Frictionless Medical Image Pipeline for AI purposes Design Document

Definitions:

DICOM: A medical image standard which specifies the sending, storing, and format of medical images. Dicom images contain two main parts, the Patient Health Information (PHI), and the image data.
PACS: Picture Archiving and Communication System. A standard used to store and transmit DICOM images.
Orthanc: A PACS server.

High-level Description:

The project aims at creating an automated x-ray image pipeline, to provide mammography scans to experts to analyze using AI for cancer prediction of the breast tissue.

Current process and problem:

Currently, a female patient attends a clinic to receive a mammogram. After an x-ray scan is complete an image file is created called a DICOM file. DICOM files contain the actual image as well as patient metadata. Patient metadata includes identifiable patient information such as, name, sex, birth date, patient id, clinic id, etc, and contains non-identifiable information such as time of the scan, type of machine for scan, image view, etc.

The image is sent to a medical server via a secure medical image protocol called DIMSE. Those images are then stuck on that physical server in the clinic. Dr. Rajapakshe then has to physically drive to the clinic (approximately every 6 months), remove the hard drive, download the data onto a research server, delete the old DICOM files from the hard drive, and finally, drive the empty hard drive back to the clinic.

Once the files are on the research server, they are anonymized. The purpose is to open these images to scientists to analyze, but these DICOM files must not contain any Patient Health Information (PHI). There are then two servers. One contains the anonymized DICOM images and the other contains only patient metadata, which is kept confidential. The anonymized DICOM images are then released for research.

Solution:

The proposed process will set up two new servers using open-source software called Orthanc. Once the mammography scan is complete the DICOM file will be sent to the first Orthanc server

(via DIMSE protocol) which will store the confidential metadata, anonymize the DICOM file, and create a random new ID to connect the patient health information to the anonymized DICOM file. The anonymized DICOM file is then sent to another Orthanc server. This server has a GUI (Graphical User Interface) which will be used by researchers to download the x-ray scans and perform analysis.

This project will save Dr. Rajapakshe time, and allow for a faster release of new anonymized DICOM files. Instead of having to wait 6 months for new scans, the scans will be available to research almost as soon as the scan itself is complete.

System Architecture Overview:

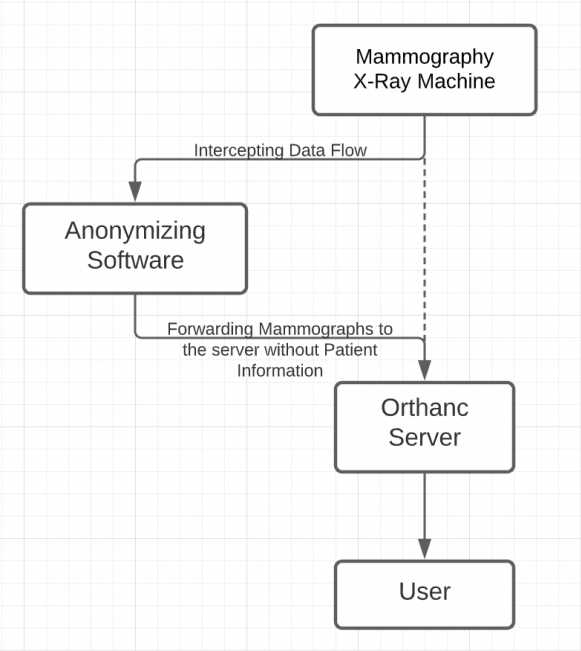
The Orthanc server is software running on top of storage hardware, in our case, it will be a NAS (Network Attached Storage) array. A NAS array is used as that is what our client will be using.

Orthanc is a PACS (picture archiving and communication system) server, which means it uses and implements secure storage and transmission of medical images. PACS is a medical image standard and must be used in this project. Orthanc PACS server is selected per client request.

The protocol used to communicate between Imaging Modalities (in our case x-ray machine) is the DIMSE protocol. This is an industry-standard among Imaging Modalities and PACS servers.

We are using Orthanc because it is an up and running PACS server and that is open source. Creating our own PACS server would be unnecessary as there are already great out-of-the-box solutions. Our client also recommends we use Orthanc.

Orthanc can be easily connected to any imaging modality and configured to send DICOM images as they are created.



The high-level flow is as follows:

- An x-ray machine performs a scan and generates a DICOM file.
- The DICOM file is sent to the Orthanc server (1).
    - Orthanc server (1) has a database.
- Orthanc server (1) stores the PHI (Patient Health Information) in a database.
- Orthanc server (1) anonymizes the DICOM file.
- Orthanc server (1) sends the anonymized DICOM file to Orthanc server (2).
- PHI data in Orthanc server (1) is connected to their respective anonymized DICOM file with a unique GUID.
    - Orthanc server (2) is the end destination of the anonymized DICOM file.
    - Orthanc server (2) will have a database to store the anonymized DICOM files.
- The end customer will communicate directly with this Orthanc server via a built-in GUI provided by Orthanc.
- The researchers download anonymized DICOM files for AI training.

Problems and Edge Cases:

There are some important problems and edge cases to understand that we will be solving in order to understand why we choose certain designs.

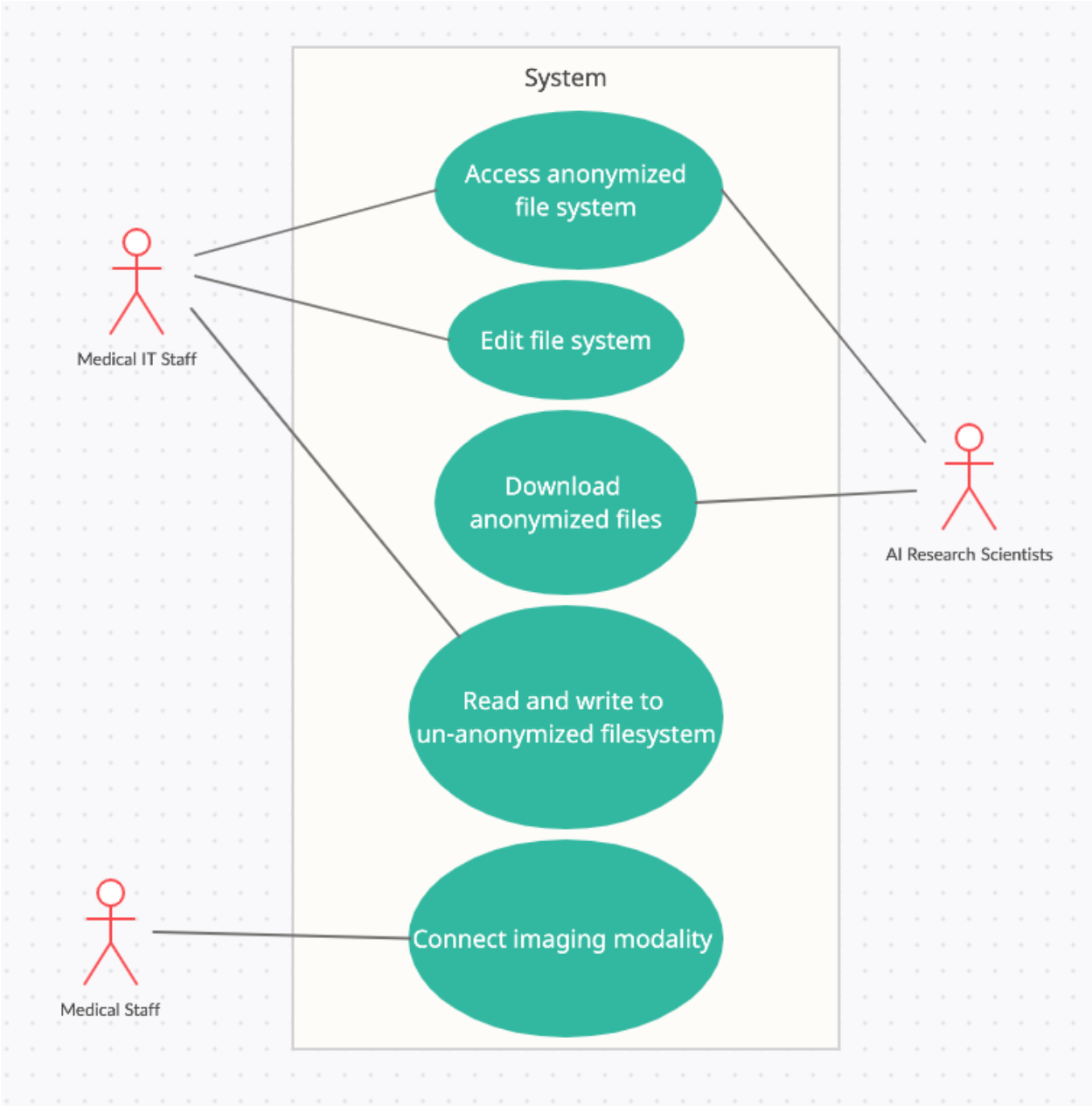
1. It is common for different patients to have the same patient ID. Therefore we cannot use the patient id's as a primary key in our database. This is possible because different clinics can assign the same patient ID to completely different patients.
2. It is possible for the same patient to have different patient IDs. This occurs when the same patient goes to different clinics and gets assigned a different ID at each clinic.

3. Missing information is very common. Patients with missing PHI is common.
4. Patients' images do not come in from the imaging modality back to back. They are out of order and can be separated by weeks.
    4.1. For example, if a patient has a mammography study that takes multiple weeks they will be getting x-rays for the next month. It is important for AI training purposes that this patient is classified as images from the same study, so the AI knows it is the same person.

User Groups:

There are three key users:

1. Medical Professionals
    - They will use the system by configuring the x-ray machine to the Orthanc server
2. Medical IT Staff
    - These are the expert users and Admins.
    - They must be able to access the file system and see all information including anonymized data and un-anonymized data.
    - They should be able to manually change Patient Health Information and add notes to each patient.
3. AI Research Scientists
    - This is our end user.
    - They must be able to download anonymized images from the GUI.
    - They should be able to filter by age, ethnicity, patient study, and patient id.

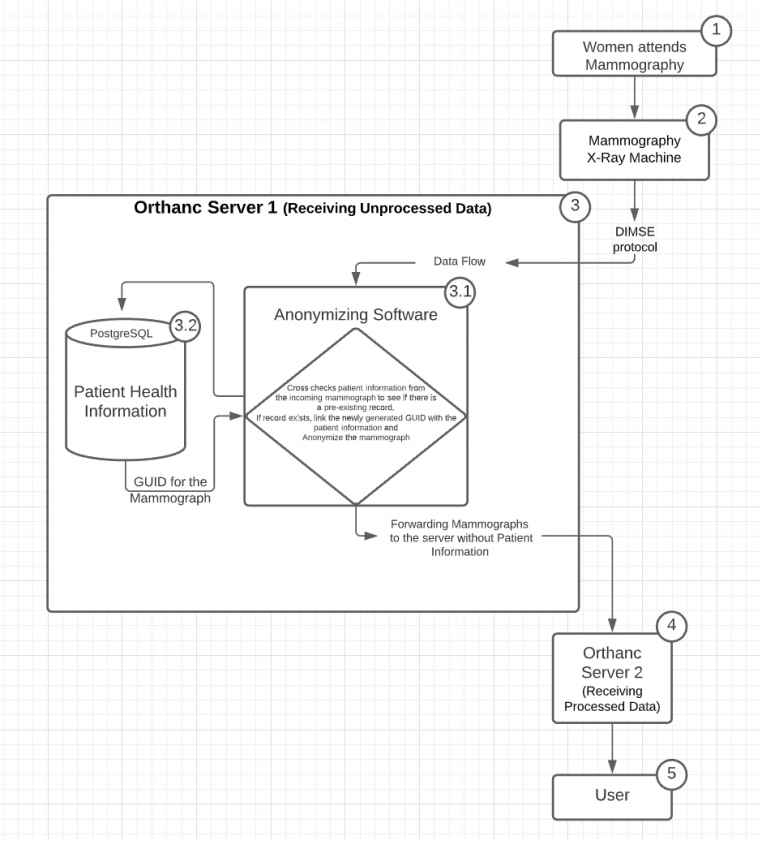

System
Access anonymized file system
Edit file system
Download anonymized files
Read and write to un-anonymized filesystem
Connect imaging modality
Medical IT Staff
AI Research Scientists
Medical Staff

The three main approaches for Databases:

Approach 1 Double Server Setup Using PostgreSQL [RECOMMENDED]:

The first approach that is recommended is using two Orthanc servers with each Orthanc server utilizing a PostgreSQL database.

In this case, PHI is a JSON file and image data is attached to it.



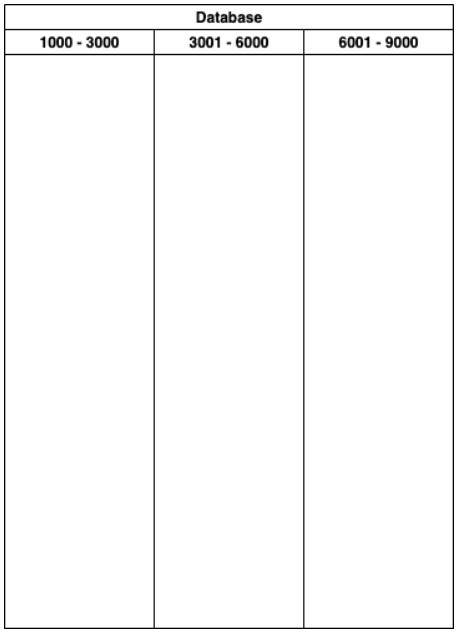
Data flow:

1. An x-ray machine performs a scan and generates a DICOM file.
2. The DICOM file is sent to the Orthanc server (1) using the DIMSE protocol.
    2.1. Orthanc server (1) has a PostgreSQL database.
    2.2. Orthanc server (1) reads the database to check for matches on PHI and study ID.

        2.2.1. If a match is found it gets added as that study.
        2.2.2. If no match is found it becomes a new record.
        2.2.3. If there is only one PHI mismatch it is flagged to be reviewed by an admin.
        2.2.4. If patient ID is different but PHI is the same it gets classified as the same patient.
3. Orthanc server (1) stores the PHI (Patient Health Information) in a PostgreSQL database.
4. Orthanc server (1) anonymizes the DICOM file.
5. Orthanc server (1) sends the anonymized DICOM file to Orthanc server (2) using the DIMSE protocol.
6. PHI data in Orthanc server (1) is connected to their respective anonymized DICOM file with a unique GUID.
    6.1. Orthanc server (2) is the end destination of the anonymized DICOM file.
    6.2. Orthanc server (2) will have a PostgreSQL database to store the anonymized DICOM files.
7. The end customer will communicate directly with this Orthanc server via a built-in GUI provided by Orthanc.
8. The researchers download anonymized DICOM files for AI training. Filtered by study or patient.

Sharding:
Sharding will be used to split the database into sections to optimize read speed as that is important to our use case.

To do this we will split the data using the GUID into sections. It will look like this:



Here the database is split into three sections but we may have more. If the GUID falls in one of those ranges that data will be placed in that range.

This way when retrieving the data we will be able to do so faster as we will have less DICOM images to shuffle through.

Database Organization:

- The database is organized using BIDS (Brain Imaging Data Structure).
- This is a file-type data structure.
- BIDS provides a common way to organize Medical Image data and is an industry-standard.
- Both PostgreSQL databases will be organized using BIDS.

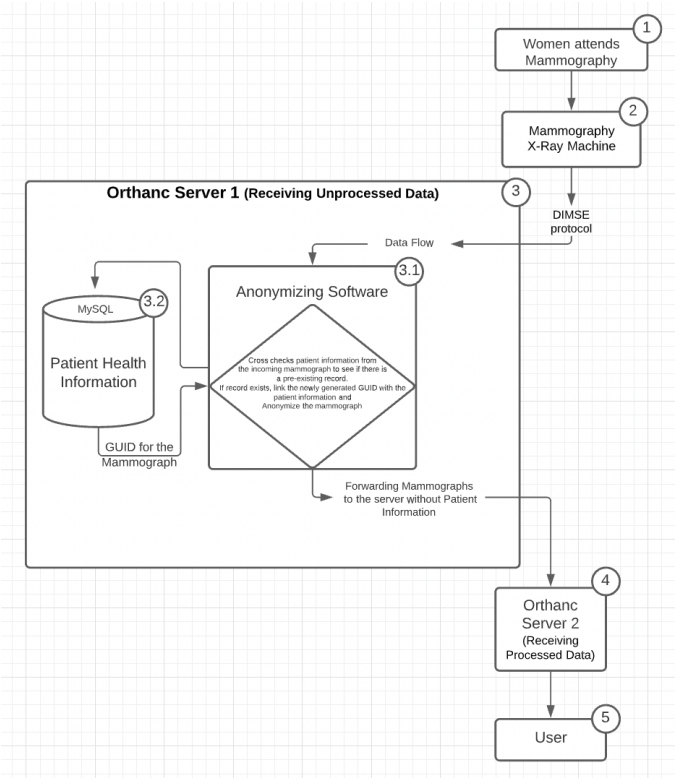
Pros:

- Remended by Orthanc for the best performance.
    - Most commonly used by large hospitals running large medical image datasets.
- Better query support.
    - This is important as we will be running many queries on the database to check for pre-existing.
- More capable of sharding and partitioning.
    - This will help increase our read and write speed.
- More well-known approach for handling large datasets with large files, especially DICOM files.
- Open-source.
- More scalable.
- Least complex to implement.
    - As PostgreSQL is optimized for Orthanc there is a lot of documentation that exists
- With partitioning, PostgreSQL will give the fastest read speed due to having the ability to run parallel queries.

Cons:

- Not traditional.
    - Traditionally relational databases are used for storing medical images.

Approach 2 Double Server Setup Using MySQL:



Data flow:
The data flow will be the same except in this case a MySQL database will be used.

Database Organization:
The database will be organized using the BIDS format.

Sharding:
The database will also utilize sharding to have a faster read speed.

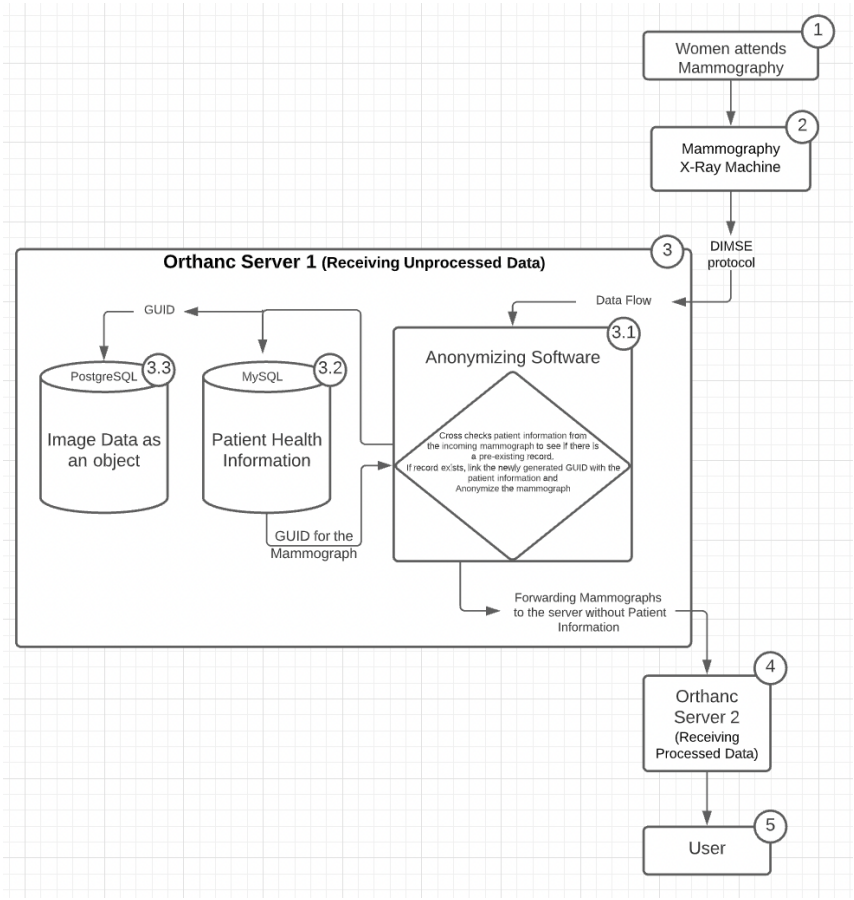
Pros:

- Slightly faster read speed.
    - This is only achievable if we do not do any database partitioning.
- More efficient queries.

Cons:

- Constricting:
    - Less scalable.
- Slower write speed (slower data loading).
    - Longer to write new study and patient records.
- If we use partitioning the read speed will be slower than PostgreSQL since there is no support for parallel queries.

Approach 3 Double Server Hybrid Setup(Using MySQL and PostgreSQL):



Since a DICOM file contains two parts, PHI and image data, it is logical to not store these together but separate them. Instead of having a DICOM file as one data entry, we have two separate databases. A relational database (MySQL) containing PHI, and a non-relational database (PostgreSQL) containing image data. These two entries are then tied together.

Dataflow:

1. A new DICOM image gets sent from an imaging modality to Orthanc Server (1).
2. Orthanc Server (1) splits the data into PHI and image data.
    2.1. The PHI gets placed in a MySQL database.
    2.2. The image data is stored as an object on a PostgreSQL database.
    2.3. A unique GUID is created to match these entries
3. Orthanc Server (1) sends the PostgreSQL image object to Orthanc Server (2).
4. Orthanc Server (1) sends anonymized PHI to Orthanc server (2)
    4.1. Orthanc Server (2) stores the anonymized PHI in a MySQL database.
5. The end customer will communicate directly with this Orthanc server via a built-in GUI provided by Orthanc.

6. The researchers download anonymized DICOM files for AI training. Filtered by study or patient.

Pros:

- Since we do not compare images but PHI with new DICOM entires we can have the fastest read speed of the three approaches.

Cons:

- Hard to do, very new technology.
- May require us to build our own PACS server to support such a setup.
- Minimal documentation is available.

Main approaches for Anonymization:

The main priority of DICOM file anonymization is that it is irreversible.

As both of our approaches utilize a two-server approach with a unique GUID it is irrelevant how we anonymize our PHI data.

Approach 1 Custom Anonymization:

- We build simple anonymization that erases patient data.
- We will loop through PHI and simply delete it.

Pros:

- Simple
- Free

Cons:

- Not as secure
    - What if an error occurs during anonymization displaying PHI to the public.

Approach 2 Anonymization tool [RECOMMENDED]:

- We use an anonymization tool.

Pros:

- Used by many hospitals.
- Very safe, less likely to have a failure in the anonymization process.

Cons:

- Not free.

Functional Requirements:

1. The first Orthanc server should handle requests from the x-ray machine via DIMSE protocol. When a woman gets mammography the x-ray machine will send the DICOM file to the Orthanc server.

2. The Orthanc server will then store the DICOM file PHI (Patient Health Information), and anonymize the DICOM files. The anonymized DICOM file and the PHI will have a random GUID linking them. The Anonymized DICOM file will then be sent to the next Orthanc server.

3. The GUID is created as a crosswalk to be able to link anonymized DICOM files to their respective patient in the PHI database on Orthanc server 1.

4. The Server will hold receive anonymized DICOM files and reveal them to the researchers. The researchers will be able to download anonymized DICOM files for AI training.

5. Orthanc server (1) will cross-reference new inbound DICOM files from imaging modalities with the database to check if the patient is a duplicate in the database. This is done to take account of the following scenarios:
    - **Scenario 1 (The new patient has the same patient ID as a patient in the database but their PHI such as name, date of birth, and gender are different):**
        - In this case, the patient will be added with a new GUID to the database.
        - If it is the same patient ID but only one field is different ex) the wrong name, we will assume this is a typo and flag the patient to be reviewed by an admin, as is industry practice.

    - **Scenario 2 (The new patient has a different patient ID but same PHI):**
        - In this case, we will add the patient with a new GUID as a separate patient as it is possible another clinic has assigned the patient a duplicate ID.

6. Orthanc server (1) should catalog and group incoming DICOM images. When researchers are downloading DICOM files they can sort by study or patient. The AI must also be told if it is analyzing the same patient or the images are from different patients.

7. Orthanc Server (1) will be able to anonymize the DICOM file in an irreversible way.

8. Orthanc Server (1) will be able to handle the attachment of additional patient/client notes which are subject to change.

    a. Doctors often write additional information about a patient in their notes that is not part of DICOM such as:
        i. If the patient is a smoker or coffee drinker.
        ii. Any additional symptoms
        iii. How the patient is feeling today
    b. These will be submitted with the DICOM image to each corresponding patient and the server must be able to connect these two pieces of data.

9. A medical IT staff should be able to have admin access to the file system in Orthanc server (1).

10. A medical IT staff should be able to manually add more patient information for each patient in the file system.

Functional requirements for each Milestone:

Peer Testing 1:

- User Sends DICOM file from one Orthanc server (Acting as x-ray machine) to another using DIMSE protocol.
- Orthanc Server 1 Receives the DICOM image and anonymizes it.
- The server sends an anonymized DICOM file to Orthanc Server 2
- Orthanc Server 2 is accessed by peers who then download the DICOM file.

Peer Testing 2:

- An Orthanc server (acting as x-ray) sends DICOM files to Orthanc server 1. Orthanc server 1 partitions the PHI and anonymized data.
- Orthanc server 1 creates a GUID as a crosswalk.
- Orthanc server 1 send anonymized DICOM files to Orthanc server 2.
- Peers download the anonymized data.

Final Product:

- X-ray is configured to send DICOM files after scan to Orthanc server 1. Orthanc Server 1 catalogs and cross-references similar data to make sure the patient is unique
- DICOM files get anonymized and sent to Orthanc server 2.
- Orthanc server 1 store PHI on a relational database.

Non-functional Requirements and Environmental Constraints

Security:

- The most important non-functional requirement is security. The security of the PHI is very important.
- Security is already handled by the Medical IT staff. The system is deployed on their private network.

- The main consideration regarding security is ensuring that non-anonymized PHI doesn't appear in the anonymized data set.

Availability:
- Availability is very important for Orthanc Server (1) and (2)
- The server must never be down during work hours for the workweek.
- Maintenance and updates can only occur on the weekends.

Scalability:
- The system should be easily expandable. Should not be a hard limit on the number of DICOM files we can receive.
- Currently, there are around 2 million files, should be able to handle much more
- Must be able to scale quickly to millions of more files
- The system should be able to be portable to other hospitals
- Must be able to handle a million requests per day

Database Performance:
- High read speed:
    - It is important for our database to have the fastest readability because we will be constantly cross-referencing data to see if a patient with the wrong id is already in the database.
    - We must also read quickly to determine if data is part of a new study.
- Easy deletion:
    - Deleting DICOM studies should be easy and quick.
- Database performance will have two primary metrics:
    - Latency
    - Throughput

Tech Stack

Programming language:

Python:
Pros:
- Python provides us with a massive advantage due to its vast collection of libraries.
  In reference to this project, Python offers libraries like Pydicom and plugin packages for Orthanc Server
- Quicker to write and use, can get protocol and anonymization software running faster (as opposed to C++).

Cons:
- Too high level
    - Sacrifice speed

    - Harder to read
- Libraries less customizable
- Less documentation
- Used less by community
- Harder to parallelize for optimal database reading speed

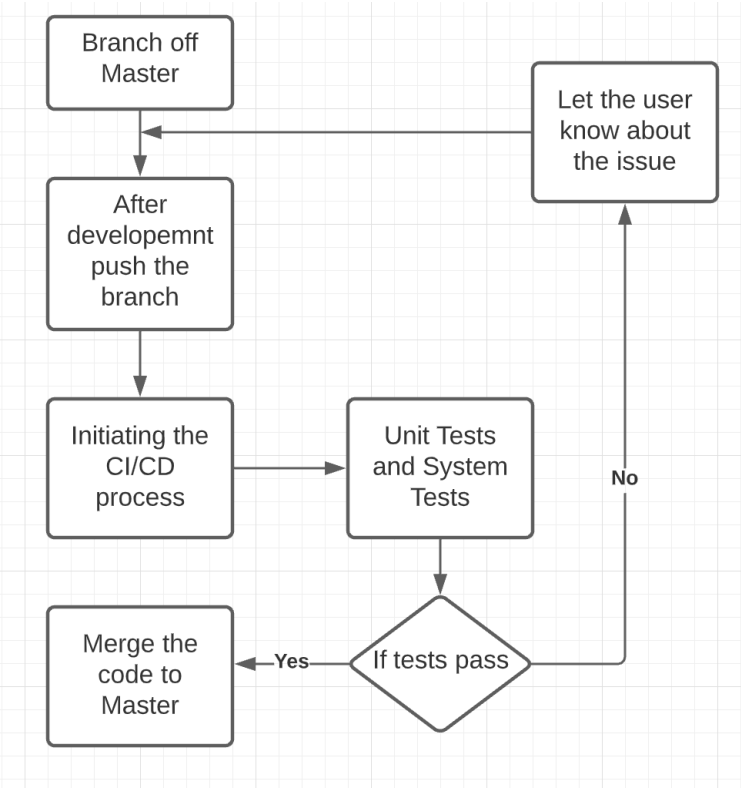
C++ (Recommended):
C++ is used in creating the Orthanc server. It will be simpler to use this language.
Pros:
- More documentation is available, as C++ is the primary language of Orthanc.
- Lower-level language:
    - Can optimize query speed.
- The direct source:
    - C++ is used to write the DICOM libraries for Python, C++ gives us more customization over those libraries.
- More reliable and easier to test.
- More common amongst the Orthanc community.
- Can use better error tracking software for logging errors (Elmah)

Cons:
- Longer to get up and running.
    - Not as easy as plug and play like Python.

Development Plan :

Our Development Plan will have the following steps involved:

1. We will make sure to always pull master before branching off to work on our individual tasks
2. After the development is complete the branch will be pushed to the remote repository and a Pull Request will be created
3. We are planning on implementing CI/CD(Continuous Implementation/Continuous Deployment) as soon as the branch is received in the remote repository all the unit tests will be run using the very same pipeline which will continue from CI/CD. Once all the tests are complete the code will ask for one last verification
4. The code has to be reviewed by every team member and the verification needs to be approved by everyone
5. Once the previous step is completed the code will automatically get implemented and deployed and will be ready for user testing.



Testing

Our testing will have two phases:

Unit Tests
These will involve testing individual methods, functions, and classes within the software. It involves using the smallest piece of code that can be isolated ensuring that they are not going to be the source of an issue.

For unit testing we will be using google test.

System Tests
There are some key high-level tests we must run that will account for edge cases. For example

1. Insert patient with the same patient ID but different PHI.
2. Inserting a patient with a different patient ID but same PHI.

These cases will be flagged with some sort of indication of their issues, and will have to be manually resolved.

Q&A

- **Will you utilize any kind of testing framework?**
    - We will be utilizing google test which is a unit test framework for C++

- **Regarding the non-functional requirement, what measures will be in place to ensure the security of the patient data?**
    - The measures are mostly in place by the hospital already.
    - It is a secure network with their private hardware, the only way the security of our application would be compromised is if the network was exposed. This would be unrelated to our application, and a serious security breach for the hospital.

- **What is the data scale that your application will have to run at? How large are these files and how long (worst case) would it take for your system to run through a set of these images? Do you think you'll have to consider writing portions of your code in something like C to take advantage of it's speed to handle large amounts of data? Do you anticipate other bottlenecks in the speed of your system?**
    - DICOM file size is 500 MB per mammogram approximately.
    - Sending and receiving images are handled by the DICOM protocol itself.
    - Images are not sent at a large enough volume to be concerned about. There will not be more than 100,000 thousand requests per day. Also should the system not keep up there is a queue for each DICOM image if the images are not being received.
    - C++ should suffice but if there is a need for C we will write certain portions to take advantage of speed.
    - A possible bottleneck is the image cross-referencing, we are unsure what the maximum amount of cross-referencing is that we can do.

- **Having every team member review every pull request seems like a potentially slow process, is there a way your team is planning to mitigate that?**
    - No this is not changing. This is the industry standard for high uptime applications. All code must be reviewed, no questions asked. The system cannot afford to be down, and everything that is built must be completely understood by all members in the team.

- **What steps will you take during the testing process to make sure that changes don't affect code that has already been tested? Have you looked into regression testing?**
    - For every code push, all test cases must pass first. We will be using a specific pipeline that will automate this.

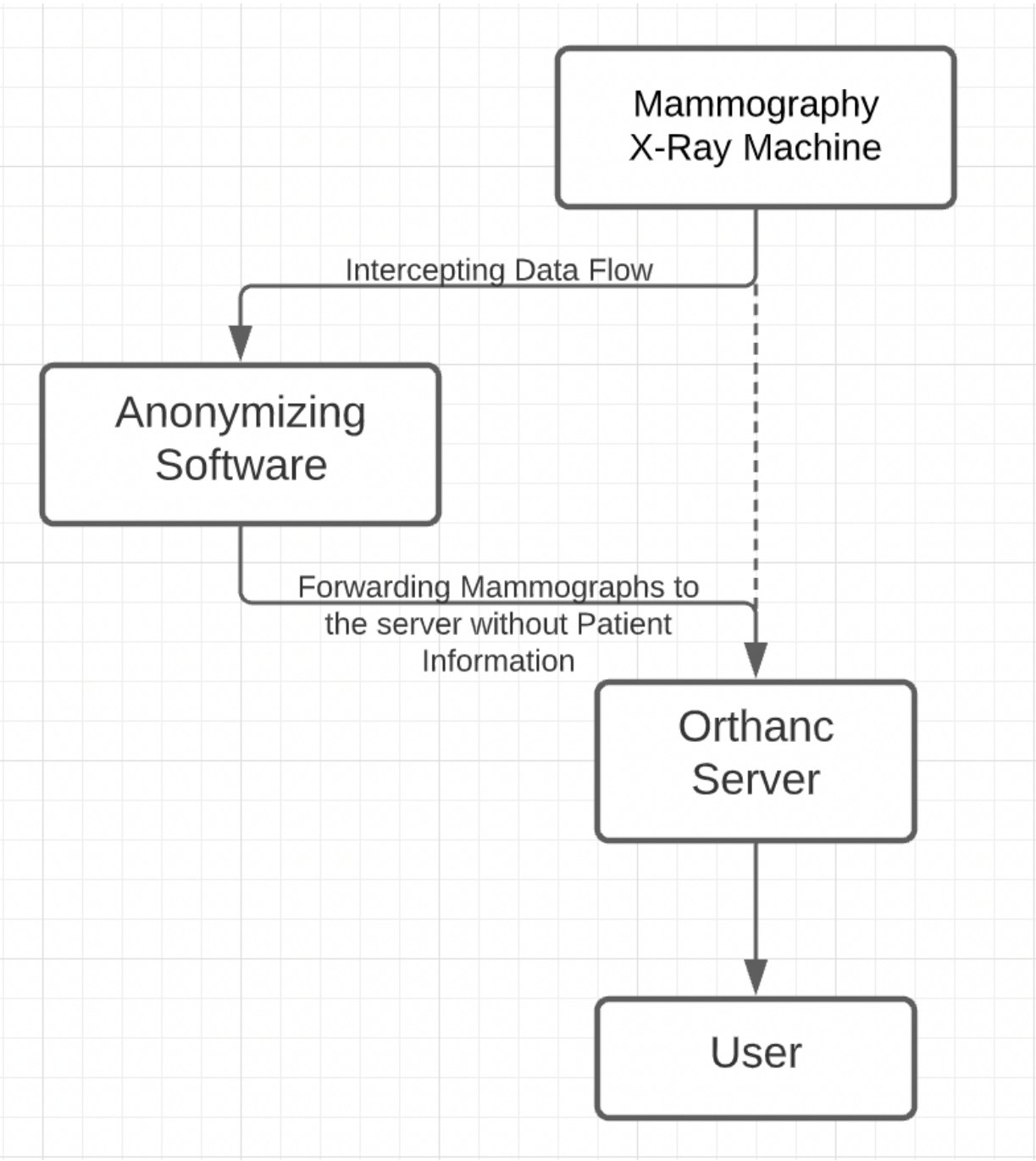
- **Do you have a plan for how you will be measuring the database performance?**
    - Throughput and latency will be the key metrics.

- How will the system know if the data is repeated? In the video it said it would see if the 'image' appears in another study, but how will they be compared? Furthermore, the video says "If there are more images coming from a study, therefore the study should not be available for researchers to

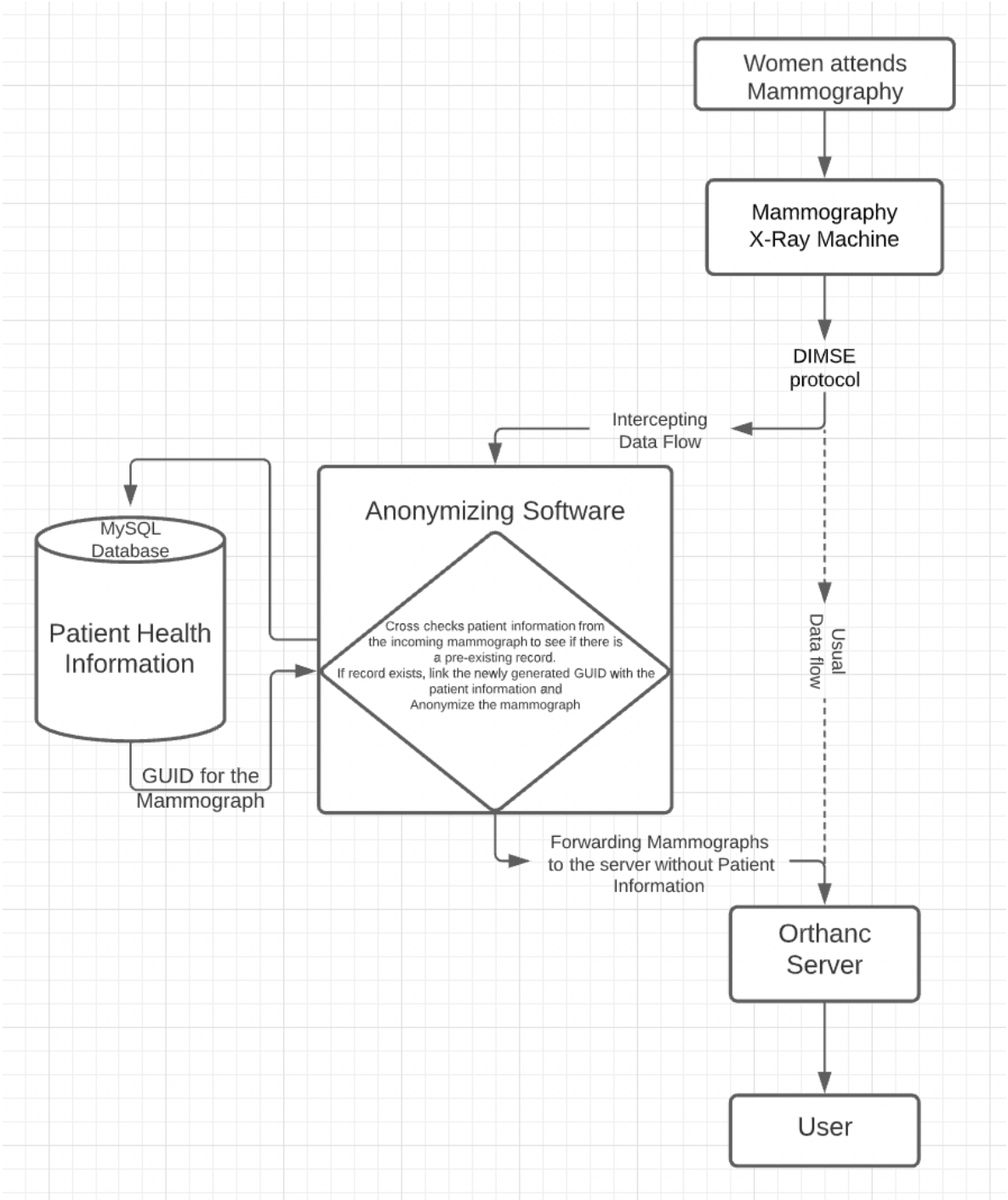
download.” I believe the video does not explain this properly. For example, will all the data present in one study not be available to another, or if there is repeated data the whole set will not be available?

Appendix:

DFD Diagram Level 0:


Mammography
X-Ray Machine
Intercepting Data Flow
Anonymizing
Software
Forwarding Mammographs to
the server without Patient
Information
Orthanc
Server
User

Level 1 DFD MySQL:



Level 1 DFD PostgreSQL: