CSIT 321 – Final Year Project

Deep Learning based COVID-19 X-ray Image Classification

Group - FYP-23-S3-07

Project Report

Name	UOW ID
Xie JiangFeng	7358453
Yin YingCheng	7083440
Zhang Boyu	7896505
Tan Wei Yang, Keith	7366644
Tan Jing Zhi	7573261

Supervisor: Mr.Liaw Chun Huei

Assessor : Mr.Japit Sionggo

Contents

1. Project Background

1.1 Executive Summary

The rise and profound impact posed by COVID-19 has globally affect millions in a short span. Therefore, there is a need for more rapid methodologies to diagnose the disease accurately and efficiently. The current diagnostic methods have faced challenges in terms of speed, accuracy, and scalability, which are largely physical diagnosis with very little form of automation. This project aims to address the issue by using Machine Learning to enhance accuracy and robustness of COVID-19 diagnoses, by making such technologies available to the masses.

The project aims to design an advanced algorithms to diagnosis the disease, enabling faster and more accurate identification of the conditions through X-ray lung images. The primary focus will be on utilising neural networks and image processing algorithms to extract information from medical images, such as X-rays or CT scans, which are commonly used in COVID-19 diagnosis.

1.2 Objectives of the project

The main objective of this project is to develop an advanced Machine Learning model that can greatly improve accuracy and efficiency of COVID-19 diagnoses. By examining existing deep learning models used in medical imaging and diagnosis, we identify the strengths and limitations to gain better insights and improvise our own model for this project. The analysis will serve as a foundation for crafting an improved model, by incorporating the most effective features while mitigating potential shortcomings post by past usage or examples of similar implementation.

By integrating of cutting-edge techniques in neural networks and image processing, the enhanced model will be designed to extract intricate patterns and relevant information from medical images related to COVID-19. To achieve higher accuracy in diagnosis, the model aims to contribute significantly to the global battle against the pandemic, assisting healthcare professionals or the general public by making informed decisions and ensuring timely interventions and detection for positive patient outcomes. The model will provide a good benchmark for COVID-19 diagnosis but it is still recommended to consult a healthcare professional or use alternative methods of testing to verify the person's contraction with the disease.

1.3 Target Audience

The project is tailored for healthcare professionals, specifically doctors, who will upload X-ray images for diagnosis using the model. Patients will have access only to view the reports generated by the doctor. Medical related personnel such as healthcare professionals will benefit from the model more from the accuracy and

the speed of detection, enabling them to make well-informed decisions and improve patient care.

1.4 Project Description

1.4.1 Purpose

Our ultimate goal is to create an advanced model that can revolutionize COVID-19 diagnosis, making it not only accurate but also incredibly efficient. By leveraging the potential of these cutting-edge technologies, we envision a future where the diagnostic process becomes faster, more reliable, and more accessible to all. The enhancement of the diagnostic process aims to contribute to the ongoing battle against the pandemic.

Goals of the project:

- Researching existing deep learning models for COVID-19 diagnosis.
- Develop an improved deep learning model for COVID-19 diagnosis.
- Evaluating the performance of the improved model.
- Deploy the improved model.

4 phases of the project:

1. Research and requirements
2. Data collection and preprocessing.
3. Model development, training and prototyping.
4. Model evaluation and deployment.

The project will be conducted using the following resources:

- Publicly available datasets of chest X-ray images of COVID-19 patients. (sites: Kaggle/ v7labs)
- Cloud computing platform for training and deploying the deep learning model. Google Colab is chosen as our model development platform due to GPU resources for image processing and collaborative functionality.
- The programming languages, Python will be used for our model development.

The project will introduce a significant contribution to the field of machine learning and deep learning. The improvement deep learning diagnosis of COVID-19 will better tackle this pandemic and lessen the spread.

1.4.2 Research Summary

Chest radiographs (CXRs) are widely used for the screening and management of COVID-19, the predominant pattern of lung abnormality was ground-glass opacity shown in the CXRs of COVID-19 patients. Ground-glass opacity (GGO)

is a non-specific term defined by the Fleischner society as the presence on high-resolution computed tomography (HRCT) of a hazy increase in lung density. It is confirmed that the ground-glass opacity and consolidation is the most common pattern across the CXRs of COVID-19 patients.

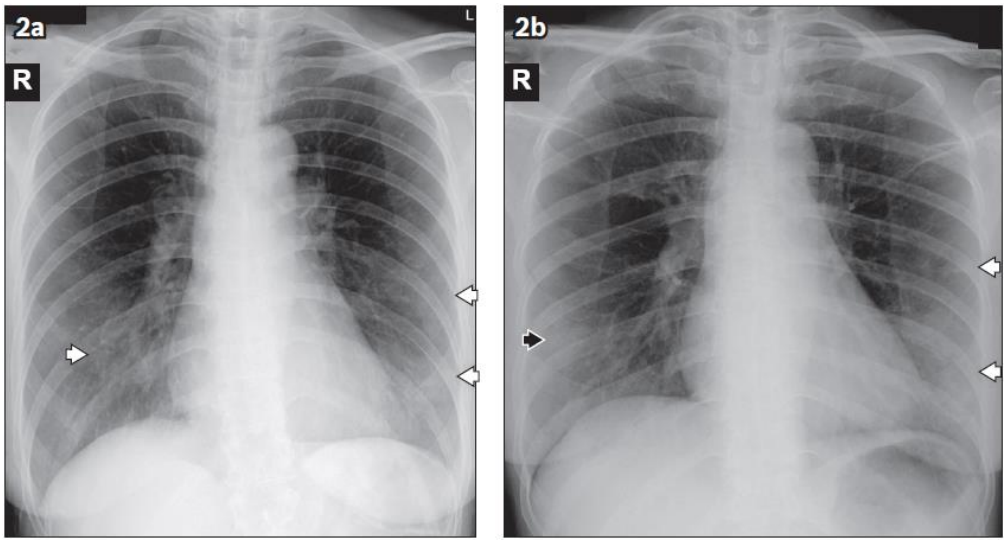


Fig. 2 A 55-year-old woman with fever, cough and dyspnoea for four days. (a) Initial chest radiograph obtained on Day 4 of symptoms shows multifocal ground-glass opacities in the bilateral lower zones (arrowheads). (b) Follow-up radiograph performed two days later shows increased density in the left lower zone (white arrowheads), in keeping with consolidation. A small focus of consolidation is also seen in the right lower zone (black arrowhead).

Since COVID-19 inflicts visible changes on the lungs of a patient, our machine learning model will be trained and tuned to recognise these changes to produce an accurate result. The data we have to use will be images of CXRs that will contain the features mentioned above. In image processing, a digital image can be regarded as a discrete function of a two-dimensional space, denoted as f (x, y). Thus, the machine learning model must be able to process 2-dimensional data. Training the algorithm will also take a long time as the machine will extract and learn from thousands of images, thus it is also important that the algorithm is fast and still accurate in feature selection.

1.4.2 Research Report

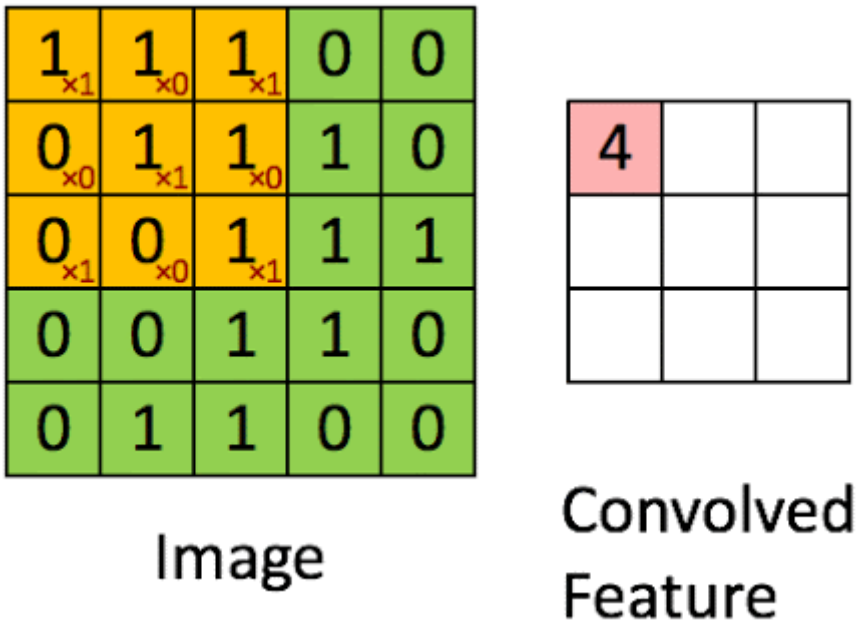
Experiments show that among Support Vector Machine(SVM), k-nearest neighbour(KNN), naive-bayes(NB) and Convolutional Neural Network(CNN)models, CNN classifier is higher than that of other classifiers in training set and test set. Additionally, compared with other commonly used classifiers, CNN has the highest accuracy, and the time spent is acceptable in the seven classifiers comparison.[3] Its built-in convolutional layer reduces the high-dimensionality of images while capturing their features for pattern-recognition and learning. Therefore, after researching, we will be

implementing the CNN model to carry out pattern-recognition and prediction on the CXRs images.

A digital image can be regarded as a discrete function of a two-dimensional space, denoted as f(x, y). Assuming the existence of a two-dimensional convolution function g(x, y), the output image z(x, y), can be represented by the following formula:

$$z(x, y) = f(x, y) * g(x, y)$$

In this way, the convolution operation can be used to extract the image features.



In this example, the yellow area represents the kernel used in CNN algorithm to extract features from the image. Red numbers in the kernel represents the user-defined weights, which can be tuned and will impact the overall convolved features.

In image recognition, each pixel of the image and its corresponding Red, Blue, Green (RGB) values are processed in the algorithm. Fortunately for us, CXRs are mainly grayscale images, meaning each pixel does not have an array of values but instead just a single value ranging from 0-255, where 0 denotes black and 255 denotes white.

Since it is mentioned above that the COVID-19 leads to ground-glass opacities in lungs, which can be observed in the CXR, we will train and tune the kernel and its weights to look for this feature effectively, so as to predict the presence of COVID-19 in the lungs shown in CXRs.

2. Project Proposal

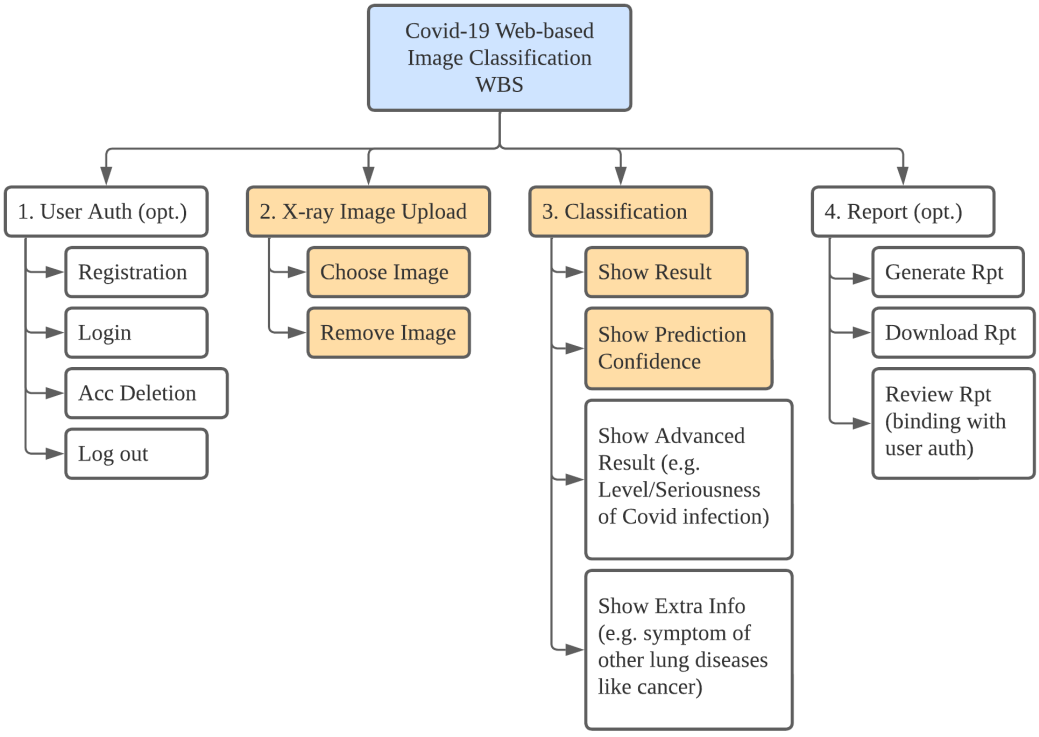
2.1 Software Development Methodology

We will be adopting the Scrum software development methodology to manage the development process. Scrum is an Agile framework emphasizing on collaboration, iterative development, and user feedback. Each sprint will consist of 7 days, it is such that the project is divided into short and manageable iterations, allowing for continuous progress and regular reviews.

Each member has clearly defined primary roles, which is project manager, Scrum Master and programmers for both frontend and backend to facilitate the process. A cross-functional working style is also adopted so that every group member will be involved in each process such as model development or front and backend development. Daily stand-up meetings will keep the team informed about progress and potential obstacles. At the end of each Sprint, there will be a Sprint Review, where stakeholders can provide feedback on the current deliverable work.

The iterative approach enables us to adapt to changing requirements and deliver increments of the product throughout the project's duration.

2.2 Work Breakdown Structure



The boxes filled with orange colour are the core functional requirements that need to be implemented with high priority which includes:

1) X-ray image uploading & removing
2) Image classification
3) Result producing & displaying

It will be desirable but not mandatory that to have:

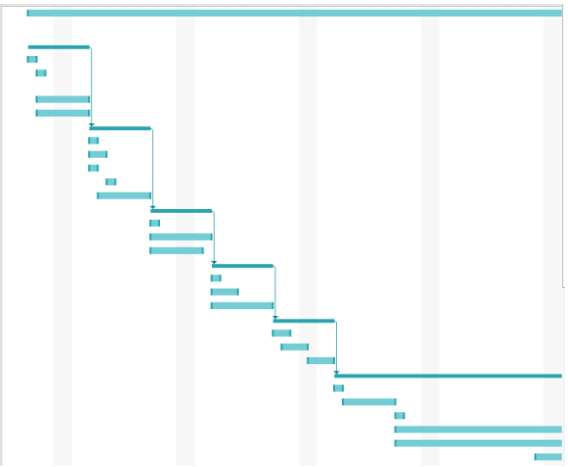
1) User Authentication & Authorization
2) Advanced result generation
3) Ability to identify other diseases besides Covid-19
4) Report generation & download
5) View previous reports

2.3 Project Deliverable Dates

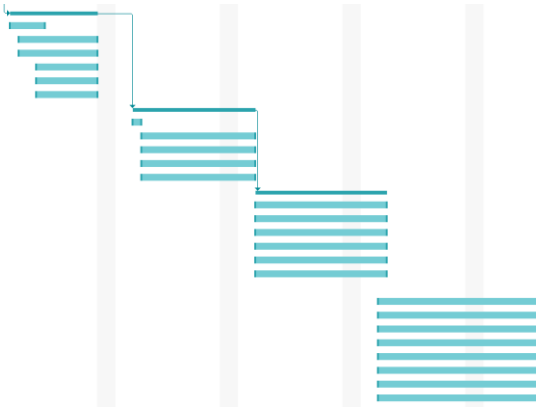
Title	Submission Date
Project requirements documentation	12 August 2023, 9:00 pm Singapore time
Project Prototype slides	23 September 2023, 9:00 pm Singapore time
Project Progress Report	23 September 2023, 9:00 pm Singapore time
Final product and documentation	18 November 2023, 9:00 pm Singapore time
Reflective Diary	18 November 2023, 9:00 pm Singapore time
Final Presentation	25 November 2023, 9:00 pm Singapore time

2.4 Project Schedule

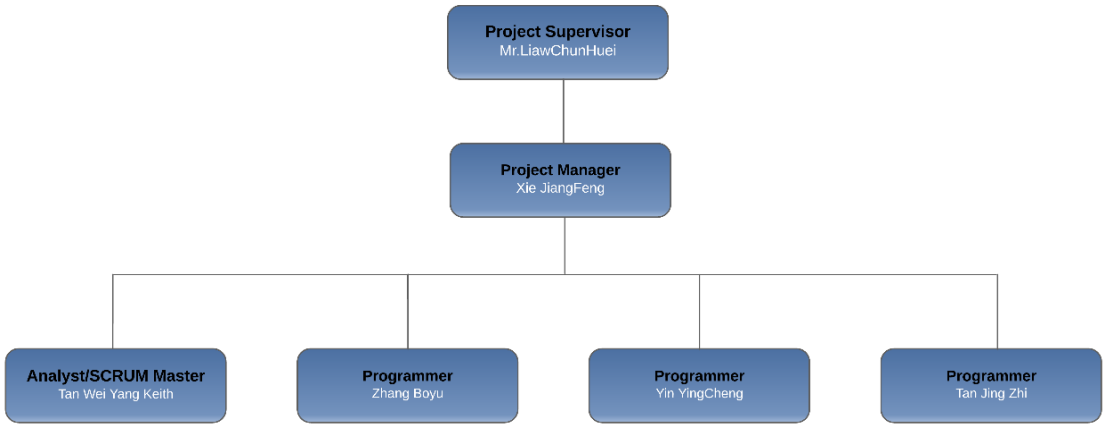
Deep Learning based COVID-19 X-ray Image Classification	Wed 12/7/23	Sat 18/11/23	94 days
Spint 1. Requirement and Env setup consolidation			
Member introduction	Wed 12/7/23	Wed 12/7/23	1 day
Set up working environment(Google Colab, Share folders)	Thu 13/7/23	Thu 13/7/23	1 day
Research on relevant databases	Thu 13/7/23	Tue 18/7/23	4 days
Research on project requirements/specifications	Thu 13/7/23	Tue 18/7/23	4 days
Sprint 2			
Outline for project requirements(headers and pointers)	Wed 19/7/23	Wed 19/7/23	1 day
Analysis on research	Wed 19/7/23	Thu 20/7/23	2 days
Assign roles and responsibilities	Wed 19/7/23	Wed 19/7/23	1 day
Set up Taiga	Fri 21/7/23	Fri 21/7/23	1 day
Developing project requirement documentation	Thu 20/7/23	Tue 25/7/23	4 days
Sprint 3			
Review and validation of the documentation	Wed 26/7/23	Wed 26/7/23	1 day
Develop project requirement documentation	Wed 26/7/23	Tue 1/8/23	5 days
Preparation for the supervisor's meeting	Wed 26/7/23	Mon 31/7/23	4 days
Sprint 4			
Review and validation of the documentation	Wed 2/8/23	Wed 2/8/23	1 day
Improvise on supervisor feedback	Wed 2/8/23	Fri 4/8/23	3 days
Develop project requirement documentation	Wed 2/8/23	Tue 8/8/23	5 days
Sprint 5			
Complete activity diagram	Wed 9/8/23	Thu 10/8/23	2 days
Review and validation of the documentation	Thu 10/8/23	Sat 12/8/23	3 days
Prepare PRD submission	Sun 13/8/23	Tue 15/8/23	3 days
Sprint 6			
Review and validation of the documentation	Wed 16/8/23	Wed 16/8/23	1 day
Content finalizing	Thu 17/8/23	Tue 22/8/23	4 days
PRD Report submission	Wed 23/8/23	Wed 23/8/23	1 day
Frontend prototype development	Wed 23/8/23	Wed 13/9/23	16 days
Backend prototype development	Wed 23/8/23	Wed 13/9/23	16 days
Preparation for the supervisor's meeting	Fri 8/9/23	Wed 13/9/23	4 days



Sprint 7			
Improvise on supervisor feedback	Wed 13/9/23	Sat 16/9/23	4 days
Frontend prototype evaluation	Thu 14/9/23	Fri 22/9/23	7 days
Model development	Thu 14/9/23	Fri 22/9/23	7 days
Database development	Sat 16/9/23	Fri 22/9/23	6 days
Prototype slides preparation	Sat 16/9/23	Fri 22/9/23	6 days
Prototype report preparation	Sat 16/9/23	Fri 22/9/23	6 days
Sprint 8			
Review of presentation	Wed 27/9/23	Wed 27/9/23	1 day
Frontend development	Thu 28/9/23	Tue 10/10/23	9 days
Backend development	Thu 28/9/23	Tue 10/10/23	9 days
Database development	Thu 28/9/23	Tue 10/10/23	9 days
Model refinement	Thu 28/9/23	Tue 10/10/23	9 days
Sprint 9			
UI/UX refinement	Wed 11/10/23	Wed 25/10/23	11 days
Integration of frontend and backend	Wed 11/10/23	Wed 25/10/23	11 days
Frontend development	Wed 11/10/23	Wed 25/10/23	11 days
Backend development	Wed 11/10/23	Wed 25/10/23	11 days
Database development	Wed 11/10/23	Wed 25/10/23	11 days
Technical documentation	Wed 11/10/23	Wed 25/10/23	11 days
Sprint 10			
Finalising technical documentation	Wed 25/10/23	Fri 17/11/23	18 days
Finalising frontend development	Wed 25/10/23	Tue 14/11/23	15 days
Finalising backend development	Wed 25/10/23	Tue 14/11/23	15 days
Finalising database development	Wed 25/10/23	Tue 14/11/23	15 days
Integration of codes	Wed 25/10/23	Wed 15/11/23	16 days
Tesing	Wed 25/10/23	Thu 16/11/23	17 days
Deployment	Wed 25/10/23	Thu 16/11/23	17 days
Prepare for submission	Wed 25/10/23	Sat 18/11/23	19 days



2.5 Project Organization



3.6 Roles and Responsibilities

Name	Role
Mr.Liaw Chun Huei	Project Supervisor
Xie JiangFeng	Project Manager
Tan Wei Yang Keith	Analyst/SCRUM Master
Zhang Boyu	Programmer
Yin YingCheng	Programmer
Tan Jing Zhi	Programmer

Project Supervisor Role:

- Provides oversight and guidance to the Project Manager and team.

- Approves project plans and major decisions.
- Monitors progress and mitigates risks.

Project Manager Role:

- Responsible for project planning and execution.
- Defines objectives, scope, and deliverables.
- Facilitates team communication and coordination.
- Ensures adherence to project methodologies.

Analyst/SCRUM Master Role:

- Gathers and analyses project requirements.
- Prioritizes tasks and features.
- Facilitates Agile processes and meetings.

Programmer Role:

- Assists in data preprocessing.
- Implements deep learning algorithms.
- Participates in code reviews and knowledge sharing.
- Implements deep learning algorithms.

2.7 Roles and Responsibilities Matrix

Roles / Names	Xie JiangFeng	Yin YingCheng	Zhang Boyu	Tan Wei Yang, Keith	Tan Jing Zhi
Project Leader	☒				
Documentation	☒	☒	☒	☒	
Frontend Development	☒	☒	☒	☒	
Backend Development	☒		☒	☒	☒
Wireframe		☒			
Testing			☒		☒

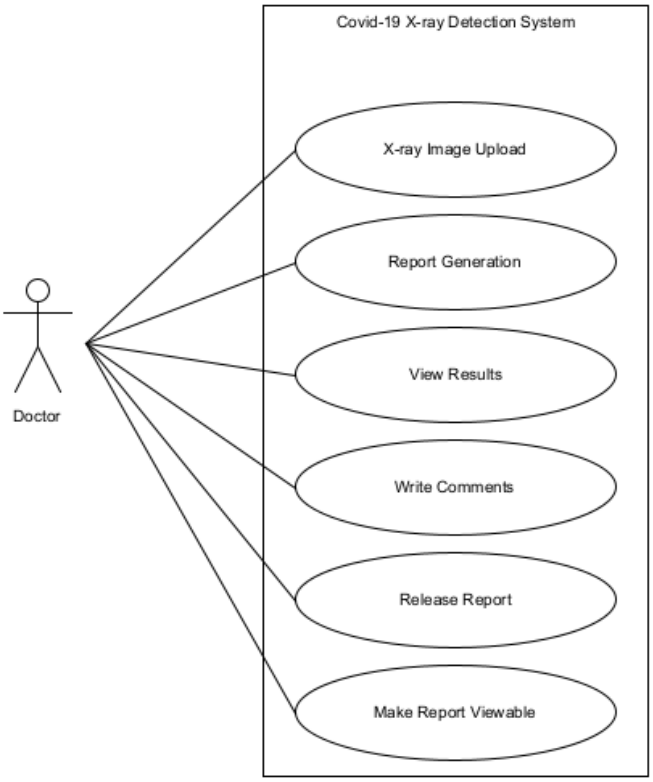
3. Product Requirement Specifications

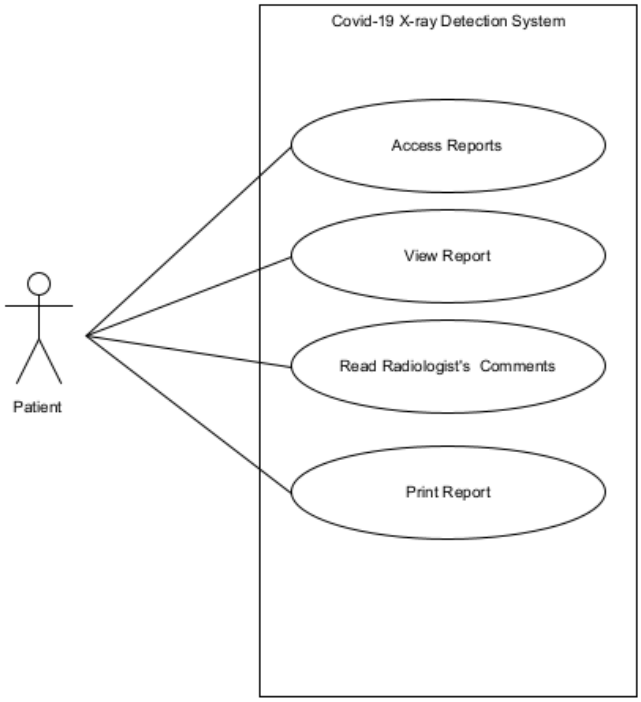
3.1 User stories

Doctor	
1	As a doctor, I want to upload X-ray images of patients, so that I can use the backend model to diagnose COVID-19 based on the X-ray images.
2	As a doctor, I want the model to generate a report with the COVID-19 diagnosis result (positive/negative), so that I can communicate the findings to the patient accurately.
3	As a doctor, I want to see the results of the machine learning model, so that I can make a diagnosis.
4	As a doctor, I want the ability to write comments or feedback on the generated report, so that I can provide additional insights or explanations to the patient.
5	As a doctor, I want to make the report accessible to the patient, so that he/she can review the results, provide feedback and ask questions.
6	As a doctor, I want to make the report viewable to the patient, so that he/she can gain additional insights on his/her diagnosis.
Patient	
7	As a patient, I want to be able check the status of my diagnosis report so that I can track the progress.
8	As a patient, I want to view the report released by the doctor's model, so that I can know if the X-ray indicates COVID-19 or not.
9	As a patient, I want to read the comments or feedback written by the doctor, so that I can better understand the diagnosis.
10	As a patient, I want to print a report of their COVID-19 diagnosis so that I can have a hard copy of the diagnosis for further analysis.

4. System Design

4.1 Use Case Diagram





4.2 Use Case Description

Use Case Name: X-ray Image Upload	**ID: 1**
Stakeholders and goals: Doctor aims to upload X-ray images of patients to the website so that the backend model can diagnose COVID-19 based on the X-ray images.	
Description: Doctor uploads X-ray images of patients for COVID-19 diagnosis. The uploaded images will be processed by the backend model, which will generate a diagnosis report indicating whether the X-ray suggests COVID-19 or not.	
Actors: Doctor	
Trigger: The doctor accesses the website and initiates the process of uploading X-ray images.	
Preconditions: • The doctor has a valid account registered on the website. • The doctor has relevant X-ray images of patients stored on their device for upload.	
Normal flow: 1. The doctor selects the option to upload X-ray images for COVID-19 diagnosis. 2. The website prompts the doctor to choose the relevant X-ray images from their local storage. 3. The website processes the uploaded X-ray images and forwards them to the backend model.	
Sub-flows: None	
Alternative/Exceptional flows: None	

Use Case Name: Report Generation	**ID: 2**
Stakeholders and goals: Doctor wants the machine learning model to generate a report with the COVID-19 diagnosis result based on X-ray images, enabling accurate communication of findings to the patient.	
Description: Doctor uses the machine learning model to analyse the uploaded X-ray images and generate a detailed report with the COVID-19 diagnosis result. The generated report will indicate whether the X-ray images suggest a positive or negative diagnosis for COVID-19, providing valuable information for accurate communication with the patient.	
Actors: Doctor	

Trigger: The doctor initiates the process of generating the COVID-19 diagnosis report after uploading the X-ray images.
Preconditions: • The doctor has successfully uploaded X-ray images of the patient to the website. • The uploaded X-ray images have been processed by the backend model.
Normal flow: 1. The doctor chooses the X-ray image for which they want to generate the COVID-19 diagnosis report. 2. The machine learning model processes the selected X-ray image and generates a detailed report with the COVID-19 diagnosis result.
Sub-flows: None
Alternative/Exceptional flows: None

Use Case Name: View Results	**ID: 3**
Stakeholders and goals: Doctor wants to view the results of the machine learning model's analysis of X-ray images, enabling them to make a diagnosis based on the provided information.	
Description: Doctor accesses the machine learning model's results, which are generated based on the analysis of uploaded X-ray images. The results will provide valuable information to the doctor, assisting them in making an accurate diagnosis for the patient.	
Actors: Doctor	
Trigger: The doctor initiates the process of viewing the results of the machine learning model after it has analyzed the X-ray images.	
Preconditions: • The doctor has successfully uploaded X-ray images of the patient to the website. • The uploaded X-ray images have been processed by the backend machine learning model.	
Normal flow: 1. The doctor chooses the X-ray image for which they want to view the machine learning model's results. 2. The website displays the results of the machine learning model's analysis for the selected X-ray image. The results may include the COVID-19 diagnosis (positive/negative) and any additional insights or details obtained from the model.	

Sub-flows: None
Alternative/Exceptional flows: None

Use Case Name: Write Comments	**ID: 4**
Stakeholders and goals: Doctor wants the ability to write comments or feedback on the generated report to provide additional insights or explanations to the patient regarding their COVID-19 diagnosis.	
Description: Doctor accesses the generated report and being able to write comments or feedback to improvise the COVID-19 diagnosis. The doctor may include additional details, recommendations, clarifications, or treatments in their comments to improve the patient's understanding of the diagnosis.	
Actors: Doctor	
Trigger: The doctor accesses the generated report and initiates the process of writing comments or feedback.	
Preconditions: • The doctor has reviewed the machine learning model's results for the patient's X-ray image. • The doctor has the necessary permissions to add comments to the report.	
Normal flow: 1. The doctor chooses the generated report for which they want to write comments or feedback. 2. The doctor enters their comments or feedback in a designated input area in the report. The comments may include additional insights, explanations, or recommendations related to the COVID-19 diagnosis.	
Sub-flows: None	
Alternative/Exceptional flows: None	

Use Case Name: Release Report	**ID: 5**

Stakeholders and goals: Doctor releases the generated report and their comments to the patient, enabling the patient to review the COVID-19 diagnosis results.
Description: Doctor finalizes the report and comments they have written and releases this report to the patient. The patient will be able to access the report and comments on the website to understand their COVID-19 diagnosis.
Actors: Doctor
Trigger: The doctor completes writing the comments on the report and initiates the process of releasing the report and comments to the patient.
Preconditions: • The doctor has successfully written comments on the generated report. • The patient has a valid account registered on the website.
Normal flow: 1. The doctor reviews and finalizes the comments on the generated report to ensure accuracy and completeness. 2. The doctor chooses the patient to whom they want to release the report and comments. 3. The doctor initiates the release of the report and comments to the selected patient.
Sub-flows: None
Alternative/Exceptional flows: None

Use Case Name: Make Report Viewable	**ID: 6**
Stakeholders and goals: Doctor wants to make their reports viewable to the patient, enabling the patient to gain additional insights on their COVID-19 diagnosis.	
Description: Doctor finalizes the comments and feedback they have written on the patient's diagnosis report. The doctor then releases these comments with the report to make them viewable to the patient.	
Actors: Doctor	
Trigger: The doctor completes writing comments and feedback on the patient's diagnosis report and initiates the process of releasing these comments to the patient.	
Preconditions: • The doctor has a valid account registered on the website.	

The doctor has successfully written comments and feedback on the patient's diagnosis report.
Normal flow: 1. The doctor will review the patient's diagnosis report to ensure accuracy and appropriateness. The doctor initiates the process to make the report viewable to the selected patient.
Sub-flows: None
Alternative/Exceptional flows: None

Use Case Name: Access Reports	**ID: 7**
Stakeholders and goals: Patient wants to check the status of their diagnosis report to track the progress and know when it becomes available for review.	
Description: Patient accesses the website to check the status of their COVID-19 diagnosis report. The status indicates whether the report is still under analysis or completed.	
Actors: Patient	
Trigger: The patient accesses the website and initiates the process of checking the status of their COVID-19 diagnosis report.	
Preconditions: • The patient has a valid account registered on the website. • The doctor has released the diagnosis report and comments for the patient.	
Normal flow: 2. The patient opens their web browser and logs in to their account on the website. 3. The patient accesses their dashboard and navigates to the section where diagnosis reports are stored. 4. The website displays the status of the COVID-19 diagnosis report, indicating whether it is still under analysis or completed.	
Sub-flows: None	
Alternative/Exceptional flows: None	

Use Case Name: View Report	**ID: 8**
Stakeholders and goals: Patient wants to view the report released by the doctor to determine if the X-ray indicates COVID-19 or not.	
Description: Patient views the COVID-19 diagnosis report on the website. The report contains the results of the analysis performed by the doctor on the uploaded X-ray images, providing the patient with information about whether the X-ray indicates a positive or negative diagnosis for COVID-19.	
Actors: Patient	
Trigger: The doctor completes and releases the diagnosis report to the patient.	
Preconditions: • The patient has a valid account registered on the website. • The doctor has generated and released the COVID-19 diagnosis report for the patient.	
Normal flow: 1. The patient opens their web browser and logs in to their account on the website. 2. The patient accesses their dashboard and navigates to the section where diagnosis reports are stored. 3. The patient chooses the specific COVID-19 diagnosis report they wish to view from the available list. 4. The website displays the COVID-19 diagnosis report, and relevant information provided by the doctor.	
Sub-flows: None	
Alternative/Exceptional flows: None	

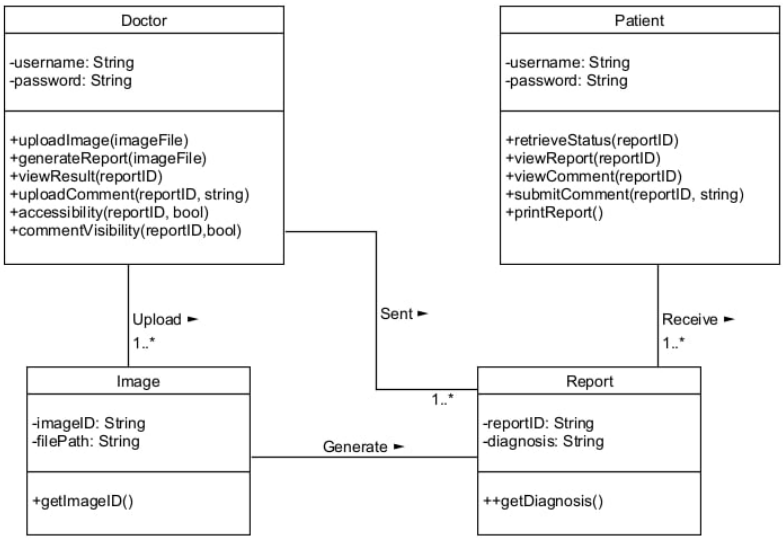
Use Case Name: Read Doctor's Comments	**ID: 9**
Stakeholders and goals: Patient wants to read the comments or feedback written by the doctor on their diagnosis report to gain a better understanding of the diagnosis.	
Description: Patient accesses the diagnosis report and reads the comments or feedback provided by the doctor. The comments may contain additional insights, explanations, or recommendations related to the COVID-19 diagnosis, helping the patient comprehend their medical condition more thoroughly.	

Actors: Patient
Trigger: The doctor writes and releases the comments, making them available for the patient to access.
Preconditions: • The patient has a valid account registered on the website. • The doctor has generated and released the COVID-19 diagnosis report for the patient.
Normal flow: 1. The patient opens their web browser and logs in to their account on the website. 2. The patient accesses their dashboard and navigates to the section where diagnosis reports are stored. 3. The patient chooses the specific COVID-19 diagnosis report they wish to view from the available list. 4. The website displays the diagnosis report along with the comments written by the doctor for the patient to read and understand the diagnosis in more detail. 5. A textbox will be displayed for the patient to write their comment and will be feedbacked to the doctor.
Sub-flows: None
Alternative/Exceptional flows: None

Use Case Name: Print Report	**ID: 10**
Stakeholders and goals: Patient wants to print a report of their COVID-19 diagnosis to have a hard copy for further analysis and reference	
Description: Patient accesses the diagnosis report and initiates the process of printing the report. Patient can print the report to keep for their records or share with other healthcare provider for further analysis and evaluation.	
Actors: Patient	
Trigger: Patient decides to print the COVID-19 diagnosis report after accessing the report on the website.	
Preconditions: • The patient has a valid account registered on the website.	

• The doctor has generated and released the COVID-19 diagnosis report for the patient.
Normal flow: 1. The patient opens their web browser and logs in to their account on the website. 2. The patient accesses their dashboard and navigates to the section where diagnosis reports are stored. 3. The patient clicks on the "Print" button, signaling their intent to print the diagnosis report.
Sub-flows: None
Alternative/Exceptional flows: None

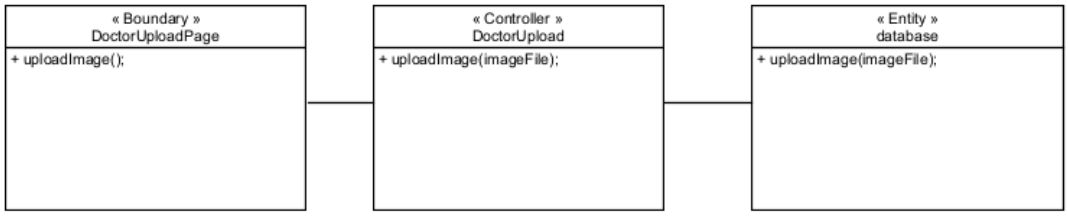
4.3 Class Diagram



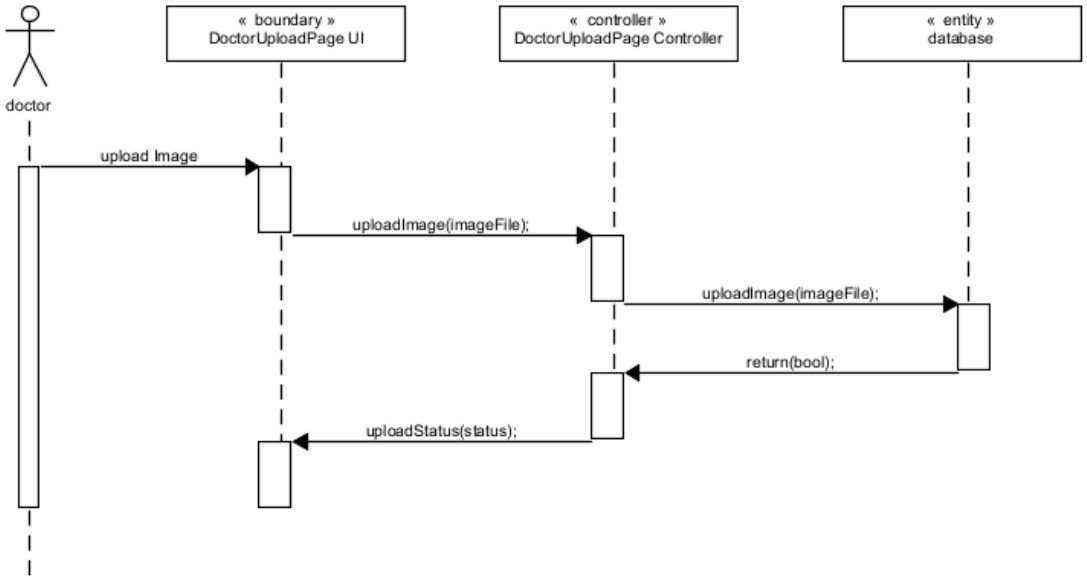
4.4 BCE and Sequential Diagram

User story1: As a doctor, I want to upload X-ray images of patients, so that I can use the backend model to diagnose COVID-19 based on the X-ray images.

BCE:

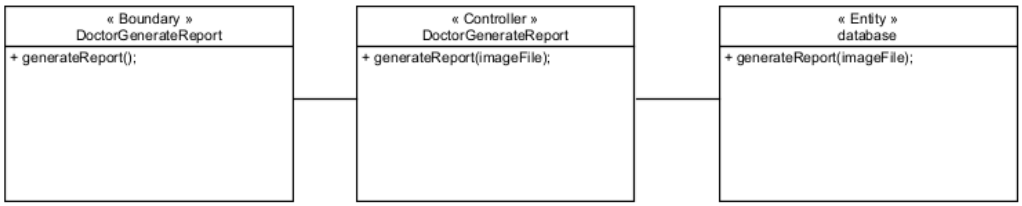


Sequential Diagram:

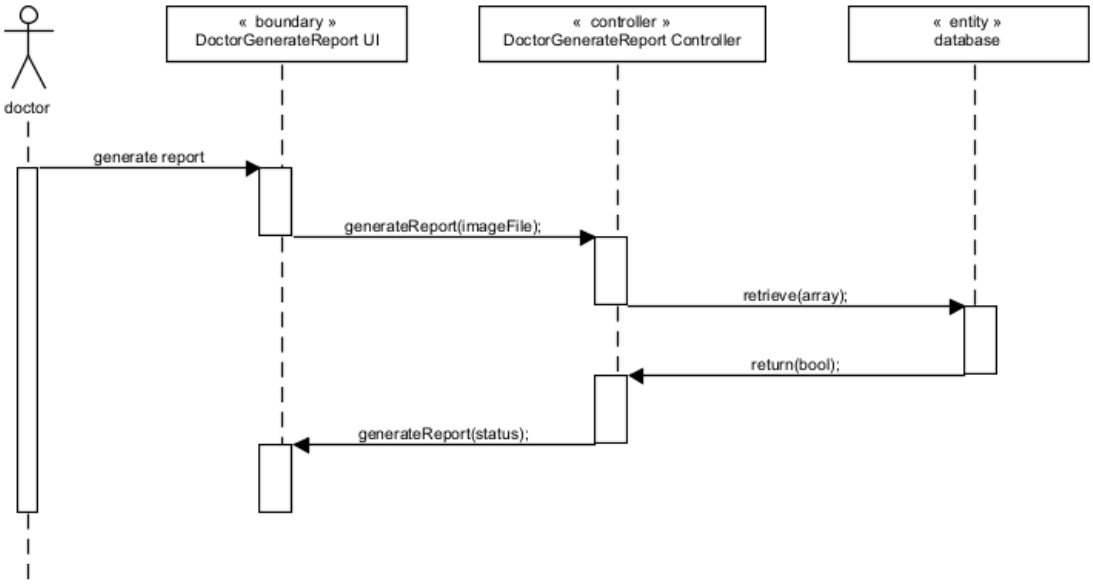


User story2: As a doctor, I want the model to generate a report with the COVID-19 diagnosis result (positive/negative), so that I can communicate the findings to the patient accurately.

BCE:

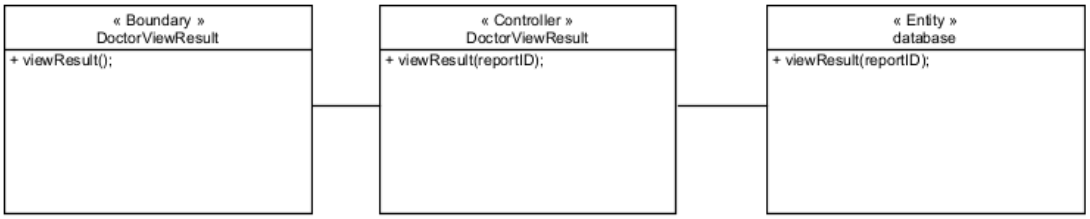


Sequential Diagram:

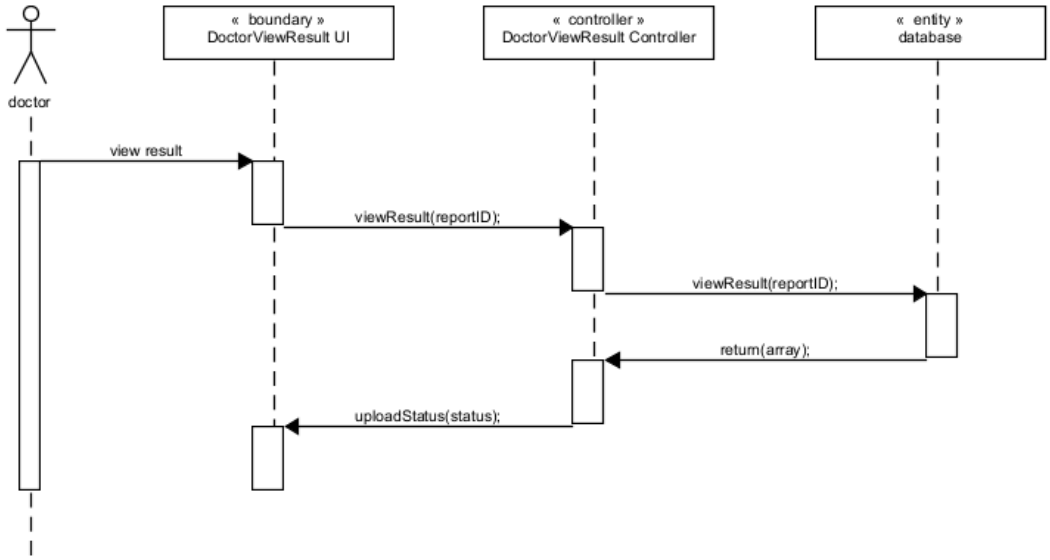


User story3: As a doctor, I want to see the results of the machine learning model, so that I can make a diagnosis.

BCE:

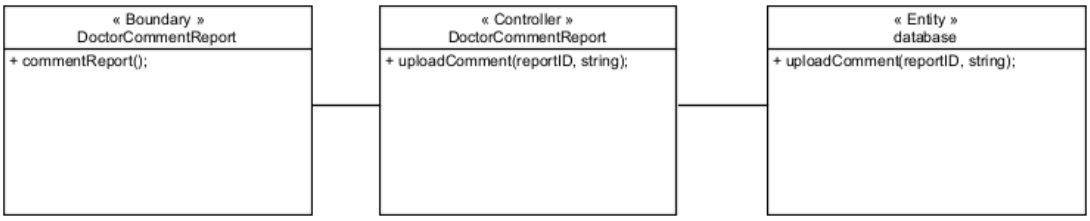


Sequential Diagram:

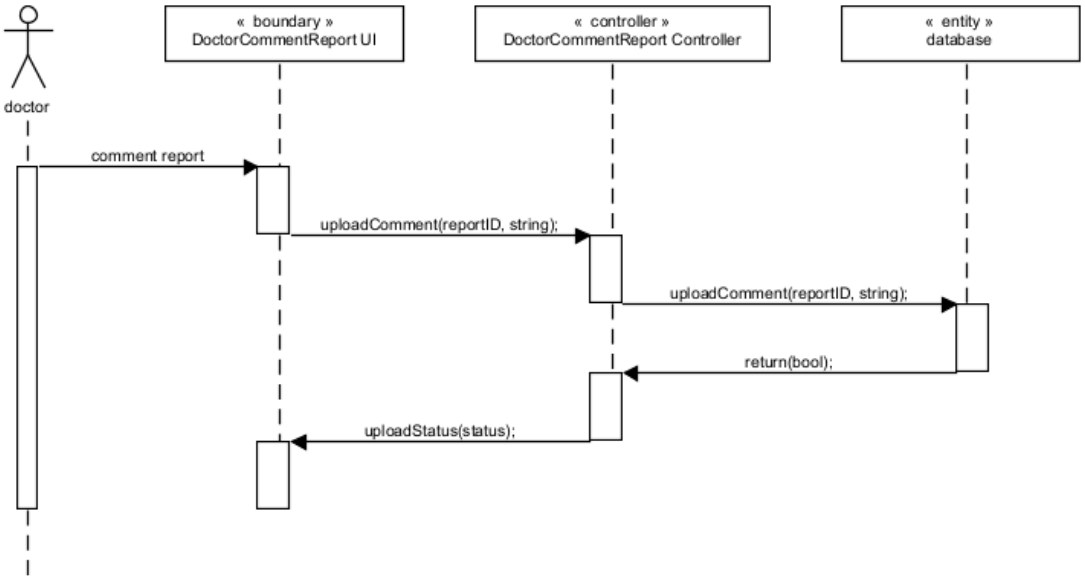


User story4: As a doctor, I want the ability to write comments or feedback on the generated report, so that I can provide additional insights or explanations to the patient.

BCE:

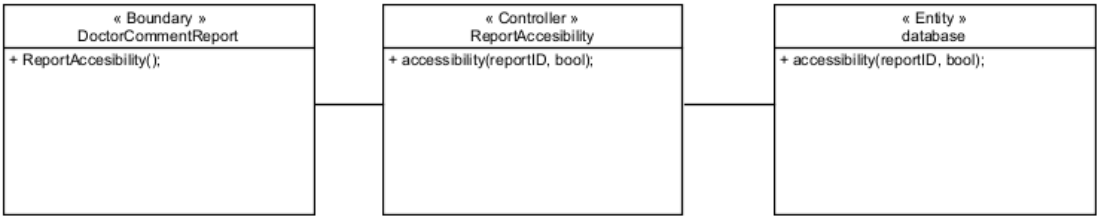


Sequential Diagram:

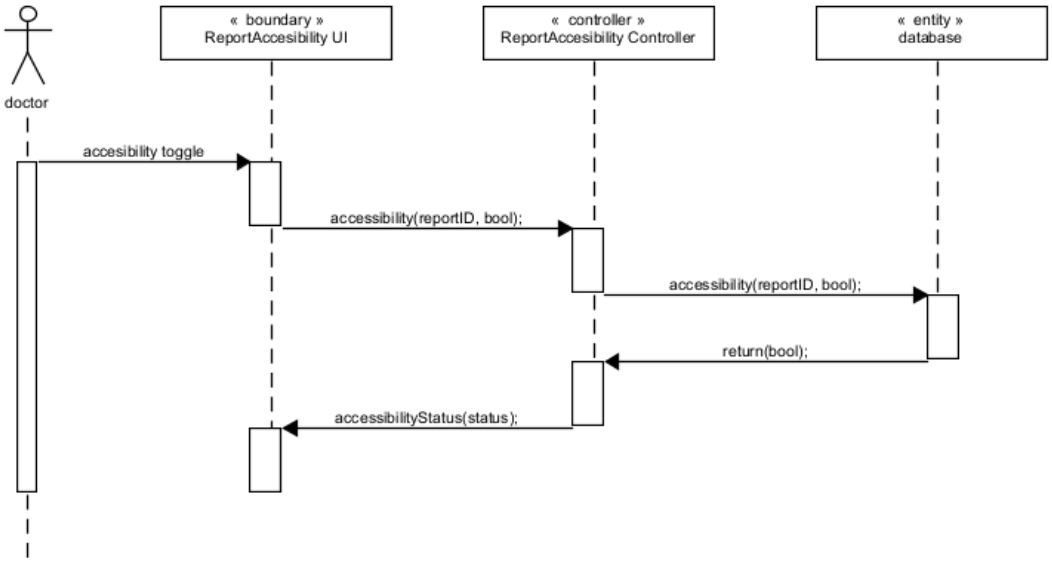


User story5: As a doctor, I want to make the report accessible to the patient, so that he/she can review the results, provide feedback and ask questions.

BCE:

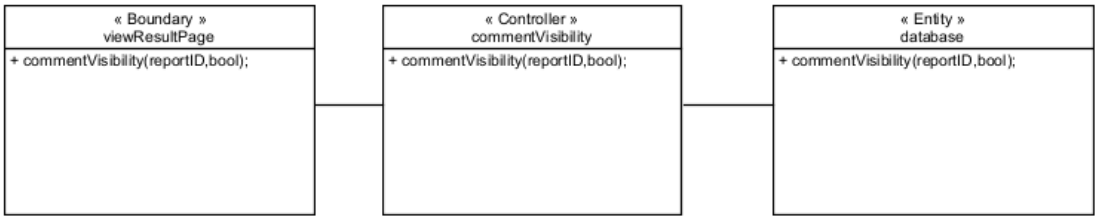


Sequential Diagram:

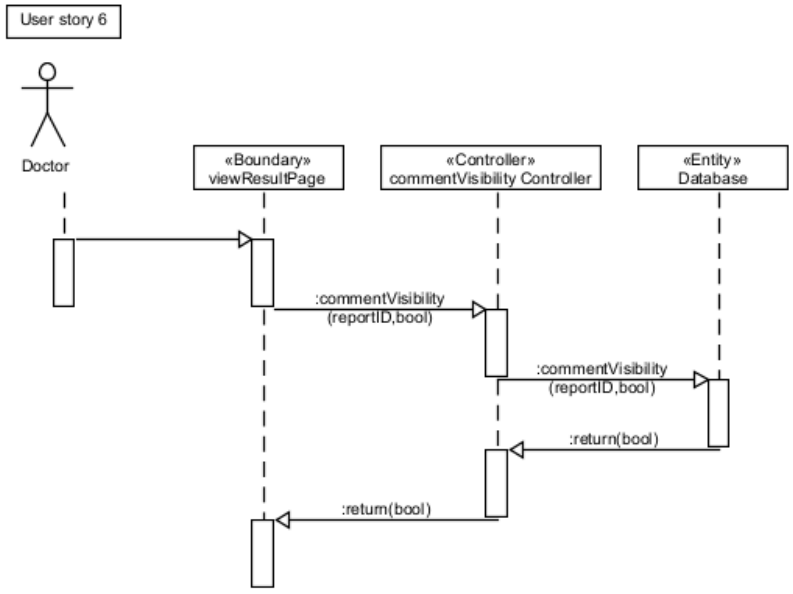


User story6: As a doctor, I want to make the report viewable to the patient, so that he/she can gain additional insights on his/her diagnosis.

BCE:

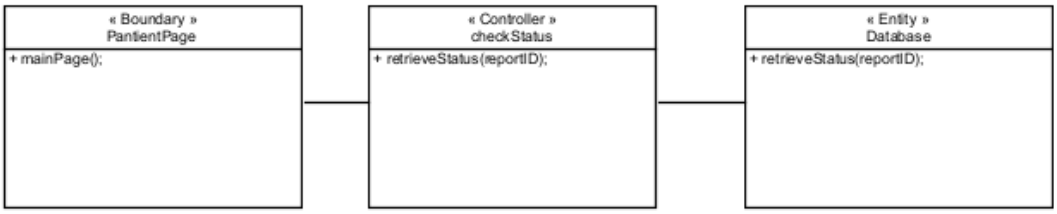


Sequential Diagram:

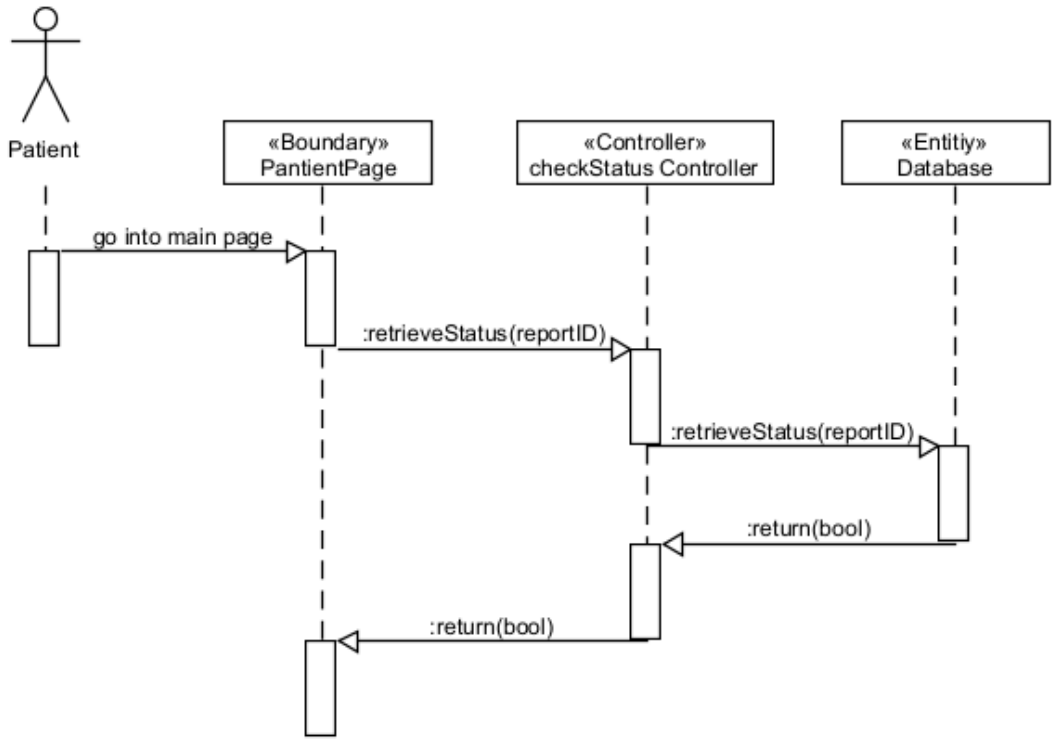


User story7: As a patient, I want to be able check the status of my diagnosis report so that I can track the progress.

BCE:

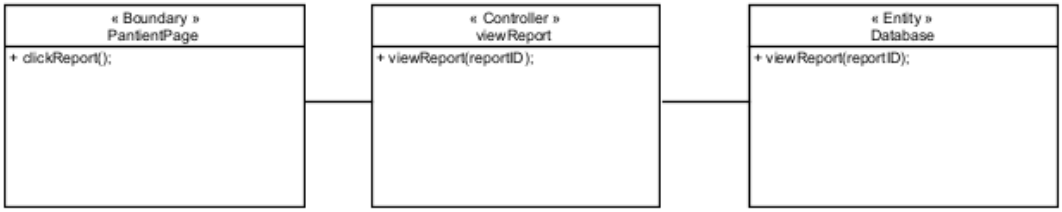


Sequential Diagram:

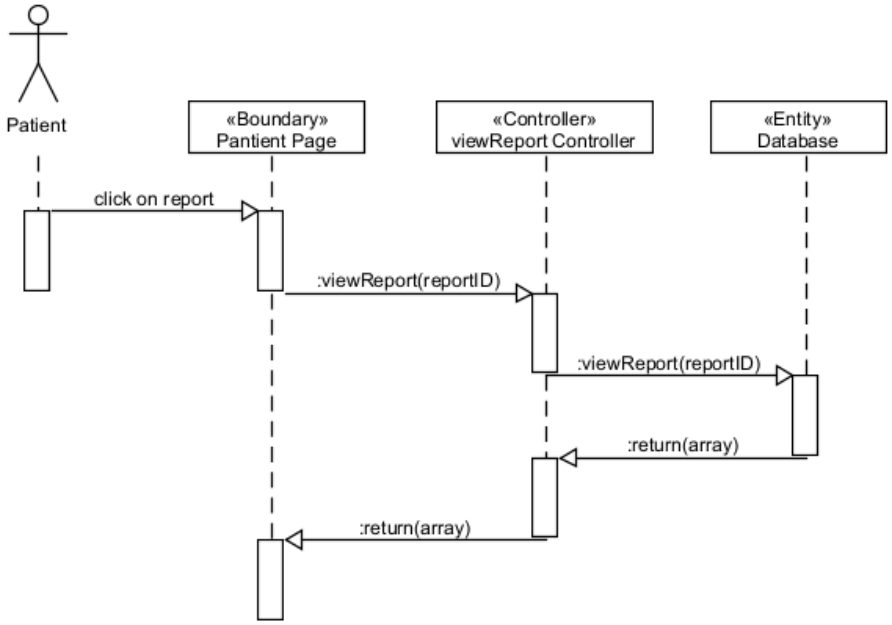


User story8: As a patient, I want to view the report released by the doctor's model, so that I can know if the X-ray indicates COVID-19 or not.

BCE:

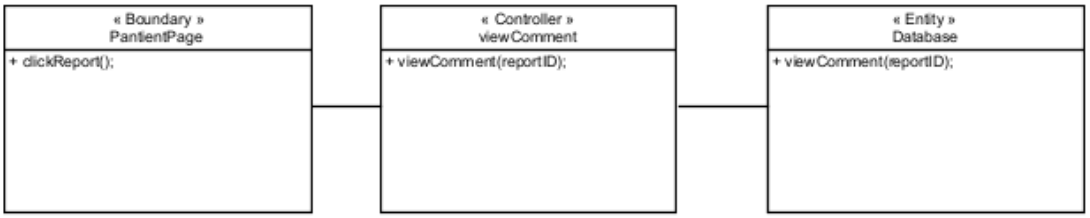


Sequential Diagram:

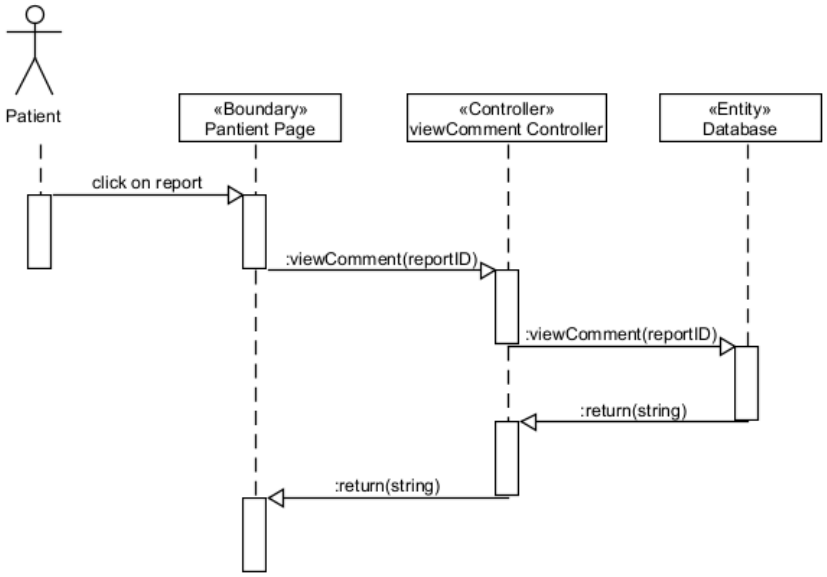


User story9: As a patient, I want to read the comments or feedback written by the doctor, so that I can better understand the diagnosis.

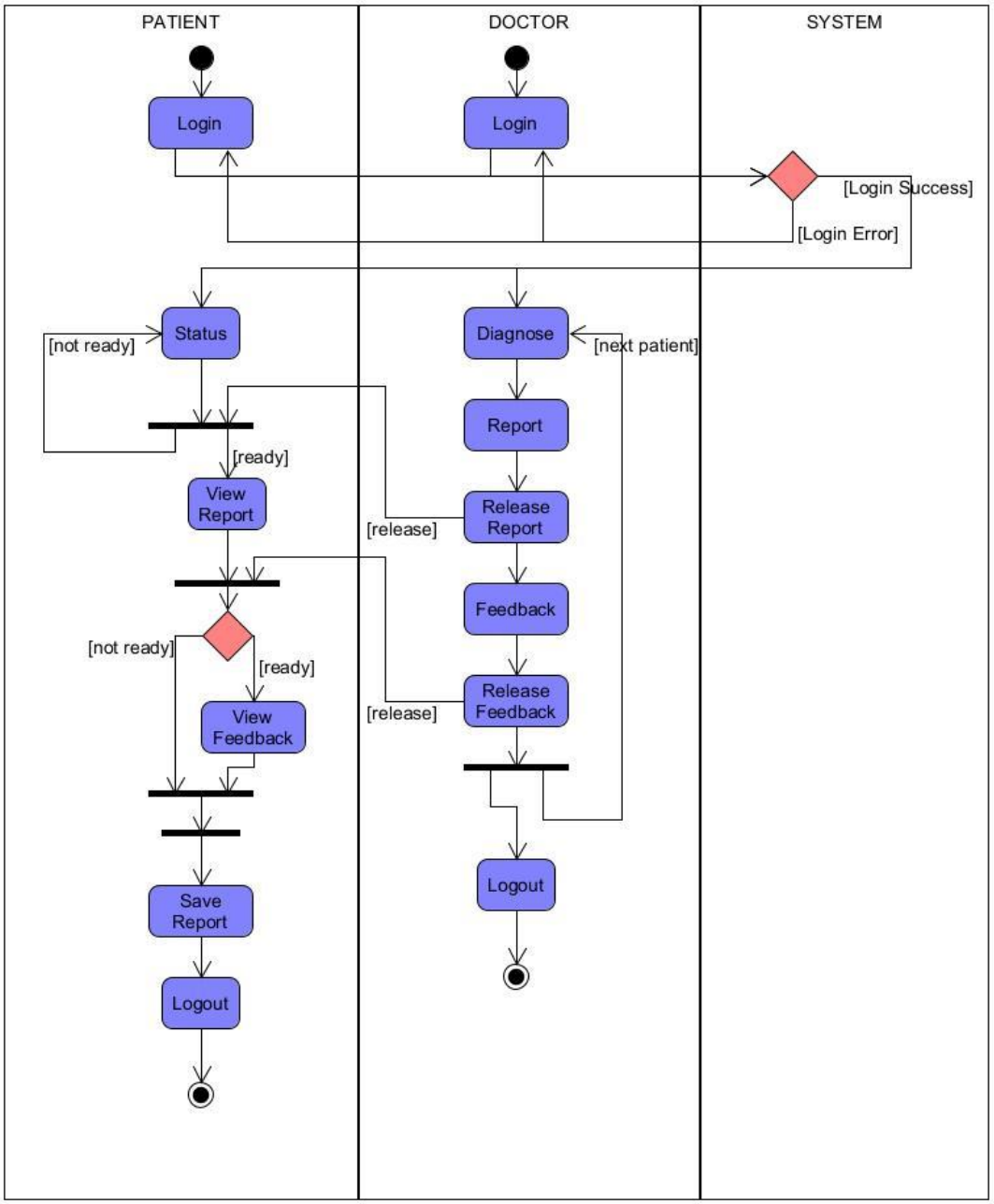
BCE:



Sequential Diagram:



4.5 Activity Diagram



4.6 Functional Requirements

4.6.1 Data Collection

- Collect reliable and relevant datasets for COVID-19 related data that contains X-rays or CT scans images.
- Ensure high-quality data is collected to train and test the machine learning models effectively.

4.6.2 Preprocessing

- Perform data preprocessing procedures to handle missing values, noise, or outliers in the collected data.
- Normalize the data to ensure that different data types and formats are compatible for training and testing the machine learning models.
- Perform feature extraction from the collected data to represent them in a suitable format for the machine learning model.

4.6.3 Model Development

- Build a machine learning model for COVID-19 diagnosis using lung detection and recognition techniques.
- Train and test the models on a public available dataset (e.g., Kaggle) and make sure the results are meeting the standard and requirements.

4.6.4 Model Evaluation

- Evaluate the accuracy, sensitivity, specificity, and F1 score of the model.
- Conduct relevant testing to ensure the models are meeting the standard and requirements.

4.6.5 User Interface

- Create an intuitive and user-friendly web interface for seamless interactions.
- Design a webpage where users can upload their images and view diagnosis results.

4.6.6 User Experience: Lung Detection

- Provide users with a lung detection feature to identify the uploaded images are showing the lungs correctly.
- Display the original image with detected lung regions highlighted.
- Allow users to adjust the detected regions manually if necessary.

4.6.7 User Experience: Lung Recognition

- Enable lung recognition functionality to predict COVID-19 diagnosis.

- Process the uploaded images through the machine learning model for lung recognition.
- Present the diagnosis results along with the scores achieved by the model and any other relevant information.

4.6.8 Privacy and Security:

- Implement secure user registration and authentication mechanisms to protect sensitive data.
- Ensure user data is encrypted and securely stored.

4.7 NON-FUNCTIONAL REQUIREMENTS

4.7.1 Performance

- The website should provide quick responses to user actions, such as image uploads and diagnosis requests, with minimal latency.
- The system should handle an increasing number of users and image processing requests without significant performance degradation.
- The website should be capable of processing multiple diagnosis requests concurrently, ensuring efficient use of computational resources.

4.7.2 Usability

- The website must be user-friendly and easy to navigate, allowing users to interact with the system effortlessly.
- Ensure the website is responsive across various devices and screen sizes for a consistent user experience.

4.7.3 Reliability

- Implement robust error handling mechanisms to gracefully handle unexpected scenarios and provide helpful error messages to users.
- Regularly backup user data and model parameters to prevent data loss in case of system failures.

4.7.4 Security

- Ensure all user data, including uploaded medical images, is encrypted during transmission and storage to protect user privacy.
- Token-based authentication can be implemented to ensure the users will be identified and direct to the respective dashboard.
- Access to the machine learning model is strictly not allowed to users.

4.7.5 Compliance

- Comply with relevant data protection regulations (e.g., GDPR) and obtain consent from users to collect and process their data.
- Ensure the machine learning model and website is only for educational purpose only and not liable for any outcome of the results.

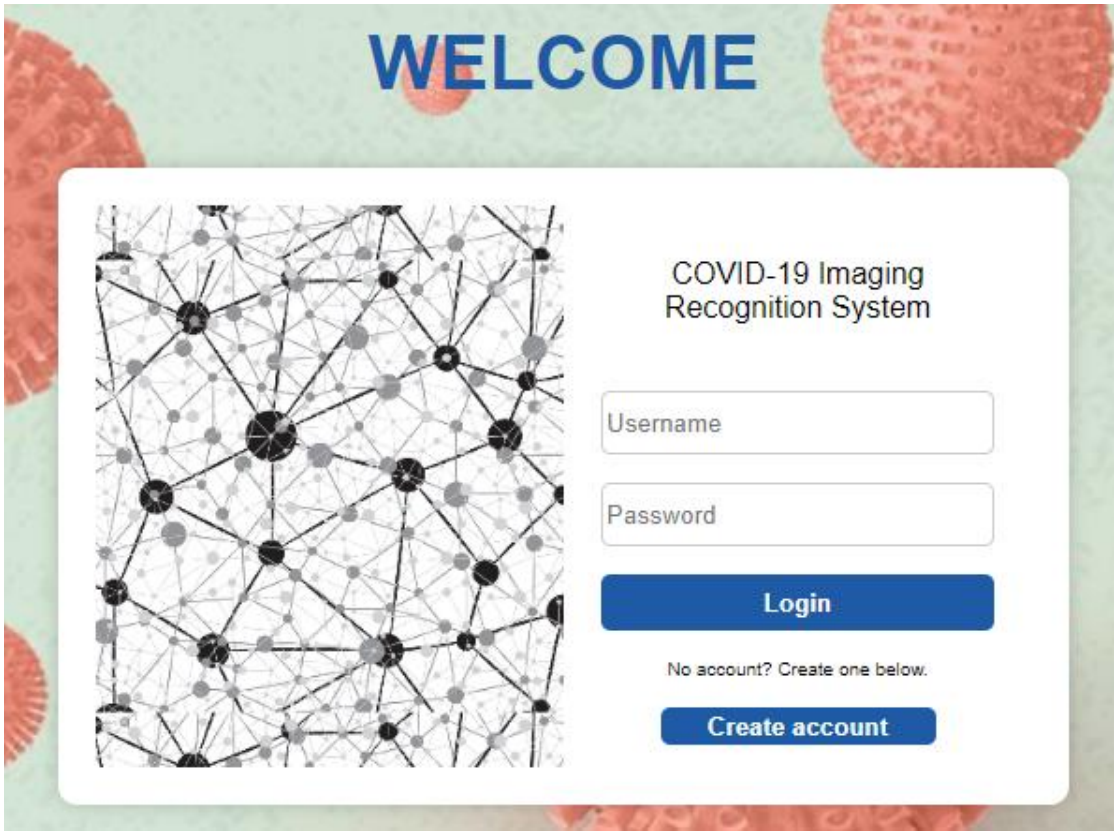
4.7.6 Maintainability

- Use version control systems (e.g., Github, Google Colab) to manage changes to the model and track updates.
- Maintain clear and comprehensive documentation for the code and machine learning model to help in future development and troubleshooting.

5. Software Design

5.1 User Interface of prototype

5.1.1 Login Page



Description:

The login page is a part of a website that serves as an entry point for users. It provides a form for users to enter their username and password. Upon submission, the page verifies the provided credentials. The user is redirected to either the Doctor Main Page or the Patient Main Page, depending on the credentials entered. There will be different privileges for doctors and patients. Otherwise, if the credentials are invalid, an alert message is displayed.

5.1.2 General Navigation bar

Description:

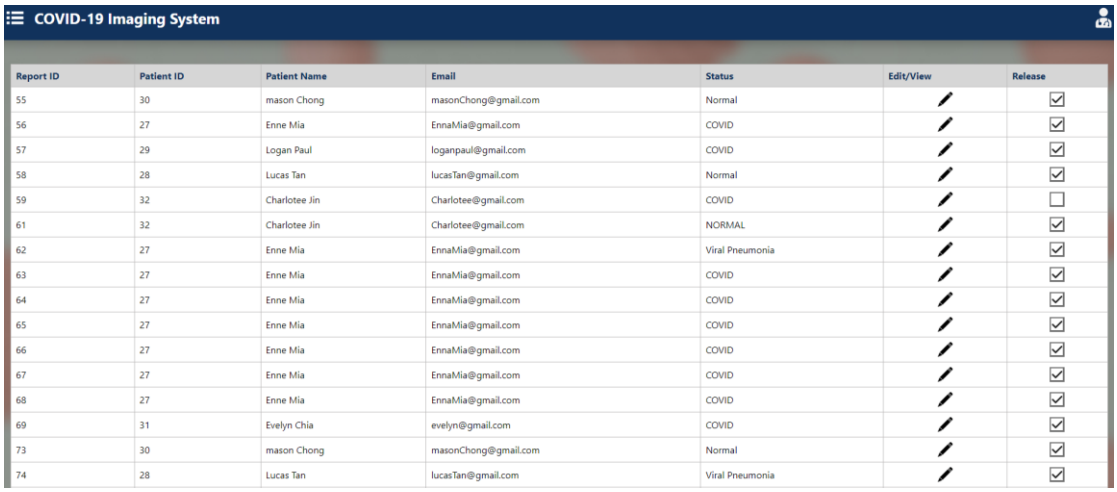
The navigation bar the has a menu tab for the user to access either “Home”, “Upload”, and “View result” where the user will be directed to upon selecting. The tab options will allow the user to select “logout” and “profile”. The main title “COVID-19 Imaging System” and the user’s name will be displayed as well.

5.1.3 Create Account

Description:

The system should provide a robust and user-friendly mechanism, empowering users to securely create and manage their accounts with confidence, ensuring the utmost protection of their personal information.

5.1.4 Doctor Main Page

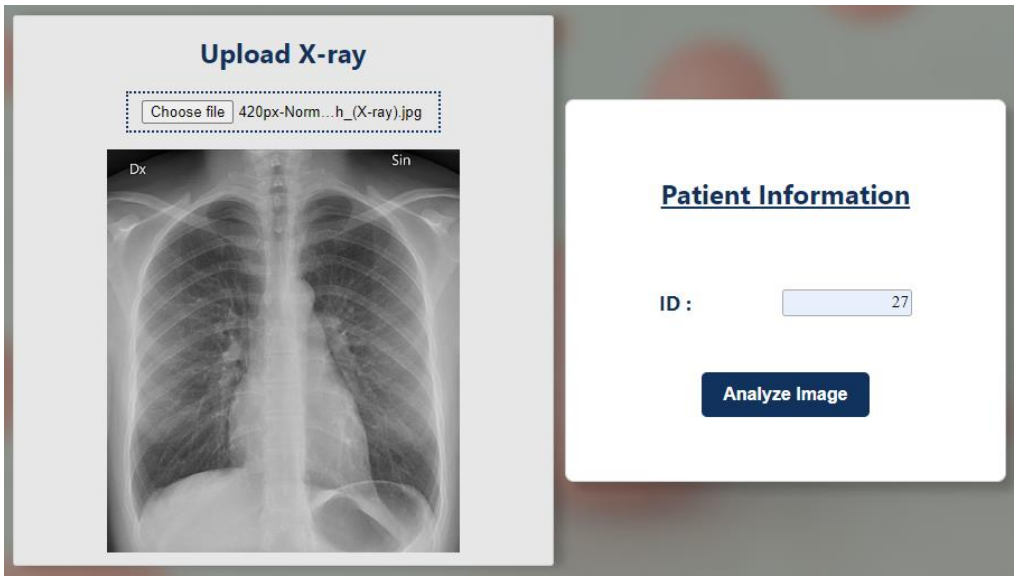


Report ID	Patient ID	Patient Name	Email	Status	Edit/View	Release
55	30	mason Chong	masonChong@gmail.com	Normal		☑
56	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
57	29	Logan Paul	loganpaul@gmail.com	COVID		☑
58	28	Lucas Tan	lucasTan@gmail.com	Normal		☑
59	32	Charlotee Jin	Charlotee@gmail.com	COVID		☐
61	32	Charlotee Jin	Charlotee@gmail.com	NORMAL		☑
62	27	Enne Mia	EnnaMia@gmail.com	Viral Pneumonia		☑
63	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
64	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
65	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
66	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
67	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
68	27	Enne Mia	EnnaMia@gmail.com	COVID		☑
69	31	Evelyn Chia	evelyn@gmail.com	COVID		☑
73	30	mason Chong	masonChong@gmail.com	Normal		☑
74	28	Lucas Tan	lucasTan@gmail.com	Viral Pneumonia		☑

Description:

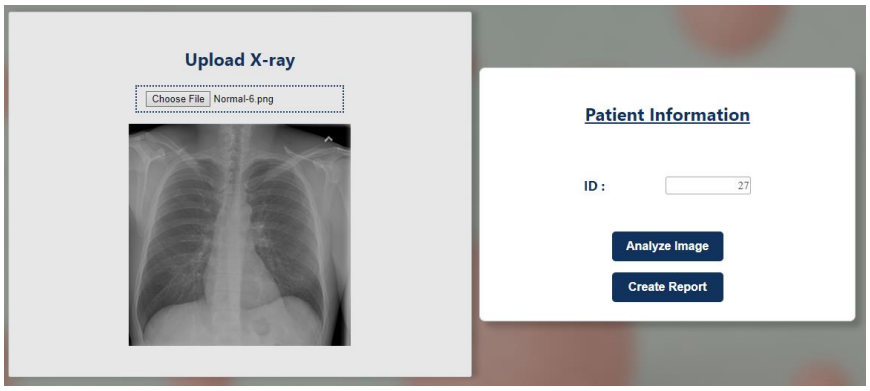
The table/dashboard below will display all existing patients records that the doctor has created or is attending to. The fields of a patient record will consist of “Report ID”, “Patient ID”, “Patient name”, “Email”, “Date created”, “Status”, “Edit/View”, “Release”.

5.1.5 Doctor Upload Page



Description:

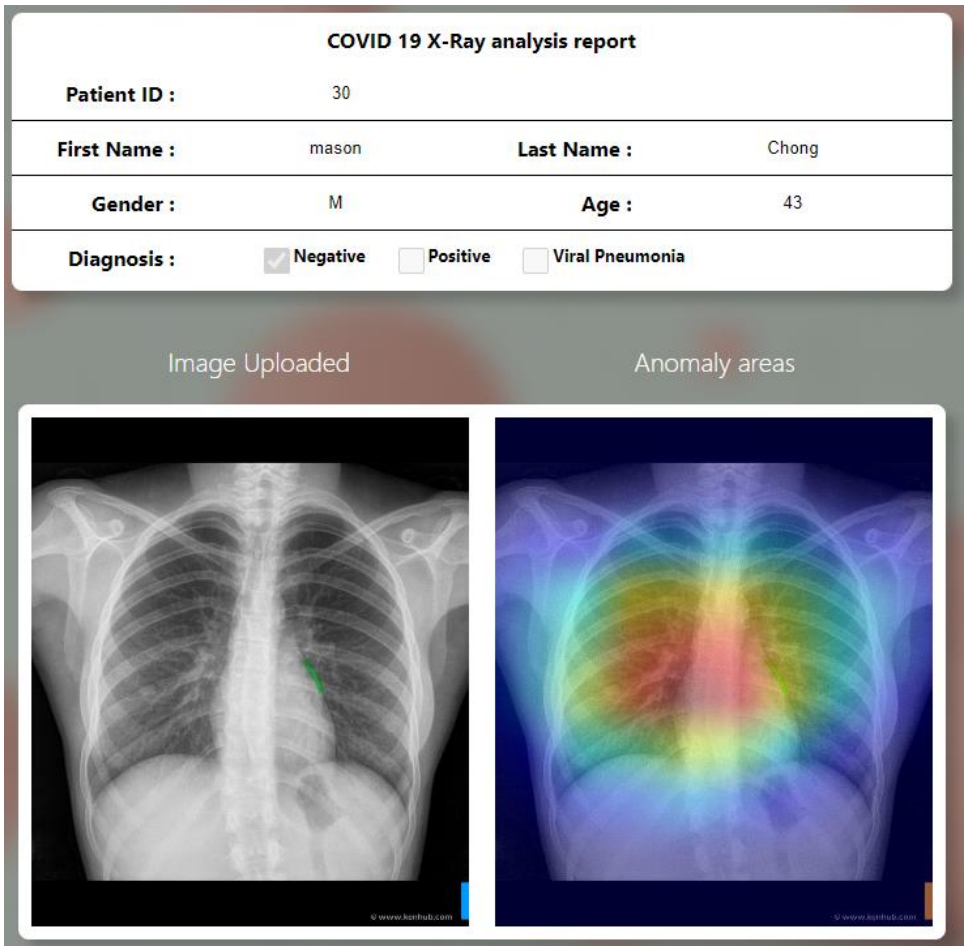
The page will allow doctors to upload an X-Ray of the patient and adding the Patient ID which is unique. The user can click on the “choose file” button where the user will be prompted to select an image file from the local device and once selected, the image will be displayed on the grey box as a preview.



Description:

After clicking on “Analyze Image”, the image will be processed once it is done all information will be collected and “Create Report” will show up upon successful detection and this will lead the user to DoctorViewResult page.

5.1.6 Doctor View Results



Description:

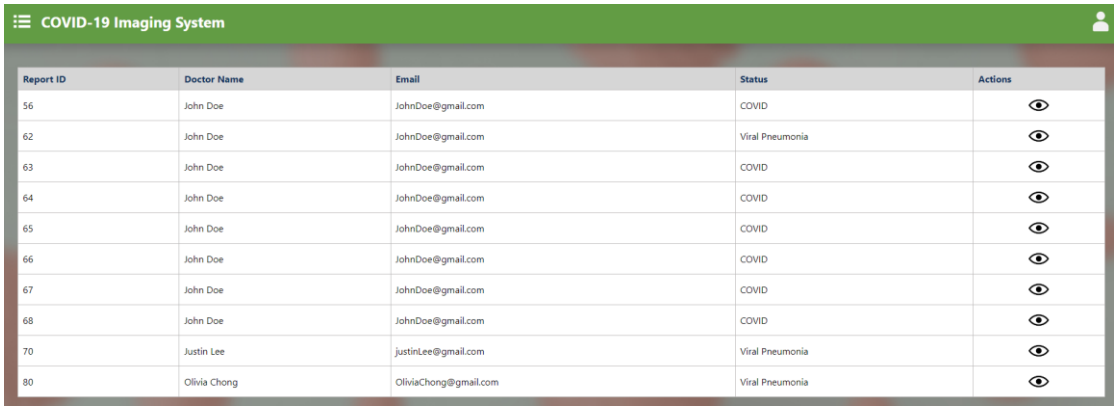
The page will display a COVID 19 X-Ray analysis report form, comprising of basic patients' fields like "Patient ID", "First Name", "Last Name", "Gender" and "Age". And the result of the covid diagnosis will be indicated by a tick under positive or negative. The photo uploaded previously and a heatmap will be shown next. The purpose of the heatmap is to make a comparison of the grey scale x-ray image than aim to help the doctor to have a better analysis of the x-ray.

5.1.7 Write comments for patients

Description:

Doctors are able to possess the capability to write supplementary comments pertaining to the patient, thereby enhancing the comprehensive nature of the patient's healthcare records.

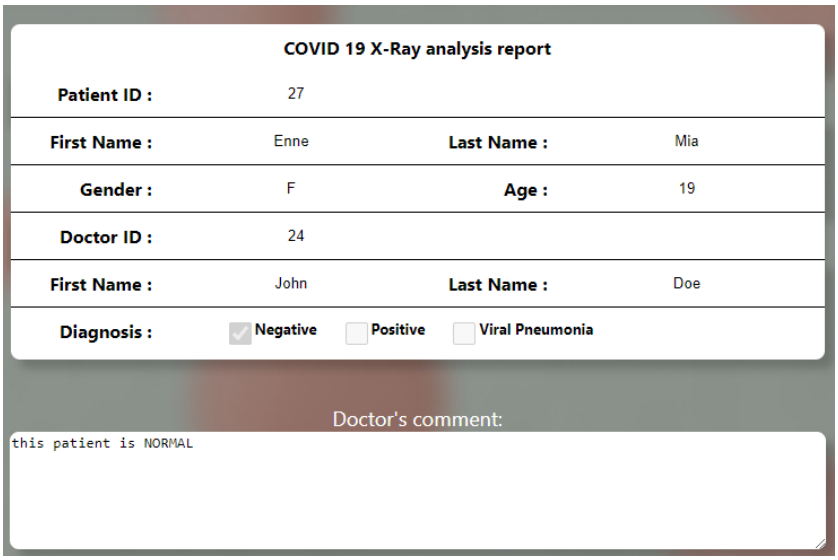
5.1.8 Patient Main Page



Description:

The table/dashboard below will display all existing patients records that the doctor has created or is attending to. The fields of a patient record will consist of “Report ID”, “Doctor ID”, “Email”, “Status”, “Actions”.

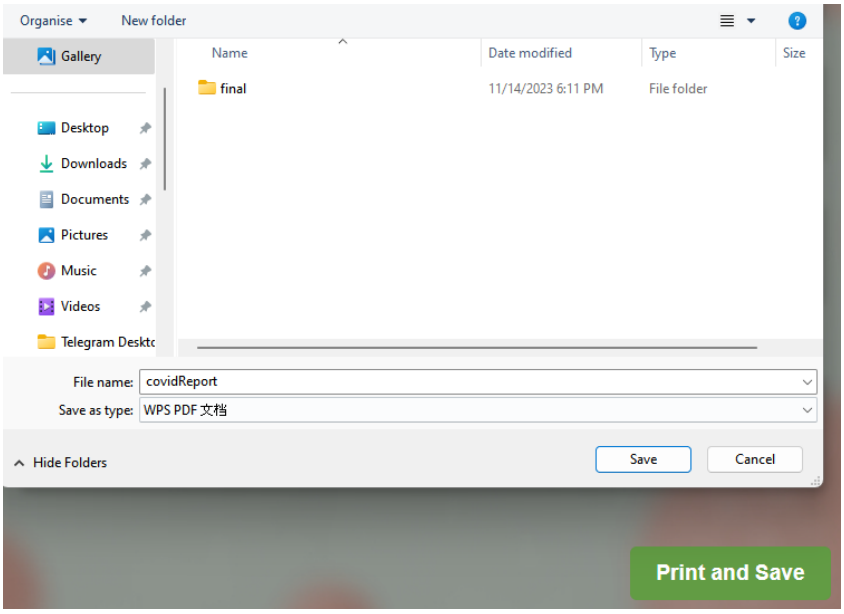
5.1.9 View analysis report and comment from doctor



Description:

Patients have the ability to access and review both the report and the insightful comments provided by the attending doctor, offering a comprehensive and transparent understanding of their healthcare information as shared by the medical practitioner.

5.2.1 View analysis report and comment from doctor



Description:

Patients are able save the analysis report.

5.3 Covid Detection Model

5.3.1 Data Collection

- COVID-19 CXR:
  - 994 images from padchest dataset
  - 183 images from a Germany medical school
  - 559 image from SIRM, Github, Kaggle & Tweeter
  - 400 images from another Github source
  - Total images: 2136
- Normal CXR:
  - 1141 images RSNA
  - 1211 images Kaggle
  - Total Images: 2352
- Viral Pneumonia CXR:
  - collected from the Chest X-Ray Images (pneumonia) database
  - Total images: 1345
- Total image samples: 5833

5.3.2 Exploratory Data Analysis

Creating a list of labels for different types of X-rays images in the dataset.

```
files = ['Normal', 'COVID','Viral Pneumonia']
```

For data processing, it iterates through files in different subdirectories specified in files and appends their paths and labels to the data list.

```
data = []
# Loop through each label in files and append file path and label to the data list
for id, level in enumerate(files):
    for file in os.listdir(os.path.join(path, level+'/'+'images')):
        data.append([level +'/' +'images'+ '/'+file, level])


data = pd.DataFrame(data, columns = ['image_file', 'corona_result'])

data['path'] = path + '/' + data['image_file']
data['corona_result'] = data['corona_result'].map({'Normal': 'Normal', 'COVID': 'Covid_positive', 'Viral Pneumonia':'Viral_Pneumonia'})

data.head()
```

	image_file	corona_result	path
0	Normal/images/Normal-2182.png	Normal	/content/drive/MyDrive/FYP/COVID-19_Radiograph...
1	Normal/images/Normal-229.png	Normal	/content/drive/MyDrive/FYP/COVID-19_Radiograph...
2	Normal/images/Normal-2311.png	Normal	/content/drive/MyDrive/FYP/COVID-19_Radiograph...
3	Normal/images/Normal-2248.png	Normal	/content/drive/MyDrive/FYP/COVID-19_Radiograph...
4	Normal/images/Normal-2150.png	Normal	/content/drive/MyDrive/FYP/COVID-19_Radiograph...

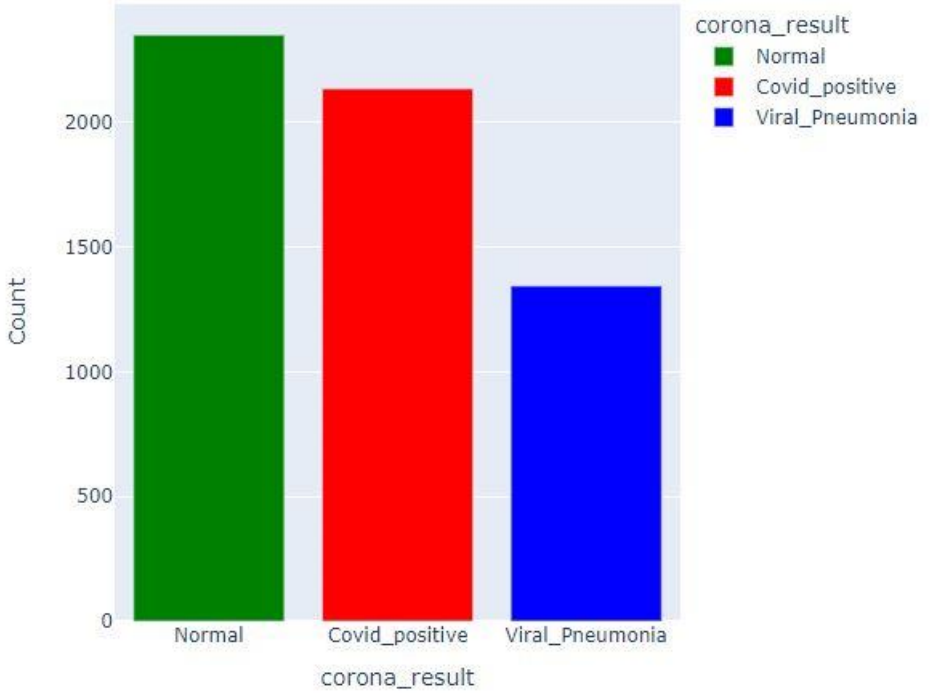
Even the dataset is carefully chosen, eds is still needed to check the data quality.

```
samples,features = data.shape
duplicated = data.duplicated().sum()
null_values = data.isnull().sum().sum()

print('Basic EDA')
print('Number of samples: %d'%(samples))
print('Number of duplicated values: %d'%(duplicated))
print('Number of Null samples: %d' % (null_values))
```

```
Basic EDA
Number of samples: 5833
Number of duplicated values: 0
Number of Null samples: 0
```

Generates a bar chart to visualize the distribution of different corona results. Normal and Covid_postive compose most of the dataset while Viral_Pneumonia has the least number. It will be interesting to see if the model will have greater difficulty in identifying Pneumonia or Covid samples.

```
# Creating a new DataFrame with counts for each corona_result
df = data['corona_result'].value_counts().reset_index()
df.columns = ['corona_result', 'Count']
df = df.sort_values(by=['Count'], ascending=False)

# Define a color mapping
color_map = {
    'Normal': 'green',
    'Covid_positive': 'red',
    'Viral_Pneumonia': 'blue'
}

fig = px.bar(df, x='corona_result', y='Count',
             color='corona_result',
             color_discrete_map=color_map,
             width=600,
             color_continuous_scale='BrBg')

fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.show()
```



5.3.4 Data Visualization

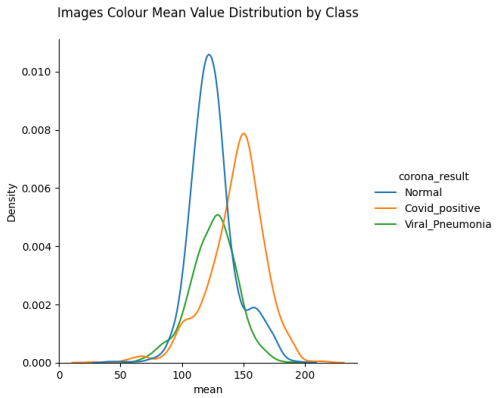
Based on the EDA:

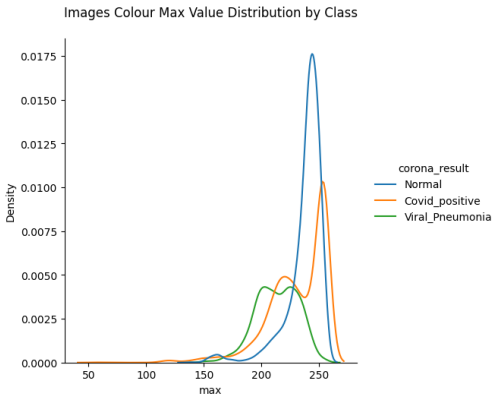
- The dataset contains a reasonable number of images.
- No data cleansing is required.
- We can investigate image patterns and relationships between the classes.

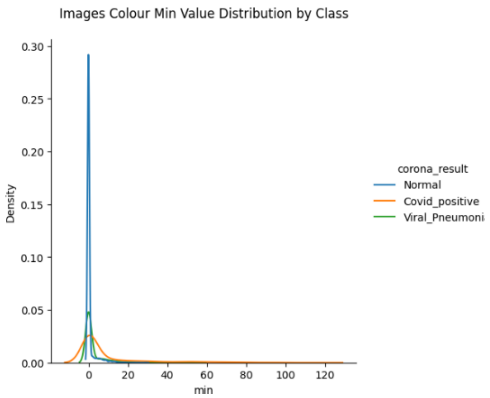
Calculates statistical values (mean, standard deviation, max, and min) for each image in the dataset. The purpose is to generate kernel density plots to visualize the distribution of mean, max, and min values of image. These plots help in understanding the characteristics of the images in different classes.

```
mean_val = []
std_dev_val = []
max_val = []
min_val = []

# Loop through each sample and calculate mean, standard deviation, max, and min values for each image
for i in range(0, no_of_samples):
    mean_val.append(data['image'][i].mean())
    std_dev_val.append(np.std(data['image'][i]))
    max_val.append(data['image'][i].max())
    min_val.append(data['image'][i].min())

imageEDA = data.loc[:,['image','corona_result','path']]
imageEDA['mean'] = mean_val
imageEDA['stedev'] = std_dev_val
imageEDA['max'] = max_val
imageEDA['min'] = min_val

imageEDA['subt_mean'] = imageEDA['mean'].mean() - imageEDA['mean']
```

```
# Create and display kernel density plots for mean, max, and min values by class
ax1 = sns.displot(data = imageEDA, x = 'mean', kind="kde", hue = 'corona_result');
plt.title('Images Colour Mean Value Distribution by Class\n', fontsize = 12);

ax2 = sns.displot(data = imageEDA, x = 'max', kind="kde", hue = 'corona_result');
plt.title('\nImages Colour Max Value Distribution by Class\n', fontsize = 12);

ax3 = sns.displot(data = imageEDA, x = 'min', kind="kde", hue = 'corona_result');
plt.title('\nImages Colour Min Value Distribution by Class\n', fontsize = 12);
```







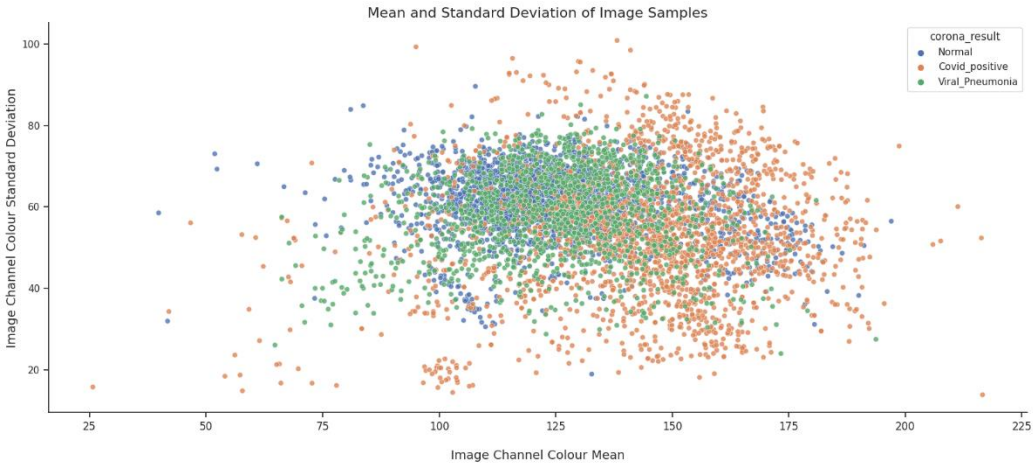
All the classes fall on the range of 100-150 for the mean value distribution.

Normal and Covid_postive have the max value close to 255 while Viral_Pneumnonia has lower range 200-225.

All the classes have 0 for min value as expected because they are x-rays images and the darkest color in the image is pure black. Viral_Pneumnonia has the highest density.

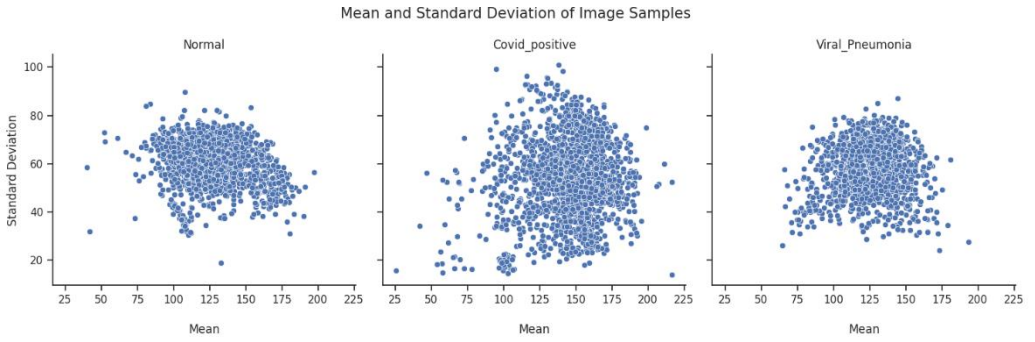
Create a scatter plot to visualize the relationship between the mean and standard deviation of color channels in the image samples.

```
plt.figure(figsize = (20, 8))
sns.set(style = "ticks", font_scale = 1)
ax = sns.scatterplot(data = imageEDA, x = "mean", y = imageEDA['stedev'], hue = 'corona_result', alpha = 0.8);
sns.despine(top = True, right = True, left = False, bottom = False)
plt.xticks(rotation = 0, fontsize = 12)
ax.set_xlabel('\nImage Channel Colour Mean', fontsize = 14)
ax.set_ylabel('Image Channel Colour Standard Deviation', fontsize = 14)
plt.title('Mean and Standard Deviation of Image Samples', fontsize = 16);
```



- Most images are gathered in the central region of the scatter plot as here is not much contrast between their pixel values
- Viral_Pneumnonia seems to be quite concentrated on the middle of the plot where Covid_postive is more spread apart to the right of the plot.
- All the classes are overlapping color mean range: 100 - 175

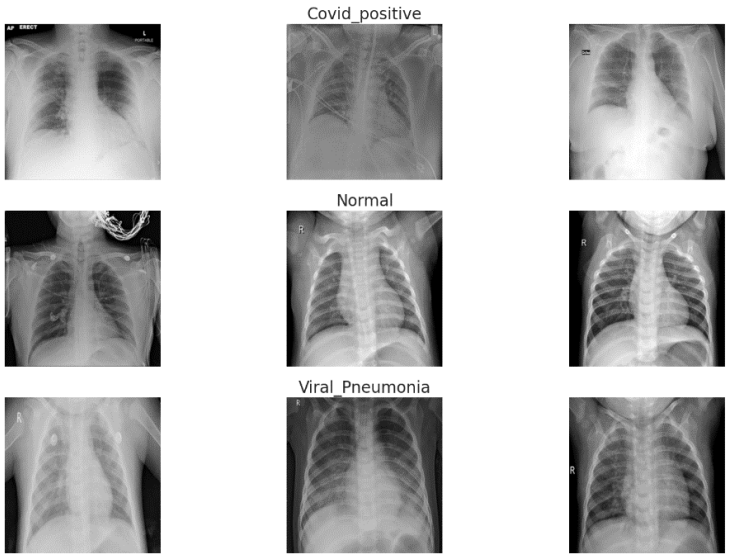
Create a scatter plot to visualize the relationship between the mean and standard deviation of the class individually.



- The Normal samples and Viral_Pneumnonia samples have a similar scatter. In other words, the spread of data points around the mean may be comparable for these two categories.
- Covid_postive samples are more spread apart. It may have a greater diversity in their color statistics compared to the other two categories.

Visualize the sample images of the dataset.

```
n_samples = 3

fig, m_axs = plt.subplots(3, n_samples, figsize = (6*n_samples, 3*4))

# Loop through each class and display a random sample of images
for n_axs, (type_name, type_rows) in zip(m_axs, data.sort_values(['corona_result']).groupby('corona_result')):
    n_axs[1].set_title(type_name, fontsize = 20)
    for c_ax, (_, c_row) in zip(n_axs, type_rows.sample(n_samples, random_state = 1234).iterrows()):
        picture = c_row['path']
        image = cv2.imread(picture)
        c_ax.imshow(image)
        c_ax.axis('off')
```

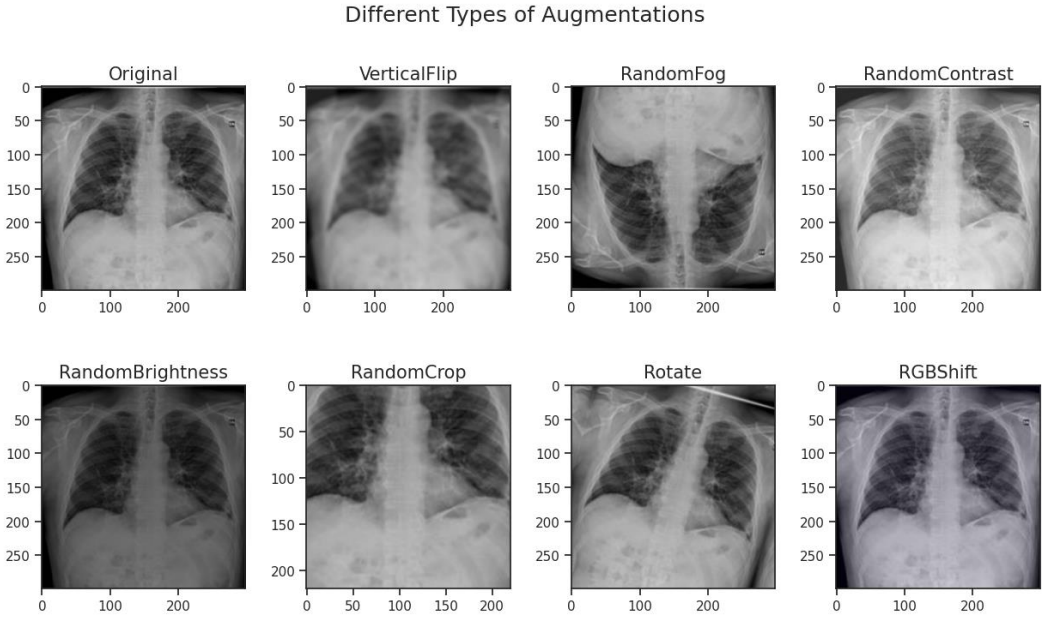


Grayscale images only have one channel representing the intensity of light, which ranges from black to white. In grayscale, each pixel is represented by a single 8-bit number, where 0 represents black and 255 represents white.

Albumentations Visualization

```
# Plot multiple images with titles
def plot_multiple_img(img_matrix_list, title_list, ncols, main_title = ""):

    fig, axis = plt.subplots(figsize = (15, 8), nrows = 2, ncols = ncols, squeeze = False)
    fig.suptitle(main_title, fontsize = 18)
    fig.subplots_adjust(wspace = 0.3)
    fig.subplots_adjust(hspace = 0.3)

    for i, (img, title) in enumerate(zip(img_matrix_list, title_list)):
        axis[i // ncols][i % ncols].imshow(img)
        axis[i // ncols][i % ncols].set_title(title, fontsize = 15)

    plt.show()
```

```
chosen_image = cv2.imread("/content/drive/MyDrive/COVID-19_Radiography_Dataset/COVID/images/COVID-101.png")

albumentation_list = [A.RandomFog(p = 1),A.VerticalFlip(p = 1), A.RandomBrightness(p = 1),A.RandomContrast(limit = 0.6, p = 1),
                      A.RandomCrop(p = 1,height = 220, width = 220),
                      A.Rotate(p = 1, limit = 90), A.RGBShift(p = 1)]

# Apply each augmentation and store the resulting images
img_matrix_list = []
bboxes_list = []
for aug_type in albumentation_list:
    img = aug_type(image = chosen_image)['image']
    img_matrix_list.append(img)

img_matrix_list.insert(0,chosen_image)

titles_list = ["Original", "VerticalFlip", "RandomFog", "RandomContrast", "RandomBrightness", "RandomCrop", "Rotate", "RGBShift"]

plot_multiple_img(img_matrix_list, titles_list, ncols = 4, main_title = "Different Types of Augmentations")
```



Visually preview how an image appears after undergoing various geometric transformations. This feature offers valuable insights into how each transformation alters the image's appearance.

5.3.5 Model Training(CNN)

Prepare the dataset for machine learning model. It reads, resizes, and normalizes the images while assigning categorical labels based on the 'corona_result' column.

```
all_data = []

# Storing images and their labels into a list for further Train Test split
{'Normal': 'Normal', 'COVID': 'Covid_positive', 'Viral Pneumonia':'Viral_Pneumonia'}
for i in range(len(data)):
    image = cv2.imread(data['path'][i])
    image = cv2.resize(image, (70, 70)) / 255.0
    label = 0
    if data['corona_result'][i] == "Normal":
        label = 0
    elif data['corona_result'][i] == "Covid_positive":
        label = 1
    elif data['corona_result'][i] == "Viral_Pneumonia":
        label = 2
    all_data.append([image, label])
```

The dataset is split into training, validation, and testing sets using the train_test_split function. The reason for splitting the data into these subsets is to enable model training on one portion, validation on another for fine-tuning, and testing on a separate, unseen set to evaluate its performance.

```
x = []
y = []

for image, label in all_data:
    x.append(image)
    y.append(label)

# Converting to Numpy Array
x = np.array(x)
y = np.array(y)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.1, random_state = 42)
```

The **CNN** consists of convolutional layers for feature extraction, max pooling layers for down-sampling, dropout layers for regularization, and dense layers for classification. The model is compiled with appropriate settings for training, including optimizer, loss function, and evaluation metric.

```
def create_model(n_classes, train_shape):
    cnn_model = models.Sequential()
    cnn_model.add(layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu', input_shape=train_shape))
    cnn_model.add(layers.MaxPooling2D((2, 2)))
    cnn_model.add(layers.Dropout(0.3))

    cnn_model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
    cnn_model.add(layers.MaxPooling2D((2, 2)))
    cnn_model.add(layers.Dropout(0.5))

    cnn_model.add(layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
    cnn_model.add(layers.Flatten())
    cnn_model.add(layers.Dense(units=16, activation='relu'))
    cnn_model.add(layers.Dropout(0.2))

    cnn_model.add(layers.Dense(units=3))

    cnn_model.compile(optimizer='adam',
                      loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                      metrics=['accuracy'])

    cnn_model.summary()
    return cnn_model
```

- Flatten converts multi-dimensional to one-dimensional array for dense layer
- Dense perform a linear operation on the input data followed by a activation function to produce the prediction

```
# Corresponds to images with dimensions 70x70 pixels and 3 color channels (RGB)
input_shape = (70, 70, 3)
n_classes = 3
conv_model = create_model(n_classes, input_shape)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 68, 68, 128)       3584

 max_pooling2d (MaxPooling2  (None, 34, 34, 128)       0
 D)

 dropout (Dropout)           (None, 34, 34, 128)       0

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        73792

 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 64)        0
 g2D)

 dropout_1 (Dropout)         (None, 16, 16, 64)        0

 conv2d_2 (Conv2D)           (None, 14, 14, 32)        18464

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 16)                100368

 dropout_2 (Dropout)         (None, 16)                0

 dense_1 (Dense)             (None, 3)                 51

=================================================================
Total params: 196259 (766.64 KB)
Trainable params: 196259 (766.64 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

The output displayed is the model summary, which provides detailed information about each layer in the network including the type, output shape, and the number of parameters. This summary helps in understanding the architecture of the CNN, which is structured with convolutional, max pooling, dropout, and dense layers for feature extraction and classification tasks.

```
early_stop = tf.keras.callbacks.EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 4)

#tf.random.set_seed(42)
history = conv_model.fit(x_train, y_train,
            epochs = 50,
            batch_size = 256,
            validation_data = (x_val, y_val),
            callbacks = [early_stop])
```

```
Epoch 1/50
17/17 [==============================] - 63s 4s/step - loss: 1.0822 - accuracy: 0.3939 - val_loss: 1.0326 - val_accuracy: 0.3619

                                                                    Min val loss
Epoch 35/50
17/17 [==============================] - 63s 4s/step - loss: 0.2588 - accuracy: 0.8945 - val_loss: 0.2382 - val_accuracy: 0.9122
Epoch 36/50
17/17 [==============================] - 62s 4s/step - loss: 0.2534 - accuracy: 0.8902 - val_loss: 0.3105 - val_accuracy: 0.8865
Epoch 37/50
17/17 [==============================] - 63s 4s/step - loss: 0.2458 - accuracy: 0.9021 - val_loss: 0.2837 - val_accuracy: 0.9079
Epoch 38/50
17/17 [==============================] - 63s 4s/step - loss: 0.2459 - accuracy: 0.8974 - val_loss: 0.3390 - val_accuracy: 0.8908
Epoch 39/50
17/17 [==============================] - 63s 4s/step - loss: 0.2556 - accuracy: 0.8914 - val_loss: 0.2889 - val_accuracy: 0.8951
Epoch 39: early stopping
```

Evaluates the model's performance on test dataset and training dataset. The evaluate function computes the model's performance metrics, including accuracy. This code allows for a quick assessment of how well the CNN model is performing on both the data it was trained on and data it has never seen before.

5.3.6 Model Evaluation

```
test_evaluation = conv_model.evaluate(x_test, y_test)
print(f"Test Accuracy using CNN: {test_evaluation[1] * 100:.2f}%")

train_evaluation = conv_model.evaluate(x_train, y_train)
print(f"Train Accuracy using CNN: {train_evaluation[1] * 100:.2f}%")
```

```
37/37 [==============================] - 4s 109ms/step - loss: 0.2644 - accuracy: 0.9135
Test Accuracy using CNN: 91.35%
132/132 [==============================] - 23s 174ms/step - loss: 0.1410 - accuracy: 0.9455
Train Accuracy using CNN: 94.55%
```

Next making predictions on three different sets of data: training, validation, and test sets, using the trained CNN model.

```
yp_train = conv_model.predict(x_train)
yp_train = np.argmax(yp_train, axis = 1)

yp_val = conv_model.predict(x_val)
yp_val = np.argmax(yp_val, axis = 1)

yp_test = conv_model.predict(x_test)
yp_test = np.argmax(yp_test, axis = 1)
```

```
132/132 [==============================] - 1s 4ms/step
15/15 [==============================] - 0s 12ms/step
37/37 [==============================] - 0s 4ms/step
```

Print a classification reports for the training, validation, and test data.

```
def confusion_matrix_train_test_val(name, y_train, yp_train, y_val, yp_val, y_test, yp_test):

    print("\n-----------------------------{}-----------------------------\n".format(name))

    print("Classification Report for Train Data\n")
    print(classification_report(y_train, yp_train))
    print("--------------------------------------------------------------------------------")

    print("\nClassification Report for Validation Data\n")
    print(classification_report(y_val, yp_val))
    print("--------------------------------------------------------------------------------")

    print("\nClassification Report for Test Data\n")
    print(classification_report(y_test, yp_test))
    print("--------------------------------------------------------------------------------")
```

```
confusion_matrix_train_test_val("CNN", y_train, yp_train, y_val, yp_val, y_test, yp_test)
```

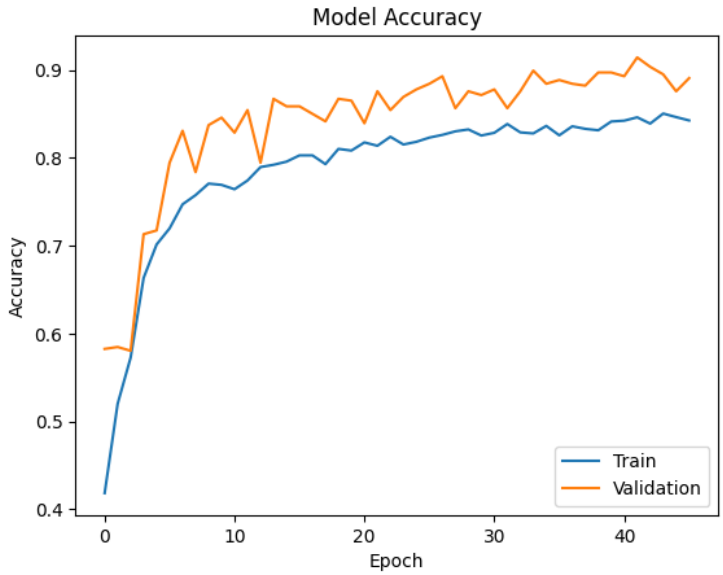
```
-----------------------------CNN-----------------------------

Classification Report for Train Data

              precision    recall  f1-score   support

           0       0.91      0.97      0.94      1703
           1       0.98      0.90      0.94      1551
           2       0.95      0.97      0.96       945

    accuracy                           0.94      4199
   macro avg       0.95      0.95      0.95      4199
weighted avg       0.95      0.94      0.94      4199

--------------------------------------------------------------------------------

Classification Report for Validation Data

              precision    recall  f1-score   support

           0       0.83      0.94      0.88       170
           1       0.94      0.82      0.87       180
           2       0.93      0.93      0.93       117

    accuracy                           0.89       467
   macro avg       0.90      0.90      0.90       467
weighted avg       0.90      0.89      0.89       467

--------------------------------------------------------------------------------

Classification Report for Test Data

              precision    recall  f1-score   support

           0       0.85      0.95      0.90       479
           1       0.94      0.81      0.87       405
           2       0.94      0.93      0.93       283

    accuracy                           0.90      1167
   macro avg       0.91      0.90      0.90      1167
weighted avg       0.90      0.90      0.90      1167

--------------------------------------------------------------------------------
```

- Training Data Report:
    - The model achieves an overall accuracy of 94%, indicating that it correctly classifies 94% of the instances in the training set. This is a good sign, suggesting that the model has learned well from the training data. The precision, recall, and F1-score metrics for each class (0, 1, and 2) are also high, indicating that the model performs well across all classes. This suggests that the model is able to effectively distinguish between different classes in the training set.

- Validation Data Report:
    - The model achieves an overall accuracy of 89% on the validation set. While this is slightly lower than the training accuracy, it's still a reasonably good performance. The precision, recall, and F1-score metrics for each class in the validation set are also relatively high, indicating that the model generalizes well to new, unseen examples.
- Test Data Report:
    - The model achieves an overall accuracy of 90% on the test set, which is another positive sign. The precision, recall, and F1-score metrics for each class in the test set are high, indicating that the model performs well on this independent dataset.

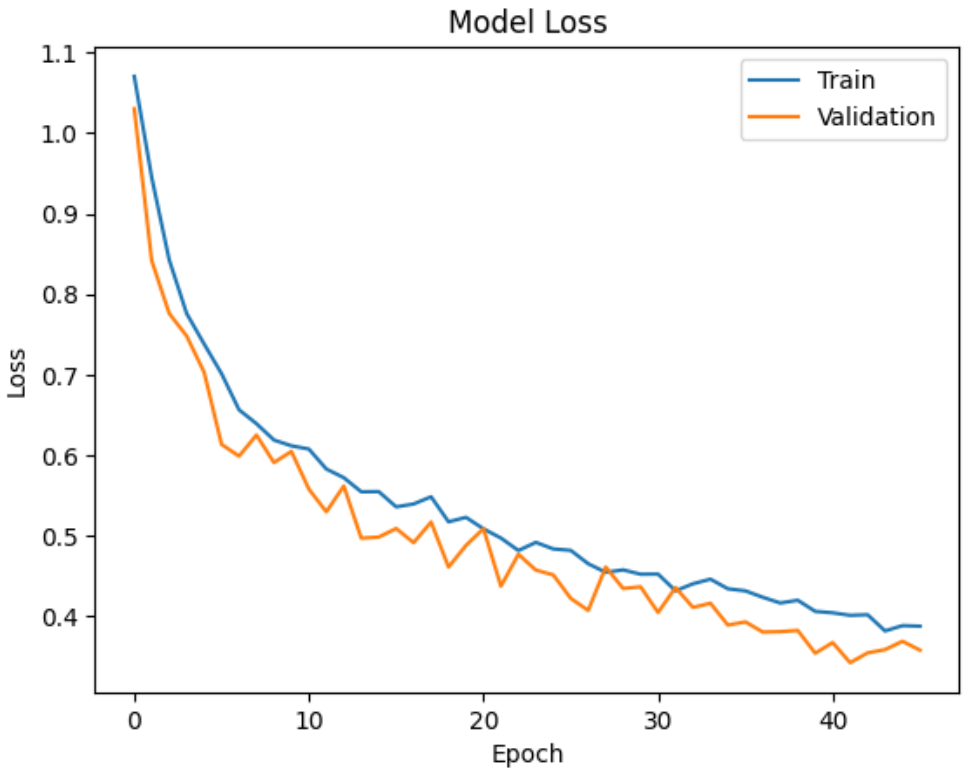
Visualize the training and validation accuracy of a machine learning model over multiple epochs

```
# Summarize History for Accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc = 'lower right')
plt.show()
```



Initially, training accuracy surges, then stabilizes near 0.85, suggesting efficient learning. Meanwhile, validation accuracy shows a more gradual increase, reaching around 0.9. This indicates the model generalizes well to unseen data, though further training may yield marginal gains.

```
# Summarize History for Loss

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc = 'upper right')
plt.show()
```



The graph illustrates the training and validation loss trends over epochs. Initially, training loss decreases rapidly, signifying effective learning. It then levels off at a low value, indicating the model's proficiency in fitting the training data. Additionally, validation loss mirrors the behavior of training loss, suggesting that the model generalizes well to unseen data.

5.3.7 Model Fine-Tuning

```
from sklearn.model_selection import ParameterGrid


# Define the hyperparameters and their possible values or ranges
param_grid = {
    'epochs': [20, 35, 50],
    'batch_size': [64, 128, 256]
}

# Generate all possible combinations of hyperparameters
param_combinations = list(ParameterGrid(param_grid))

best_val_loss = float('inf')
best_params = None

for params in param_combinations:
    # Train the model
    history = conv_model.fit(x_train, y_train,
                epochs=params['epochs'],
                batch_size=params['batch_size'],
                validation_data=(x_val, y_val),
                callbacks=[early_stop])

    # Evaluate on validation set
    val_loss = np.min(history.history['val_loss'])

    # Check if this is the best model so far
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        best_params = params

print("Best validation loss:", best_val_loss)
print("Best hyperparameters:", best_params)
```

```
Epoch 7: early stopping
Best validation loss: 0.2292184829711914
Best hyperparameters: {'batch_size': 64, 'epochs': 20, 'learning_rate': 0.001}
```

Train the model for multiple times with **different parameters** => check if there is potential pair of parameters which provides a higher accuracy

- **GridSearch:** Retrieves all combinations of parameters(“epochs” & “batch_size”)

Updates if current **“val_loss” < “best_val_loss”**

Updates “best_params” accordingly

Evaluates the **tuned** model's performance on test dataset and training dataset. The evaluate function computes the model's performance metrics, including accuracy. This code allows for a quick assessment of how well the CNN model is performing on both the data it was trained on and data it has never seen before

```
test_evaluation = conv_model.evaluate(x_test, y_test)
print(f"Test Accuracy using CNN: {test_evaluation[1] * 100:.2f}%")

train_evaluation = conv_model.evaluate(x_train, y_train)
print(f"Train Accuracy using CNN: {train_evaluation[1] * 100:.2f}%")
```

```
37/37 [==============================] - 7s 196ms/step - loss: 0.3353 - accuracy: 0.9340
Test Accuracy using CNN: 93.40%
132/132 [==============================] - 15s 112ms/step - loss: 0.0128 - accuracy: 0.9964
Train Accuracy using CNN: 99.64%
```

Next making predictions on three different sets of data: training, validation, and test sets, using the **tuned** CNN model.

```
yp_train = conv_model.predict(x_train)
yp_train = np.argmax(yp_train, axis = 1)

yp_val = conv_model.predict(x_val)
yp_val = np.argmax(yp_val, axis = 1)

yp_test = conv_model.predict(x_test)
yp_test = np.argmax(yp_test, axis = 1)
```

```
132/132 [==============================] - 15s 111ms/step
15/15 [==============================] - 2s 101ms/step
37/37 [==============================] - 4s 103ms/step
```

Print a classification reports for the training, validation, and test data.

```
def confusion_matrix_train_test_val(name, y_train, yp_train, y_val, yp_val, y_test, yp_test):

    print("\n----------------------------{}----------------------------\n".format(name))

    print("Classification Report for Train Data\n")
    print(classification_report(y_train, yp_train))
    print("-------------------------------------------------------------------------------")

    print("\nClassification Report for Validation Data\n")
    print(classification_report(y_val, yp_val))
    print("-------------------------------------------------------------------------------")

    print("\nClassification Report for Test Data\n")
    print(classification_report(y_test, yp_test))
    print("-------------------------------------------------------------------------------")
```

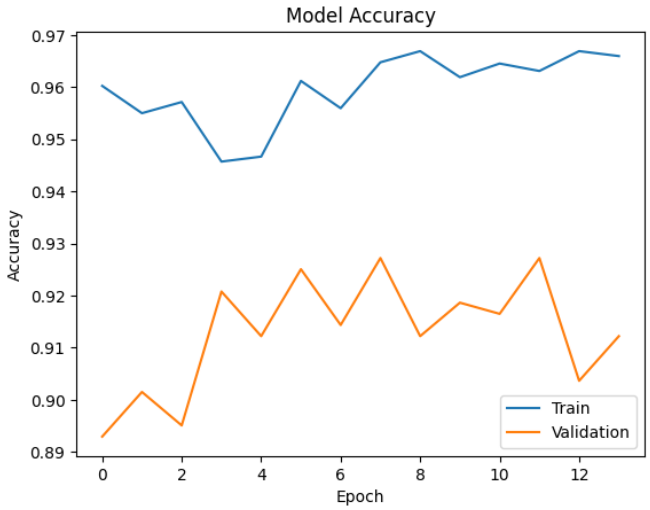
```
confusion_matrix_train_test_val("CNN", y_train, yp_train, y_val, yp_val, y_test, yp_test)
```

```
-----------------------------CNN----------------------------

Classification Report for Train Data

              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1703
           1       1.00      0.99      1.00      1551
           2       1.00      1.00      1.00       945

    accuracy                           1.00      4199
   macro avg       1.00      1.00      1.00      4199
weighted avg       1.00      1.00      1.00      4199

--------------------------------------------------------------------------------

Classification Report for Validation Data

              precision    recall  f1-score   support

           0       0.88      0.91      0.89       170
           1       0.93      0.91      0.92       180
           2       0.94      0.91      0.93       117

    accuracy                           0.91       467
   macro avg       0.92      0.91      0.91       467
weighted avg       0.91      0.91      0.91       467

--------------------------------------------------------------------------------

Classification Report for Test Data

              precision    recall  f1-score   support

           0       0.93      0.95      0.94       479
           1       0.94      0.92      0.93       405
           2       0.93      0.93      0.93       283

    accuracy                           0.93      1167
   macro avg       0.93      0.93      0.93      1167
weighted avg       0.93      0.93      0.93      1167

--------------------------------------------------------------------------------
```

These are the numbers after tuning the model, the accuracy of training data improved to 1.00, Validation data improved to 0.92 and testing data improved by 0.93. This means that the model's performance improves in terms of how well it learns from the training data, how well a model generalizes to new data and how it performs in the real-world scenario.

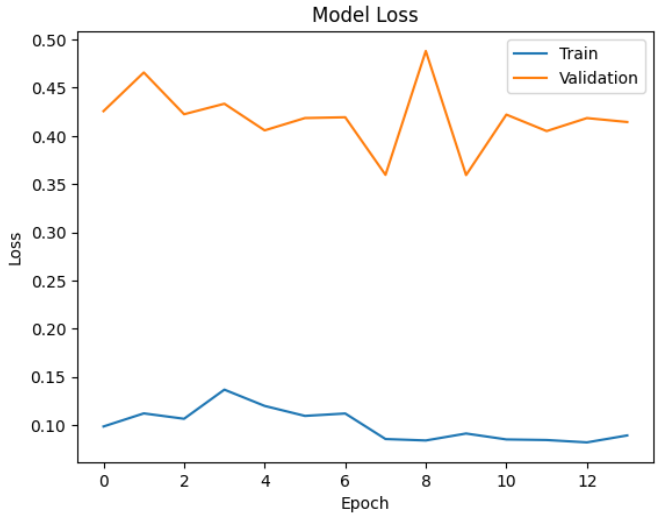
Visualize the training and validation accuracy of a machine learning model over multiple epochs.

```python
# Summarize History for Accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc = 'lower right')
plt.show()
```



After tuning the model, the accuracy started at an impressive 0.89 and 0.96, validation accuracy also increased gradually, which means our model generalizes new data better after each iteration.

```
# Summarize History for Loss

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc = 'upper right')
plt.show()
```



After tuning the model, both validation and model accuracy experienced an overall decrease in value, which indicates both efficient learning and how well the model generalizes to new and unseen data. The starting loss dropped for both validation and training, and both experienced slight overall decrease, which means the predictions of our model is closer than before tuning.

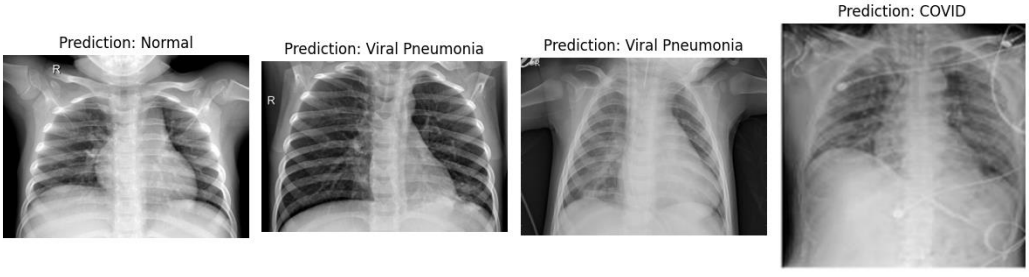
5.3.8 Model Testing

```
conv_model.save('/content/drive/MyDrive/FYP/COVID-19_Radiography_Dataset/conv_model_tuned.h5')
```

```
cnn_no_fine_tuning = load_model('/content/drive/MyDrive/COVID-19_Radiography_Dataset/conv_model.h5')

image_folder = "/content/drive/MyDrive/COVID-19_Radiography_Dataset/Test_Images/"

# Use glob to get a list of all image files in the folder
image_paths = glob.glob(image_folder + "*.jpg")

# Define class labels
class_labels = ['Normal', 'COVID', 'Viral Pneumonia']

# Set up a grid for the plots
num_cols = 4
num_rows = (len(image_paths) // num_cols) + 1
fig, axes = plt.subplots(num_rows, num_cols, figsize=(12, 12))
```

```
# Load and process each image
for idx, image_path in enumerate(image_paths):
    input_image = cv2.imread(image_path)

    # Calculate the position in the grid
    row = idx // num_cols
    col = idx % num_cols

    # Display the image
    axes[row, col].imshow(input_image)
    axes[row, col].axis('off')

    input_image = cv2.resize(input_image, (70, 70))
    input_image = np.expand_dims(input_image, axis=0)  # Add batch dimension
    input_image = input_image / 255.0

    predictions = cnn_no_fine_tuning.predict(input_image)

    # Convert the predictions to labels
    predicted_labels = np.argmax(predictions, axis=1)

    predicted_classes = [class_labels[label] for label in predicted_labels]

    # Set title with the predicted class and print the prediction
    axes[row, col].set_title(f"Prediction: {predicted_classes[0]}")

    # Print the prediction
    print(f"Prediction for image {image_path}: {predicted_classes[0]}")


# Remove any empty subplots
for i in range(len(image_paths), num_rows*num_cols):
    fig.delaxes(axes.flatten()[i])

plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 0s 35ms/step
Prediction for image /content/drive/MyDrive/COVID-19_Radiography_Dataset/Test_Images/NORMAL(1267).jpg: Normal
1/1 [==============================] - 0s 25ms/step
Prediction for image /content/drive/MyDrive/COVID-19_Radiography_Dataset/Test_Images/PNEUMONIA(4023).jpg: Viral Pneumonia
1/1 [==============================] - 0s 29ms/step
Prediction for image /content/drive/MyDrive/COVID-19_Radiography_Dataset/Test_Images/PNEUMONIA(4083).jpg: Viral Pneumonia
1/1 [==============================] - 0s 28ms/step
Prediction for image /content/drive/MyDrive/COVID-19_Radiography_Dataset/Test_Images/COVID-19 (1029).jpg: COVID
```



- The images used for testing are in the order of NORMAL, PNEUMONIA, PNEUMONIA, and COVID. As the result shown on the output with the corresponding images are Normal, Pneumonia, Pneumonia, and COVID.
- This shows that the CNN model is predicting the X-ray images correctly as shown in the output results.

5.4 Lung X-ray Image Classifier Model

5.4.1 Data Collection

Lung X-rays:

- 1000 images from:
    - Kaggle: COVID-19 Radiography Database
    - Radiology Masterclass
    - GitHub

Other X-rays:

- 777 images from:
    - Kaggle: Body Parts X-Ray Images in PNG
    - Kaggle: The UNIFESP X-Ray Body Part Classification Dataset

Total image samples: **1777**, belonging to **2 classes**.

5.4.2 Reading Data

- Importing necessary packages and libraries

```
import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd
import numpy as np
```

Matplotlib is a commonly used package for data visualization, chart plotting which helps to dig out the potential relationships between the attributes.
TensorFlow is an open-source machine learning framework developed by the Google Brain team. It is widely used for building and training machine learning models, particularly deep learning models. In this section, we are creating a Convolutional Neural Network (CNN) model using TensorFlow.
Pandas and Numpy are used for numerical computing and data manipulation.

- Mounting at Google Drive

```
from google.colab import files
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

To make use of the online IDE, Google Colab with its resources, it is necessary to store the data in Google Drive.

- Defining the path of the images in Google Drive. Getting the class labels

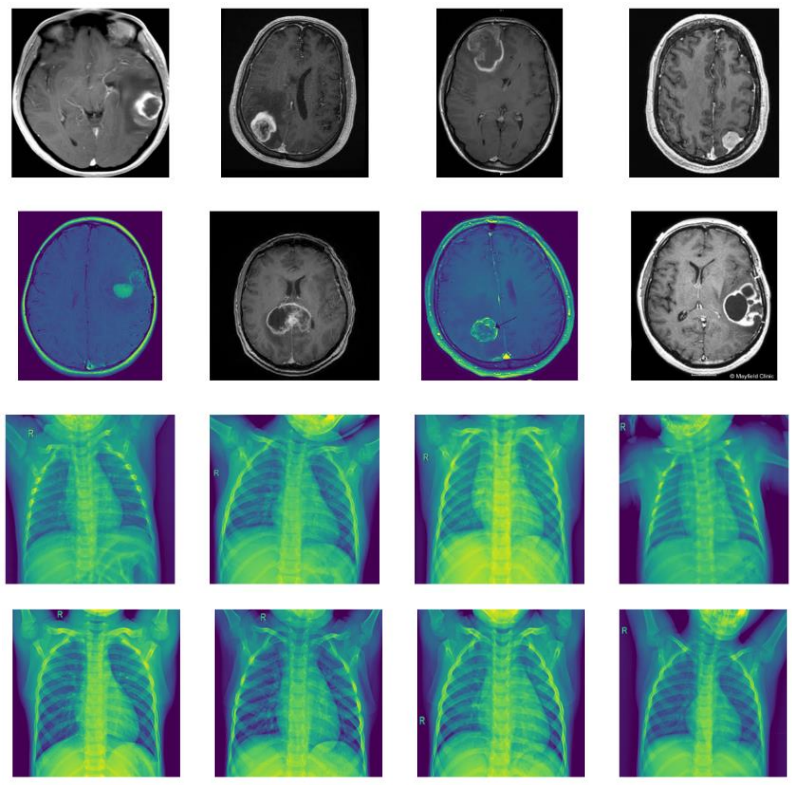
```
path = '/content/drive/MyDrive/FYP/xray_recog/xray_dataset'
classes = os.listdir(path)
classes
```

```
['other', 'lung']
```

Storing the images in Google Drive under the path, and make sure that lung x-rays and other x-rays are in respective directories.

5.4.3 Data Visualization

```
fig = plt.gcf()
fig.set_size_inches(16, 16)


other_dir = os.path.join('/content/drive/MyDrive/FYP/xray_recog/xray_dataset/other')
lung_dir = os.path.join('/content/drive/MyDrive/FYP/xray_recog/xray_dataset/lung')
other_names = os.listdir(other_dir)
lung_names = os.listdir(lung_dir)

pic_index = 210

other_images = [os.path.join(other_dir, fname) for fname in other_names[pic_index-8:pic_index]]
lung_images = [os.path.join(lung_dir, fname) for fname in lung_names[pic_index-8:pic_index]]

for i, img_path in enumerate(other_images + lung_images):
        sp = plt.subplot(4, 4, i+1)
        sp.axis('Off')

        img = mpimg.imread(img_path)
        plt.imshow(img)

plt.show()
```



It first retrieves the current figure using plt.gcf() and set its size to 16x16 inches. This ensures that the images displayed are large enough for detailed examination.

The script then loops through the selected images, combining both 'other' and 'lung' images. For each image, it creates a subplot in a 4x4 grid without axes (using sp.axis('Off')) and reads the image from the file path (mpimg.imread(img_path)). It then displays the image in the designated subplot using plt.imshow(img).

Based on the output, it is obvious that the images are quite different in sizes, colour channels, and angles. These problems are to be handles in the preprocessing.

5.4.3 Model Training Preparations 1

```
# Define the rescaling layer
rescale = layers.Rescaling(1./255)

base_dir = '/content/drive/MyDrive/FYP/xray_recog/xray_dataset'

# Create datasets
train_datagen = image_dataset_from_directory(
        base_dir,
        image_size=(200, 200),
        subset='training',
        seed=1,
        validation_split=0.1,
        batch_size=32
)

# Apply normalization using the rescaling layer
train_datagen = train_datagen.map(lambda x, y: (rescale(x), y))
```

This part of the code creates a rescaling layer that normalizes image pixel values. The 1./255 factor is used to scale the RGB color values (ranging from 0 to 255) to a range of 0 to 1. This normalization is a common practice in image processing as it helps in speeding up the training and leads to better model performance.

The 'image_dataset_from_directory' is used to automatically load images for the training dataset. Images are resized to 200x200 pixels. The function also supports splitting the dataset into training and validation subsets. In this case, 10% of the data (indicated by validation_split=0.1) is reserved for validation. The seed ensures consistency in the splitting process.

In the end we apply the earlier defined rescaling layer to the training dataset. The map function is used to apply the normalization to each image in the dataset (x), while keeping its label (y) unchanged.

5.4.3 Model Training Preparations 2

```
# Create the validation dataset with normalization
test_datagen = image_dataset_from_directory(
        base_dir,
        image_size=(200, 200),
        subset='validation',
        seed=1,
        validation_split=0.1,
        batch_size=32
)

# Apply normalization using the rescaling layer
test_datagen = test_datagen.map(lambda x, y: (rescale(x), y))
```

This code block performs a similar operation as the previous, but for creating the validation dataset. The only difference is the subset='validation' parameter, which ensures that the data split for validation is used.

5.4.3 Model Training(CNN)

```
input_shape = (200, 200, 3)

# Create a Sequential model
binary_classification_model = models.Sequential()

# Convolutional layers
binary_classification_model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
binary_classification_model.add(layers.MaxPooling2D((2, 2)))
binary_classification_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
binary_classification_model.add(layers.MaxPooling2D((2, 2)))
binary_classification_model.add(layers.Conv2D(128, (3, 3), activation='relu'))
binary_classification_model.add(layers.MaxPooling2D((2, 2)))

# Flatten before fully connected layers
binary_classification_model.add(layers.Flatten())

# Fully connected layers
binary_classification_model.add(layers.Dense(128, activation='relu'))
binary_classification_model.add(layers.Dropout(0.5))    # Dropout for regularization
binary_classification_model.add(layers.Dense(1, activation='sigmoid'))    # Sigmoid for binary classification

# Compile the model
binary_classification_model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

First of all, we set our model expecting input images of size 200x200 pixels with 3 color channels (RGB). Then, a Sequential model is initiated. This type of model allows for the creation of layers in a step-by-step fashion.

The model includes three convolutional layers (Conv2D) with increasing numbers of filters (32, 64, 128). Each convolutional layer uses a 3x3 kernel and is followed by a max-pooling layer (MaxPooling2D) with a 2x2 pool size, reducing the spatial dimensions of the output. The 'relu' activation function is used for introducing non-linearity.

The Flatten layer transforms the 2D matrix data to a 1D vector before feeding it into the fully connected layers.

After flattening, the network includes a fully connected (Dense) layer with 128 units and 'relu' activation. A dropout layer with a rate of 0.5 is included for regularization to prevent overfitting. The final layer is another Dense layer with a single unit and a 'sigmoid' activation function, suitable for binary classification.

The last essential step is to compile the model, with the Adam optimizer and 'binary_crossentropy' as the loss function, which is appropriate for binary classification tasks. The performance metric is set to 'accuracy'.

```
# Display the model summary
binary_classification_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 198, 198, 32)      896

max_pooling2d_3 (MaxPoolin   (None, 99, 99, 32)        0
g2D)

conv2d_4 (Conv2D)            (None, 97, 97, 64)        18496

max_pooling2d_4 (MaxPoolin   (None, 48, 48, 64)        0
g2D)

conv2d_5 (Conv2D)            (None, 46, 46, 128)       73856

max_pooling2d_5 (MaxPoolin   (None, 23, 23, 128)       0
g2D)

flatten_1 (Flatten)          (None, 67712)             0

dense_2 (Dense)              (None, 128)               8667264

dropout_1 (Dropout)          (None, 128)               0

dense_3 (Dense)              (None, 1)                 129

=================================================================
Total params: 8760641 (33.42 MB)
Trainable params: 8760641 (33.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

5.4.3 Model Training Fitting

```
# Fit the model to the training dataset
history = binary_classification_model.fit(
        train_datagen,
        validation_data=test_datagen,
        epochs=15
)
```

```
Epoch 1/15
50/50 [==============================] - 144s 3s/step - loss: 0.2342 - accuracy: 0.9144 - val_loss: 0.1162 - val_accuracy: 0.9605

Epoch 13/15
50/50 [==============================] - 147s 3s/step - loss: 0.0014 - accuracy: 0.9994 - val_loss: 0.0706 - val_accuracy: 0.9831
Epoch 14/15
50/50 [==============================] - 142s 3s/step - loss: 0.8882 - accuracy: 0.9187 - val_loss: 0.1341 - val_accuracy: 0.9718
Epoch 15/15
50/50 [==============================] - 149s 3s/step - loss: 0.0213 - accuracy: 0.9931 - val_loss: 0.1423 - val_accuracy: 0.9774
```

This line of code is where the actual training of the model occurs. The fit method is used to train the model with the specified training and validation datasets.

The model is trained using the 'train_datagen' dataset, which contains preprocessed and normalized training images, along with their corresponding labels.

The 'validation_data' argument is set to 'test_datagen', which is the dataset reserved for validating the model's performance after each training epoch. This dataset is not used for training but to evaluate how well the model is learning and generalizing to new data.

The training process is set to run for 15 epochs. An epoch is one complete pass through the entire training dataset. This means the model will go through the training dataset 15 times, each time updating its weights and biases to minimize the loss and improve accuracy.

5.4.3 Model Predicting

```
from keras.preprocessing import image

test_image = image.load_img('/content/drive/MyDrive/FYP/xray_recog/brain-1.jpg',target_size=(200,200))
#For show image
plt.imshow(test_image)
test_image = image.img_to_array(test_image)
test_image = test_image / 255.0
test_image = np.expand_dims(test_image,axis=0)


# Result array
result = binary_classification_model.predict(test_image)

print(result)
if result[0][0] >= 0.5:
        print("Predicted Class: Other")
else:
        print("Predicted Class: Lung")
```

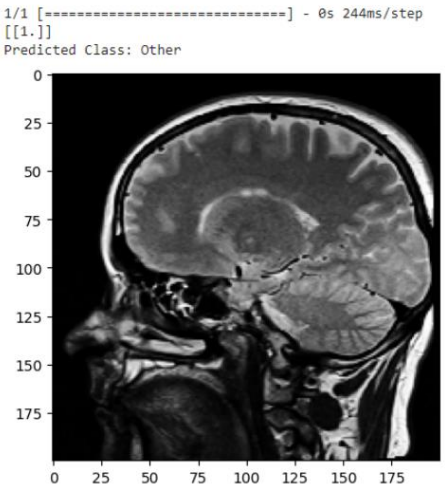
The ‘load_img’ function from Keras' image module is used to load the test image, resizing it to 200x200 pixels to match the input shape of the model.

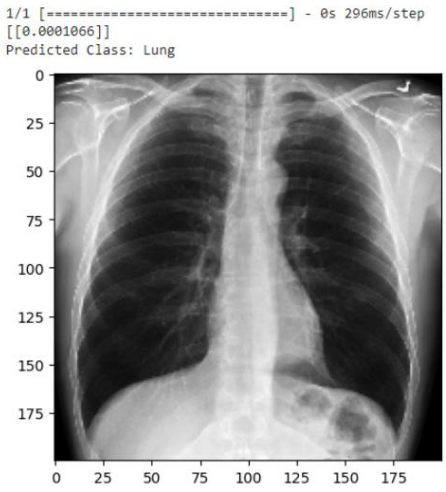
The image is converted to a NumPy array and normalized (divided by 255) to match the preprocessing done on the training data.

‘np.expand_dims’ adds an extra dimension to the image array, converting it from a shape of (200, 200, 3) to (1, 200, 200, 3). This is necessary because the model expects input data in batches, even if only one image is being predicted.

The output, result, is a probability score. In binary classification with a sigmoid activation function, a threshold of 0.5 is typically used to classify the results: if the score is 0.5 or higher, the image is predicted as belonging to one class ("Other" in this case), and if it is lower than 0.5, it is predicted as the other class ("Lung").

5.4.4 Model Result





5.5 Heat Map

5.5.1 Xception Model

```
model_builder = keras.applications.xception.Xception
img_size = (299, 299)
preprocess_input = keras.applications.xception.preprocess_input
decode_predictions = keras.applications.xception.decode_predictions
last_conv_layer_name = "block14_sepconv2_act"

def get_img_array(img_path, size):
    img = keras.preprocessing.image.load_img(img_path, target_size = size)
    array = keras.preprocessing.image.img_to_array(img)
    array = np.expand_dims(array, axis = 0)
    return array
```

This code configures the Xception model for image classification and Grad-CAM visualization. It defines model parameters, including the architecture, input size, preprocessing, and decoding functions. The function get_img_array loads and preprocesses an image, returning the corresponding numpy array for model input. This setup is essential for later implementing Grad-CAM visualization on the Xception model for a given image.

5.5.2 Generation Function

```
def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index = None):
    grad_model = tf.keras.models.Model([model.inputs], [model.get_layer(last_conv_layer_name).output, model.output])

    with tf.GradientTape() as tape:
        last_conv_layer_output, preds = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(preds[0])
        class_channel = preds[:, pred_index]

    grads = tape.gradient(class_channel, last_conv_layer_output)
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

    last_conv_layer_output = last_conv_layer_output[0]
    heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
    heatmap = tf.squeeze(heatmap)
    heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
    return heatmap.numpy()
```

The make_gradcam_heatmap function generates a Grad-CAM heatmap for an input image array using a pre-trained neural network model. It calculates gradients with respect to the last convolutional layer's output, weights the pixels based on these gradients, and produces a heatmap that highlights regions contributing most to the model's prediction for a specified class. The function returns the heatmap as a NumPy array.

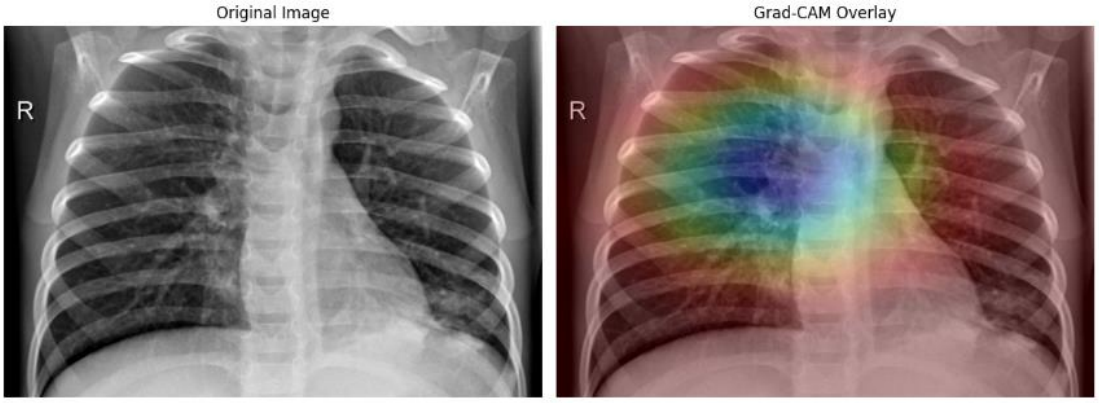
5.5.3 Display Overlay

```
def save_and_display_gradcam(img_path, heatmap, cam_path = "cam.jpg", alpha = 0.4):
    img = keras.preprocessing.image.load_img(img_path)
    img = keras.preprocessing.image.img_to_array(img)

    heatmap = np.uint8(255 * heatmap)

    jet = cm.get_cmap("jet")
    jet_colors = jet(np.arange(256))[:, :3]
    jet_heatmap = jet_colors[heatmap]
    jet_heatmap = keras.preprocessing.image.array_to_img(jet_heatmap)
    jet_heatmap = jet_heatmap.resize((img.shape[1], img.shape[0]))
    jet_heatmap = keras.preprocessing.image.img_to_array(jet_heatmap)

    superimposed_img = jet_heatmap * alpha + img
    superimposed_img = keras.preprocessing.image.array_to_img(superimposed_img)
    superimposed_img.save(cam_path)

    # Display the original image and the Grad-CAM overlay
    fig, axes = plt.subplots(1, 2, figsize=(12, 6))
    axes[0].imshow(img / 255.0)
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    cam = cv2.imread(cam_path)
    axes[1].imshow(cam / 255.0)
    axes[1].set_title('Grad-CAM Overlay')
    axes[1].axis('off')

    plt.tight_layout()
    plt.show()
```

The save_and_display_gradcam function processes an input image and Grad-CAM heatmap, creating a blended overlay image. It saves the overlay and displays the original image alongside it. This function is designed for visualizing the model's focus areas in the original image during prediction.

5.5.4 Testing

```
# Define your image path
image_path = "/content/drive/MyDrive/COVID-19_Radiography_Dataset/Test_Images/PNEUMONIA(4023).jpg"

# Load and preprocess the image
img_array = preprocess_input(get_img_array(image_path, size=img_size))

# Load the Xception model
model = model_builder(weights="imagenet")
model.layers[-1].activation = None

# Generate the Grad-CAM heatmap
heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)

# Save and display the Grad-CAM
save_and_display_gradcam(image_path, heatmap)
```



There are several reasons why Grad-CAM visualization can be important:

- Model understanding: Grad-CAM visualization can be useful for understanding how a deep learning model is making its predictions. By highlighting the regions of the image that are most important for the model's predictions, Grad-CAM can help to provide insight into the decision-making process of the model.
- Model debugging: Grad-CAM visualization can also be useful for debugging problems with a deep learning model. For example, if the model is making incorrect predictions, Grad-CAM can help to identify which parts of the image the model is focusing on and potentially identify any issues with the model's training or architecture.
- Model explanation: Grad-CAM visualization can be useful for explaining the predictions of a deep learning model to non-technical stakeholders. By highlighting the regions of the image that are most important for the model's predictions, Grad-CAM can provide a more intuitive understanding of the model's decision-making process.

6. Test Plan

User	Features	Scenario Description	Test Case ID	Status
All User	Login	Existing app users must be able to successfully log in to their account	TC A1.1	PASS
	Login	User key in wrong username or password or both, error message appears to prompt re-entry	TC A1.2	PASS
	Login	User clicks on login button without filling in either username or password or both	TC A1.3	PASS
	Logout	When user hovers on right profile icon and clicks on logout under the navigation bar on any page, they will be successfully redirected to the login page and session token is removed	TC A2.1	PASS
	Main page Dashboard	Doctor dashboard can display all records that was created by the doctor profile that is logged in.	TC B1.1	PASS
	Main page Dashboard	Doctor can edit record comments by clicking on the edit icon on the dashboard and redirected to the doctorViewResult page to edit the comments.	TC B1.2	PASS
	Main page Dashboard	Doctor can release report to patient by clicking on the release checkbox to a “Tick” to enable visibility of the report to the patients.	TC B1.3	PASS
	Main page Dashboard	Doctor can hide report to patient by clicking on the release checkbox to “Un-Tick” to hide report from the patients.	TC B1.4	PASS
	Upload page	User hover on left navigation bar tab and clicks on upload will be redirected to upload page	TC B2.1	PASS
	Upload page	Doctor able to upload a valid X-ray image and existing patient ID and clicks on “Analyse Image”. Upon successful analysis, “Create Report” button will appear.	TC B2.2	PASS
	Upload page	Doctor clicks on analyse button with out uploading image and patient ID, 2 error messages will be displayed prompting to upload an image and fill in patient field.	TC B2.3	PASS
	Upload page	Doctor uploads X-ray image but did not fill in patient ID when clicking on “Analysis image” button, error message	TC B2.4	

		will be displayed prompting to fill in patient ID.		PASS
	Upload page	Doctor did not upload X-ray image but fills in patient ID when clicking on "Analysis image" button, error message will be displayed prompting to upload an image.	TC B2.5	PASS
	Upload page	Doctor clicks on "Analysis Image" button but did not upload valid Xray image. Error message will prompt user to select a valid lung X-ray image.	TC B2.6	PASS
	Upload page	Doctor can create report by clicking on "Create Report" button and redirected to doctorViewResult page to view the full report that is created.	TC B2.7	PASS
	Doctor view result page	Doctor can view full report inside the doctorViewResult page	TC B3.1	PASS
	Doctor view result page	Doctor can add or edit comments, by clicking on the "Submit" button. A message will be displayed indicating comment successfully updated and redirect to main page.	TC B3.2	PASS
	Doctor view result page	Doctor clicks on "Submit" button but no changes were made, error message indicating no changes to save will be displayed. No changes made	TC B3.3	PASS
	Doctor view result page	Doctor clicks on "Submit" button and comment field is empty, error message comments cannot be empty will be displayed. No changes made	TC B3.4	PASS
Patient	Create Account	Patient successfully create account, message showing account successfully created and redirected to login page	TC C1.1	PASS
	Create Account	Patient enters firstName and lastName with numbers or symbols, error message will prompt user to only key in alphabets	TC C1.2	PASS
	Create Account	Patient is not able to enter phone number with alphabets, symbols or more than 10 digits	TC C1.3	PASS
	Create Account	Patient will not be able to create new account if account, phone number, email or username already exist. An error message will be prompted indicating account already exist	TC C1.4	PASS
	Main page Dashboard	Patient dashboard can display all records that was created by the doctor	TC C2.1	PASS
	Main page Dashboard	Patient can view record by clicking on the eye icon on the dashboard and	TC C2.2	

		redirected to the patientViewReport page		PASS
	Main page Dashboard	Patient unable view record when clicking on the dashed eye icon on the dashboard. No action performed	TC C2.3	PASS
	Patient view report page	Patient can view full report inside the patientViewReport page	TC C3.1	PASS
	Patient view report page	Patient can print and save the report by clicking on the “print and save” button	TC C3.2	PASS

Appendix

[1] Hansell DM, Bankier AA, MacMahon H, McLoud TC, Muller NL, Remy J. Fleischner society: glossary of terms for thoracic imaging. Radiology. 2008;246(3):697–722. doi: 10.1148/radiol.2462070712

[2] Khoo HWW, Hui TCH, Mohideen SMH, Lee YS, Liew CJY, Kok SSX, Young BE, Ong SWX, Kalimuddin S, Tan SY, Loh J, Chan LP, Poh ACC, Wong SBS, Leo YS, Lye DC, Kaw GJL, Tan CH. Radiographic features of COVID-19 based on an initial cohort of 96 patients in Singapore. Singapore Med J. 2021 Sep;62(9):458-465. doi: 10.11622/smedj.2020142. Epub 2020 Sep 21. PMID: 33047143; PMCID: PMC9251244.

[3] Xin, M., Wang, Y. Research on image classification model based on deep convolution neural network. J Image Video Proc. 2019, 40 (2019). https://doi.org/10.1186/s13640-019-0417-8

(2022) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=KuXjwB4LzSA

Rovai, M. (2020) Applying artificial intelligence techniques in the development of a web-app for the detection of..., Medium. Available at: https://towardsdatascience.com/applying-artificial-intelligence-techniques-in-the-development-of-a-web-app-for-the-detection-of-9225f0225b4 (Accessed: 26 July 2023).

Rosebrock, A. (2021) Detecting covid-19 in X-ray images with Keras, tensorflow, and Deep Learning, PyImageSearch. Available at: https://pyimagesearch.com/2020/03/16/detecting-covid-19-in-x-ray-images-with-keras-tensorflow-and-deep-learning/ (Accessed: 26 July 2023).

******COVID-19 CHEST X-RAY DATABASE*
-M.E.H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M.A. Kadir, Z.B. Mahbub, K.R. Islam, M.S. Khan, A. Iqbal, N. Al-Emadi, M.B.I. Reaz, M. T. Islam, "Can AI help in screening Viral and COVID-19 pneumonia?" IEEE Access, Vol. 8, 2020, pp. 132665 - 132676.
-Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S. and Chowdhury, M.E., 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images. arXiv preprint arXiv:2012.02238.

***Reference:*
[1]https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711
[2]https://github.com/ml-workgroup/covid-19-image-repository/tree/master/png
[3]https://sirm.org/category/senza-categoria/covid-19/
[4]https://eurorad.org
[5]https://github.com/ieee8023/covid-chestxray-dataset
[6]https://figshare.com/articles/COVID-19_Chest_X-Ray_Image_Repository/12580328
[7]https://github.com/armiro/COVID-CXNet
[8]https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data
[9] https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

****Formats*

*- All the images are in Portable Network Graphics (PNG) file format and resolution are 299*299 pixels.*

*****Objective*

- Researchers can use this database to produce useful and impactful scholarly work on COVID-19, which can help in tackling this pandemic.