This model has been trained on diverse and extensive datasets, enabling it to capture nuanced language patterns and produce coherent, contextually appropriate responses. LLaMA 2 also incorporates advanced training techniques such as mixed precision training and gradient checkpointing, which enhance computational efficiency and reduce memory usage, making it accessible for a broader range of applications and environments.

5.4. FLASK API

Flask is a lightweight web framework written in Python, designed to make it easy to build web applications and APIs. It follows a minimalist approach, providing only the essential components to get a web application up and running, which makes it highly flexible and scalable. A Flask API is an application programming interface built using the Flask framework. It allows developers to create RESTful web services that can handle HTTP requests and responses, enabling communication between different software systems.

HTTP methods used: GET, POST

Endpoints used:

1. '/login'
2. '/stats'
3. '/convert'
4. '/mask_image'

6. Mobile App

Flutter is an open-source UI software development toolkit created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Dart is the programming language used to write Flutter applications, known for its performance and ease of use. It enables developers to create visually appealing and high-performance apps with a single codebase that works across multiple platforms, including iOS, Android, web, and desktop. It uses a reactive framework, which allows for efficient and dynamic UI updates.

6.1. Technical aspects of Flutter

Flutter employs a unique architecture based on widgets. Everything in Flutter is a widget, from structural elements like buttons and text to stylistic elements like padding and color schemes. This widget-based approach allows for highly customizable and flexible UI designs. The framework includes a rich set of pre-designed widgets that adhere to both Material Design (for Android) and Cupertino (for iOS) guidelines, ensuring a native look and feel across platforms.

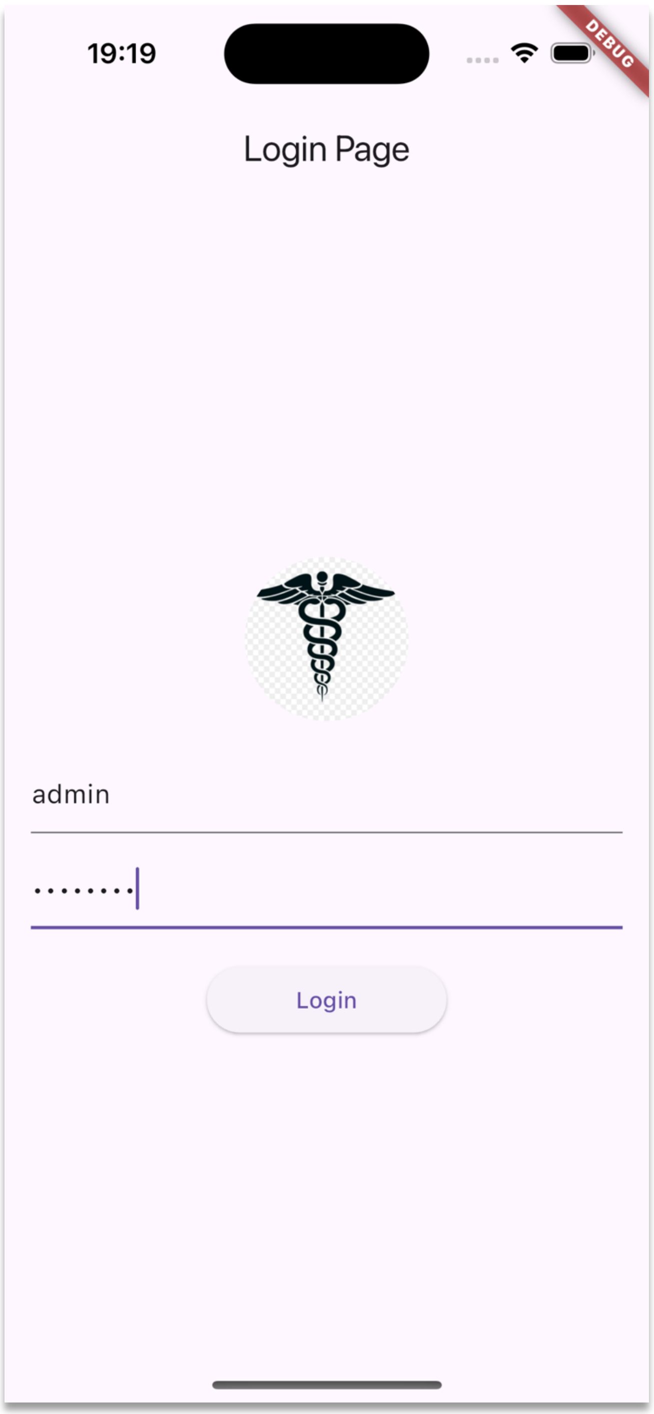
Dart is the programming language used by Flutter. It is an object-oriented, class-based language with a syntax similar to JavaScript, making it easy to learn for developers familiar with JavaScript or Java.

Flutter uses the Skia graphics engine, which provides fast and smooth rendering of UIs. It's hot-reload feature significantly speeds up the development process by allowing developers to see changes in real-time without restarting the app.

6.2. Screens

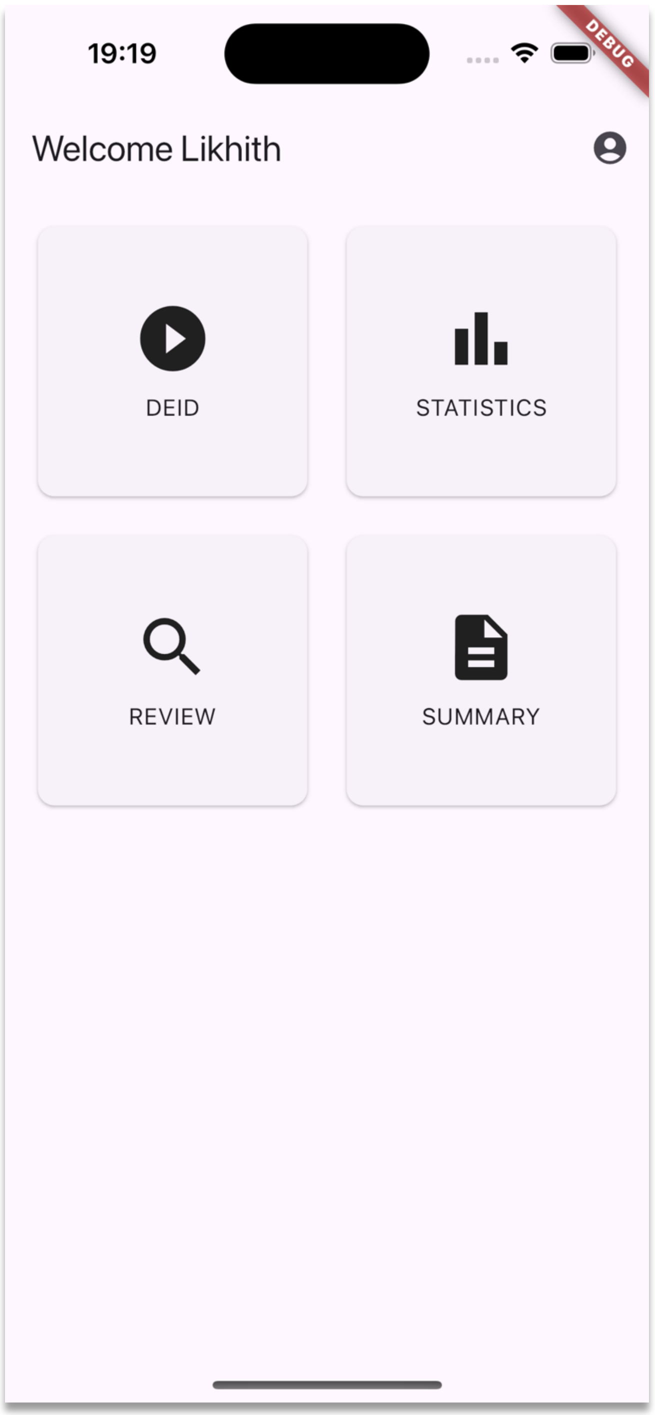
1. Sign-in Page

User logs in, into this page using their credentials. The credentials are fetched and validated from the back-end MySQL database.



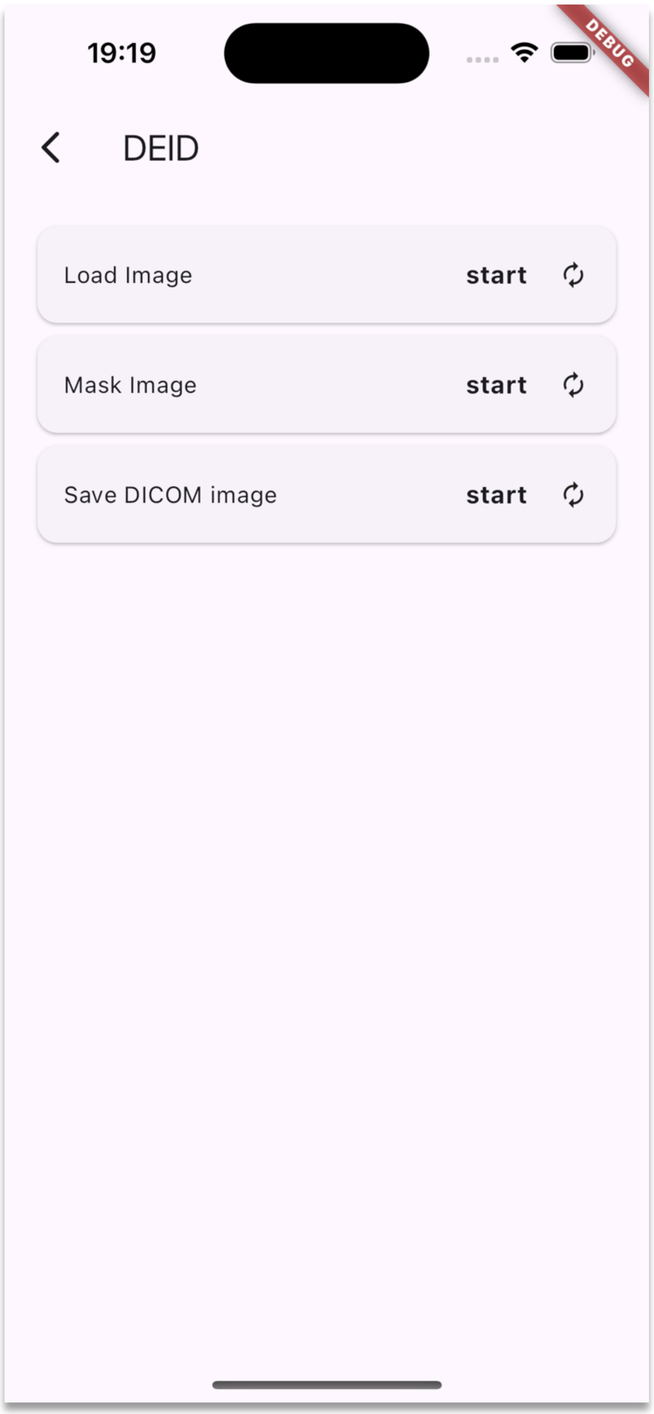
2. Dashboard Page

This page is the landing page after signing-in into the app. User's name is dynamically fetched from the MySQL database. Users can navigate to pages like 'Account Settings Page', 'DEID Pipeline Page', 'Statistics Page', etc.



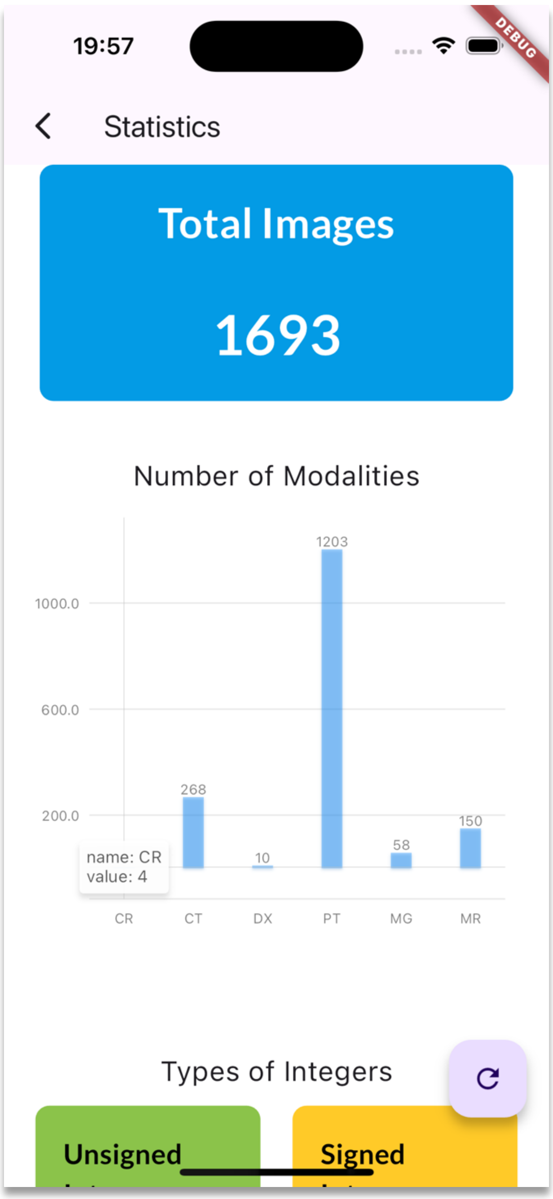
3. DEID Pipeline Page

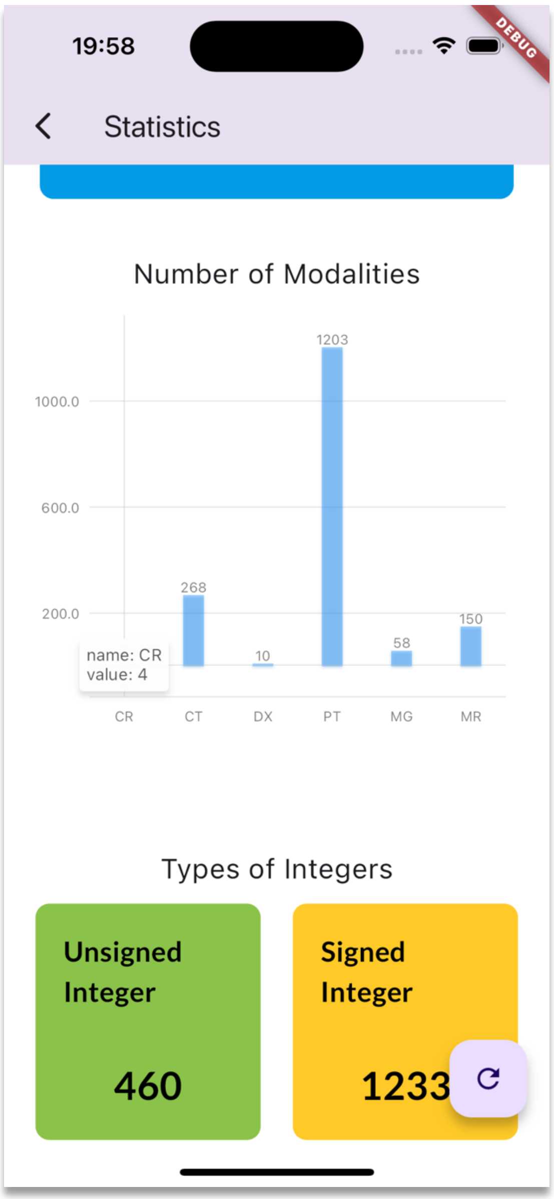
Users can trigger a batch/ image for de-identification. At the moment, the pipeline is broken down into steps and users have the option to trigger each step separately.



4. Statistics Page

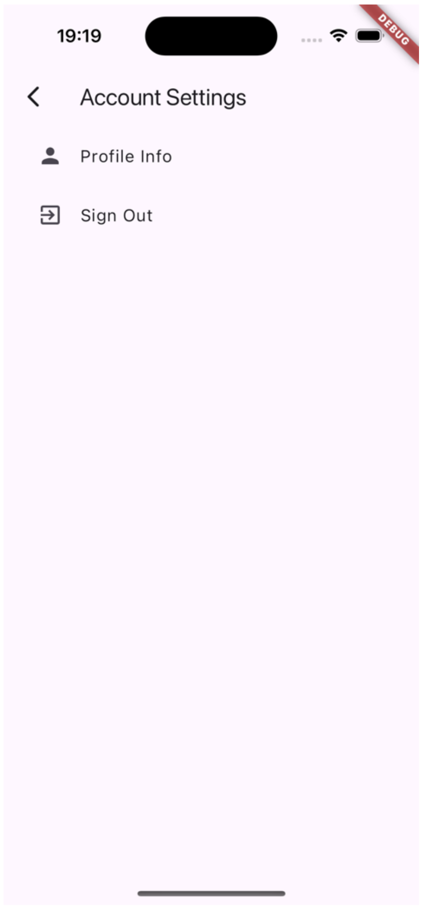
Users can see few quick statistics in this page. It shows the overall images in the given batch and shows counts like total images, number of images with singed and unsigned pixels.





5. Account Settings Page

Users can see their profile info and sign-out of their working instance. The profile information will be dynamically retrieved from the database.



7. Future Scope

7.1. An end-to-end MVP

An entire end-to-end product can be built by leveraging the technologies used in this project. From designing, developing, and implementing a full-scale sophisticated back-end database to improving the animations and introducing new graphs, from fine-tuning and refining the model to implementing an automated end-to-end pipeline.

7.2. Further usage of LLaMA2 model:

LLaMA 2, as a sophisticated large language model, offers numerous applications in the field of medical reports. Its advanced natural language processing capabilities can significantly enhance various aspects of medical documentation and analysis. DICOM images are often associated with a medical report or doctor written description. The current project aims to redact the sensitive text, but the future scope of the project includes using this LLM to:

1. Improved Accuracy and Consistency

   LLaMA 2 can assist in generating medical reports with improved accuracy and consistency. By analyzing patient data and medical records, it ensures that the terminology used is precise and standardized, reducing the risk of errors and discrepancies.

2. Automated Report Generation

   LLaMA 2 can automate the generation of medical reports by extracting relevant information from patient records, lab results, and diagnostic images. This automation saves time for healthcare professionals, allowing them to focus more on patient care rather than administrative tasks.

3. Enhanced Data Analysis

   LLaMA 2 can analyze large volumes of medical data to identify patterns and correlations that might be missed by human analysts. This capability is particularly useful in research, where insights from data can lead to new medical discoveries and improved patient outcomes.

4. Multilingual Support

   With its ability to understand and generate text in multiple languages, LLaMA 2 can assist in creating medical reports for diverse patient populations, ensuring that language barriers do not impede the quality of care and documentation.

Among many other uses and scope of using an open-source LLM such as LLaMA2, the above are the best use cases that I would like to target.

8. Literature and citations

[1] Macdonald JA, Morgan KR, Konkel B, Abdullah K, Martin M, Ennis C, Lo JY, Stroo M, Snyder DC, Bashir MR. A Method for Efficient De-identification of DICOM Metadata and

Burned-in Pixel Text. J Imaging Inform Med. 2024 Apr 8. doi: 10.1007/s10278-024-01098-7. Epub ahead of print. PMID: 38587767.

[2] Vcelak P, Kryl M, Kratochvil M, Kleckova J. Identification and classification of DICOM files with burned-in text content. Int J Med Inform. 2019 Jun;126:128-137. doi: 10.1016/j.ijmedinf.2019.02.011. Epub 2019 Mar 1. PMID: 31029254.

[3] Monteiro E, Costa C, Oliveira JL. A De-Identification Pipeline for Ultrasound Medical Images in DICOM Format. J Med Syst. 2017 May;41(5):89. doi: 10.1007/s10916-017-0736-1. Epub 2017 Apr 13. PMID: 28405948.

[4] Presidio Image Redactor, Documentation

[5] Presidio Redactor sample python implementation, Notebook

[6] Redacting sensitive text from DICOM medical images in Python, Article

[7] De-identification in Medical Imaging, Article