Design Document

Computer Science Capstone Project

AI FOR CHEST X-RAY READ

Group Number 15

Supervisor: Dr. Mehdi Moradi
Email: moradm4@mcmaster.ca
Organization: McMaster University, CAS

NAME	**MacID**	**Student No.**
Suvansh Dutt	dutts1	400395711
Yuvraj Singh Sandhu	sandhy1	400367208
Ujjwal Raj	raju1	400397330
Suhaas Parcha	parchas	400420817

January 24, 2025

1.0 Revision History

Version Number	Authors	Description	Date
0	● Suvansh Dutt ● Yuvraj Singh Sandhu	Started Initial Draft	January 17th, 2025

	• Suhaas Parcha • Ujjwal Raj		
1	• Suvansh Dutt • Yuvraj Singh Sandhu • Suhaas Parcha • Ujjwal Raj	Finalized Design Doc	April 2nd, 2025

2.0 Introduction

2.1 Purpose Statement

In the field of Chest X-Ray imaging, radiographers play a crucial role in capturing X-ray images, while radiologists are responsible for analyzing these images to detect abnormalities, injuries, or diseases. However, this manual analysis process can be time-consuming and potentially prone to errors. We are implementing an AI model which would act as a second opinion for patients. The main objectives are to Identify negative results or detect the presence of specific diseases, reduce the time-consuming nature of manual analysis and minimize the potential for human error.

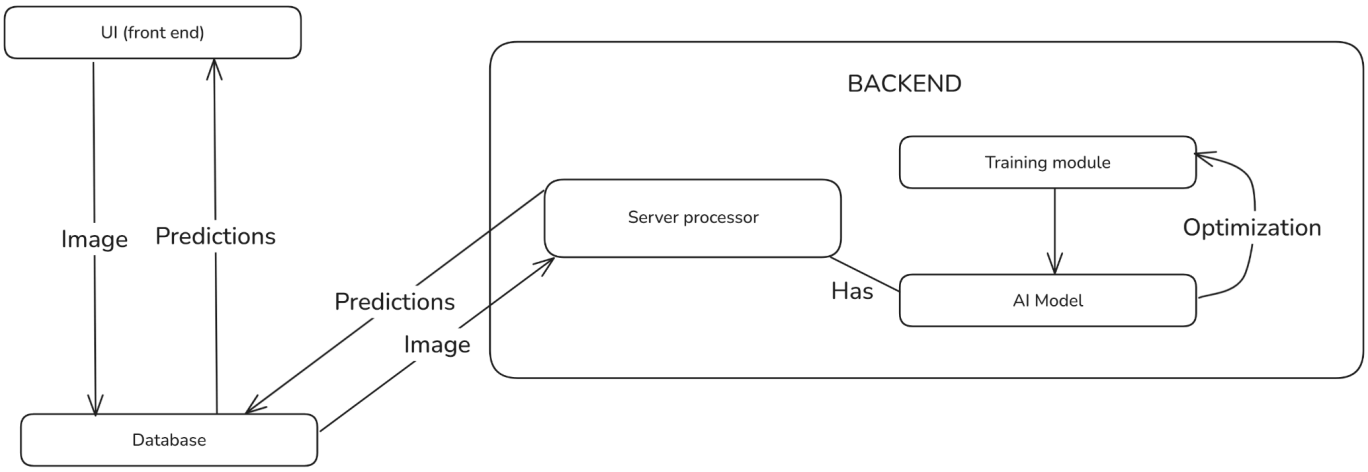
2.2 Document Purpose

The purpose of this document is to offer a comprehensive and detailed analysis of the system architecture. It aims to provide an in-depth description of all system components, illustrate the relationships between these components, demonstrate how each component aligns with and fulfills the requirements outlined in the Software Requirements Specification (SRS) document. This architectural overview serves as a bridge between the conceptual requirements and the practical implementation, ensuring that all stakeholders have a clear understanding of how the system is structured to meet its intended functionality and performance goals.

2.3 Project Scope

The application features a patient-centric web interface designed to provide accessible and informative second opinions on chest X-ray diagnosis. This user-friendly platform serves as a crucial bridge between advanced AI technology and patients seeking additional insights into their health. The key features of this project are: Secure patient portal, Historical diagnosis viewer and easy upload x-ray system.

3.0 Project components Diagram



3.1 Process

- ➔ The user uploades an Image to the UI (front end).
- ➔ The image is sent to the database which stores it and makes a call which sends the image to the backend (Server processor).
- ➔ The backend, which is another server, contains the AI model which will process the image and return a probability report.
- ➔ The prediction is sent back to the database to store.
- ➔ The database makes a call to the frontend, the final prediction would then be sent to the front end and displayed to the user.

4.0 Relationship Between Project Components and Requirements

System Component	Requirements
UI	P0: Develop a website for uploading chest X-ray images ● Develop a drag and drop interface for easy X-ray image upload ● Added Account validation and authorization. ● File type validation to only accept images of certain types ● Create an interactive overlay ● Display the Heatmap generated on top of the chest x-ray to show key areas of interest. P1:

	● Added a disclaimer so that the users do not blindly believe the AI predictions. ● Developed a dashboard for users to view their analysis history P2: ● Developed a mobile friendly version P3: ● Implemented a theme switch between light and dark mode ● Adding support for DICOM, jpeg, png images
Database	P0: ● Encrypted password and email ● Save previous scans and reports pertaining to each individual user Non functional : Privacy and security of the user is maintained.
AI model / Server processor	P0: Trained, analyzed and validated a convolutional based model ● For disease detection for a subsets of diseases found in chest X-rays P1: Disease Localization - Generate a heatmap which would highlight key areas of interest.
Training Module	P0: Detect Diseases with 80% accuracy P1: Achieve 0.8 AUC-ROC, 0.8 AU-PRC

5.0 Project components

5.1 User Interface

Normal behaviour

A user would be able to login and be directed to a dashboard. The dashboard would contain options to drag and drop or upload files (which are images) from their local machine. Upon completing the scan, the user would be shown results of the images. The dashboard would also contain a history of previous scans performed by the user, clicking on these images would show results of the scan accordingly.

API and structure

The signup/login credentials are sent to the database to store/verify. The input would be images of a certain format (jpg, jpeg, png). These images would be sent to the server processor to be analyzed by the AI model.

Implementation

Svelte framework is used to build the website. Additionally, HTML, CSS, Tailwind and Javascript are used to build the webpages. The main page which is called homepage.svelte is used to login or signup. After signing up, the user credentials are stored in the database. After logging in on the login.svelte, the user can view their history, upload a new chest X-ray image on the dashboard page. After uploading the image, it is sent to the database where it is stored. The database makes an API call to the server processor which infers the image and sends back the predictions to the database which stores it and sends it to UI for displaying.

Potential undesired behaviours

The website is hosted on a 3rd party platform, the website could experience downtime or slower response times due to factors outside of our control. Currently, the upload system is honor based. We don't have checks in place to actually determine if the image uploaded is valid and/or relevant to X-rays.

5.2 Database

Normal behaviour

At the beginning of the process pipeline, the database would respond to the calls made by the UI and store the images properly. Similarly, it would respond to the calls made by the backend; fetch the image and send it to the server processor. After the AI model processes the image, the resultant predictions would be sent to the database to store them. Later it would respond to the UI's fetch query and send the predictions to the front end.

API and structure

The database communicates to every part of the process elements.
Endpoints:

- api/upload (POST)- Input: image, Output: image_id
    - Stores the image in a database and returns a unique image_id
- api/upload (GET)- Input: image_id, Output: image and metadata
    - Retrieves the image and metadata for processing
- api/upload (Save_predictions) - Input: predictions, Output: image_id

    - save the processed predictions into the database
- api/upload (Fetch_predictions) - Input: image_id, Output: predictions
    - Retrieves predictions for the given image_id
- api/auth (POST) - Input: Str : Unique_username, Str: Hashed_password
    - Sends a check request to authenticate the user
- api/auth (GET) - Output: JSON: response
    - An okay or error response are invoked based on the legitimacy of the POST authentication request.

These endpoints and API calls would be used to progress the pipeline.

Implementation

The implementation of the database is mentioned in the schema charts in the appendix column.

Potential undesired behaviours

Similar to the front end, the database is hosted on 3rd party apps which could be unresponsive or down, this behaviour is unpredictable.

5.3 Server processor

Normal behaviour

The system is made to communicate with an AI model, a database, and endpoints that control the chest X-ray image's lifecycle, from retrieving the image to making predictions and returning them to the database.

API and structure

The server processor communicates with the database and already contains the trained AI model.
Endpoints:

- api/upload (Post) - Input: Image_id, Output : image and metadata
    - Receives the image from the database.
- api/upload (generate_predictions) - Input : image_id, Output : vector of probabilities.
    - Uses the AI model to generate predictions of a chest X-ray image
- api/upload (send_predictions) - Input : Vector of probabilities, Output : Vector of probabilities.
    - Sends predictions back to the database for that corresponding image.

These endpoints and API calls would be used to progress the pipeline.
The server processor would receive the images from the database. If the image is DICOM, the server processor extracts the jpg from the image, this image would later be sent as an input to the AI model. It would receive a vector of probabilities from the AI model and send them to the database.

Implementation

The server model would contain 2 classes. It would contain the AI model itself and the training module.
The processor will be hosted on AWS cloud servers. It will use python to process everything and make further API calls.

Potential undesired behaviours

Server could be down as it will be hosted on third party services.

5.4 AI model

Normal behaviour

Following training, the model is used for inference, which allows it to react to prediction requests in real time. For detection or classification, the model should process each image independently, and the API should manage several concurrent queries.

API and structure

Depending on the type of image, DICOM would be processed into a jpg, would be the input, the output would be a vector of probabilities which indicate the probability of a certain disease being present in that image. The AI model file will be stored in the server process to be used during inference.

Implementation

The model will be trained using the Stanford cheXpert dataset (400GB) using pytorch lightning as the framework during training. The images are divided into 80% train and 20% test split, A CSV file contains the labels for each image. We have created a custom dataset class called “CheXpert_Dataset.py”. This class takes in the root directory of the CSV file and the image folders. This class converts the images and labels to tensors and joins them to make a tuple. We have created different classes for each model architecture like DenseNet, ResNet and EfficientNet. We are constantly improving the model based on the testing metrics. The training module is defined in “train.py”. It takes in the hyperparameters like epochs, batch size, learning

rate, etc. The training module uses Adam as an optimizer. The training module loops over the image and label tuples in the data loader in batches. After the training is complete the model is saved and tested by the test class.

Potential undesired behaviours

Inaccurate predictions (false positives and false negatives) due to lack of enough training, class imbalancing, etc.

6.0 User Interface Details

The UI consists of a homepage, signup page, login page, about page, contact page and the user dashboard. The homepage would be an entry point offering a brief overview of the project's purpose. First time users would have to create an account through our signup page. Returning users can proceed to the login page, where they authenticate themselves using valid credentials. Once logged in, users can either see their upload history or upload another image and receive feedback on it. The about and contact page contains information for our project. We tried to follow Don Norman's design principles to ensure a seamless experience for the user. After the AI generates predictions, the user can go the the results page to view the heatmap and predictions.

Affordances - The UI elements are clear on their functionality. Buttons like "Log-in", "Sign Up", "Upload Image" are styled to look clickable and the texts indicate their purpose.

Signifiers - The elements guide the user to required actions. Labels like " Enter your email", "Password" fields clarify their purpose and what is expected of the user

Constraints - Certain elements follow constraints for optimized performance. For example, the login restricts the users when invalid login credentials are used, the dropbox would restrict the user from uploading files that are not of recognized image extensions.

Visibility, **Discoverability** and **Consistency -** We adhere to visibility by using contrasting color for certain elements to stand out and capture the user's attention . Certain elements like buttons, links and feedback are kept in expected locations and adhere to discoverability clauses. We use the same color scheme across all the pages to ensure consistency.

7.0 Appendix

7.1 Database

```
{
        "_id": "ObjectId", "Unique : True";
        "_username": "String", " Unique : True" ;
        "user_pass": "String" ;
        “Full name” :
        "images": [
                { "filename": "BinDataString", "data": "BinData", "predictions": [float]}, “Date” :
        “Time”, “status”: “String, heatmap : “BindataString”}
}
```

Images - List of images and their predictions which is a list of diseases present.
_id, _username - Unique identifiers.
user_pass - Encrypted password.