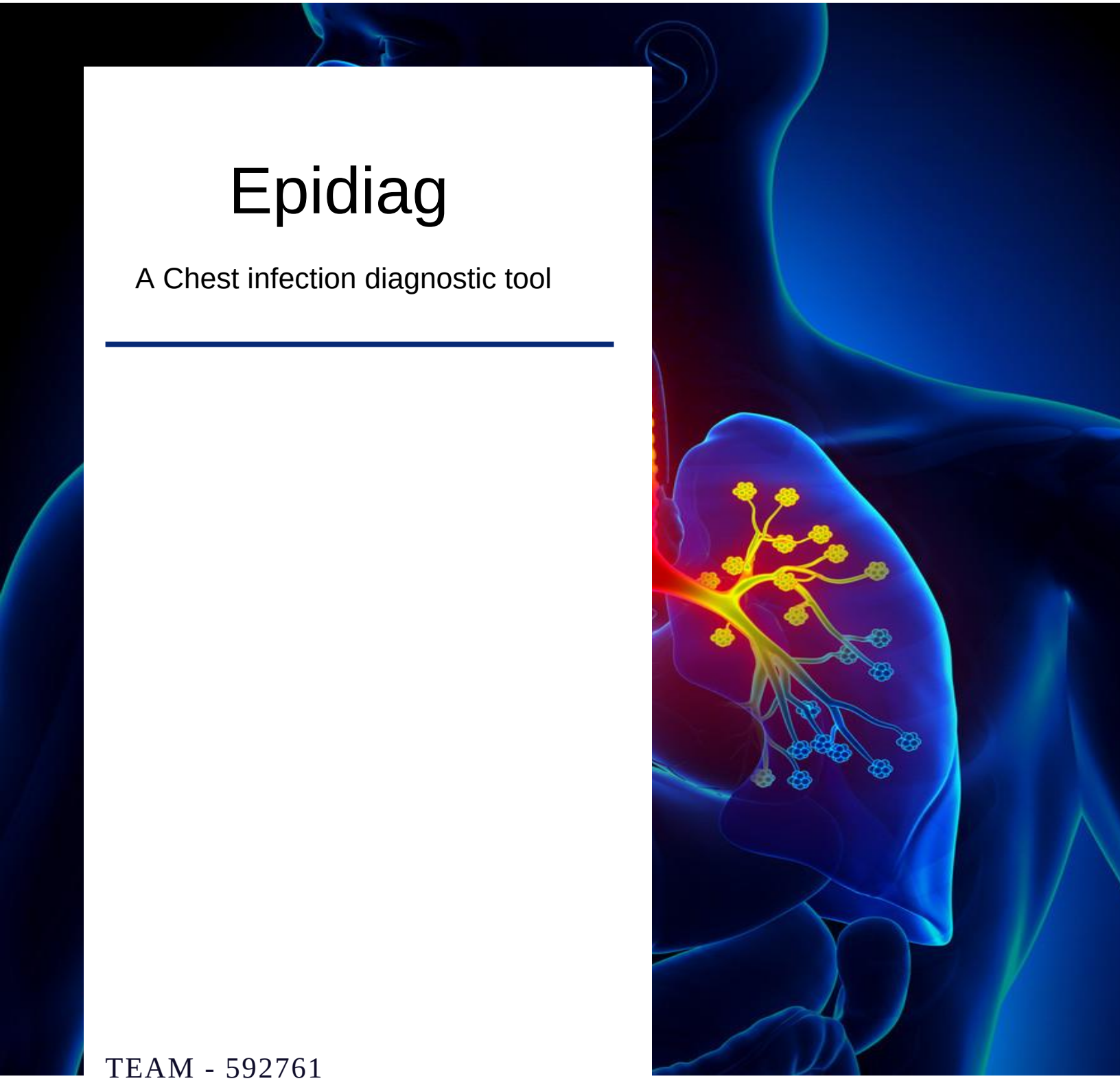
Epidiag

A Chest infection diagnostic tool



TEAM - 592761

Authored by:
1. Harish Thangaraj
2. Kanishka Verma

Table of contents

1. Introduction

1.1 Project Overview

The bloom in AI technology has enabled human like intellectualism in almost all domains and is being pushed to its limits. The healthcare industry is a crucial domain keeping in mind the stressful job nature of workers to provide accurate diagnostics and the stakes are simply high. As engineering students, we see the potential of implementing AI stack here along with business incentives. Machine learning algorithms are not bound by human like exhaustion and perform with high level of accuracy and reliability. The usage of such algorithms yields an error free disease detection process which improves the overall well-being of its users.

1.2 Purpose

The detection of chest infections involves a combination of physical and diagnostic tests. Chest x-rays and CT scan reports are important to detect such infections and usually need the involvement of a doctor to understand it. Our aim is to build a deep learning model that can aid to diagnose infections using images of chest X-rays and present results directly to the user irrespective of their qualification. This project bridges the gap between patients and straightforward treatment.

2. Literature Survey

2.1 Existing Problem

The detection of chest infections from chest x-rays using deep learning models is a well explored research problem and there are several potential solutions presented across journals. One of the common challenges faced in developing such a solution is the scarce availability of data quality and quantity used for training and testing purposes. This causes the model to underperform and yield poor results which can be catastrophic in an industry like healthcare.

2.2 References

Article	Journal	Authors
Diagnosis of pneumonia from chest x-ray images using deep learning	IEEE	Enes Ayan ; Halil Mural Unver
Deep learning on chest x-rays to detect and evaluate pneumonia cases at the era of COVID - 19	Journal of medical systems	Karim Hammoudi ; Halim Benhabiles
A deep learning approach for automatic detection of COVID-19 cases using chest x- ray images	Biomedical signal processing and control	Abhijit Bhattacharyya; Sunil Kumar
A novel transfer learning-based approach for pneumonia detection in chest x-ray images.	Applied sciences	Vikash Chouhan; Deepak Gupta
Automated deep transfer learning – based approach for detection of COVID – 19 infection in chest X-rays	IRBM	N.Narayan Das ; N. Kumar; M. Kaur

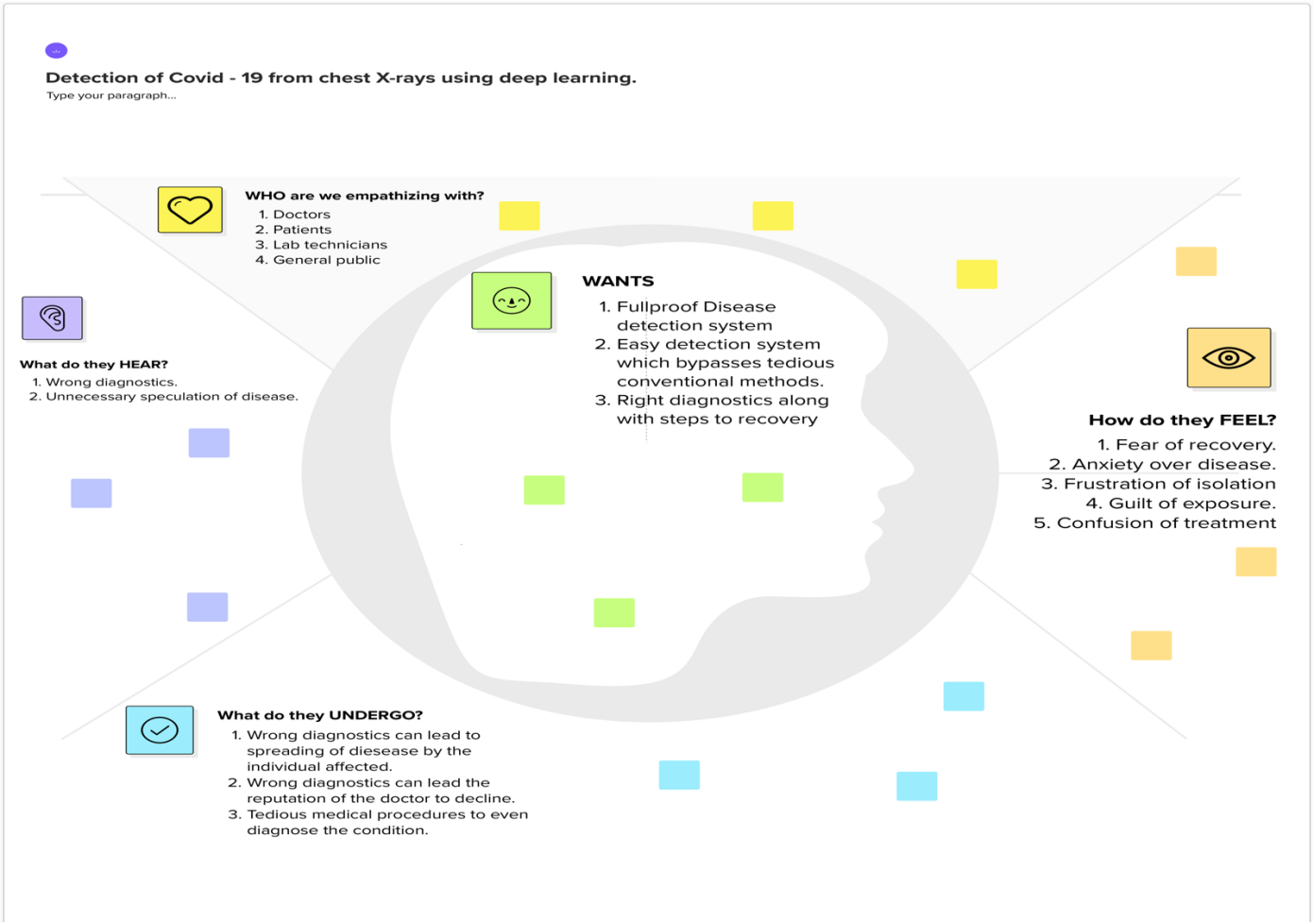
2.3 Problem Statement Definition

Developing a transfer learning model for the automated and accurate detection of chest infections from chest x-ray images to assist patients and healthcare workers in early diagnosis and treatment planning.

3. Ideation And Proposed Solution

3.1 Empathy Map Canvas

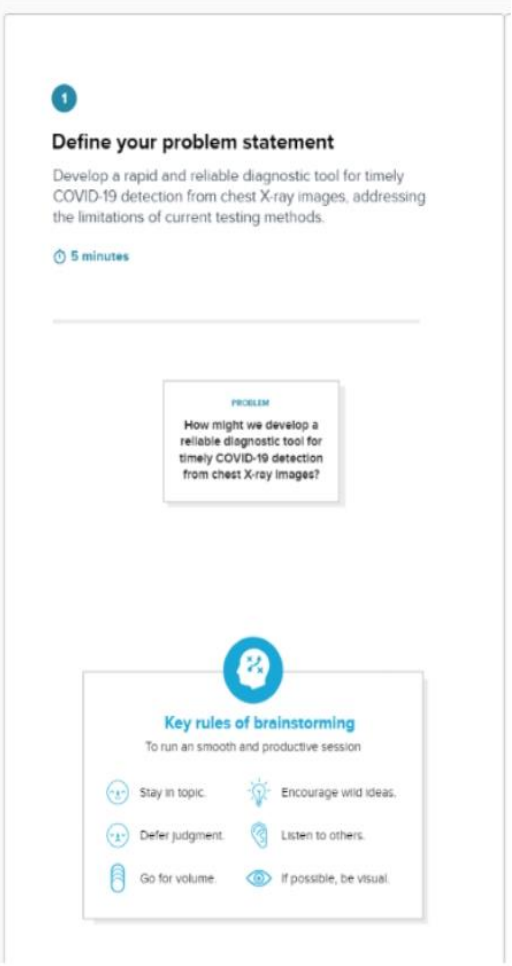
Empathy map helps empathise with and understand prospect users. It summarizes their needs, motivations, and behaviours which allows developers to build viable applications.



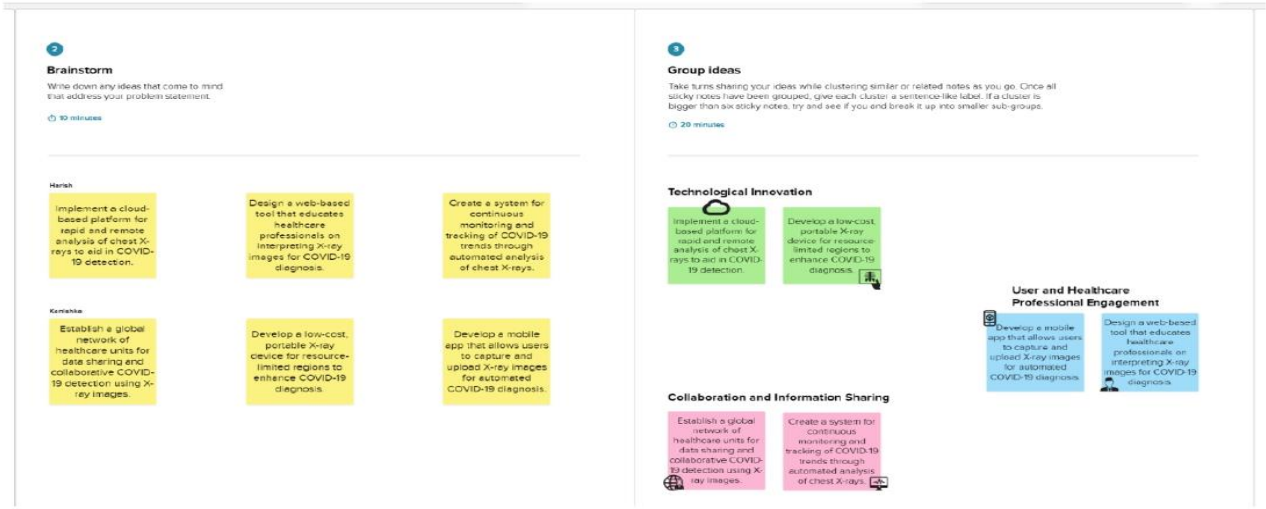
3.2 Ideation And Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing value over volume, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate and help each other to develop a rich amount of creative solutions.

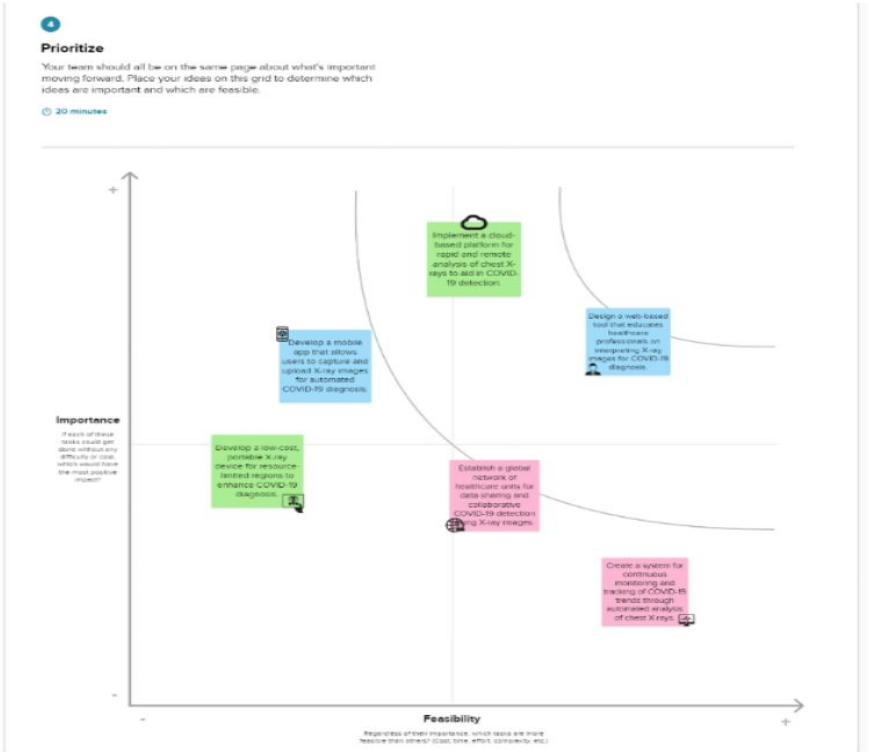
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



4. Requirement Analysis

4.1 Functional Requirements

Defines what a product must do and what its features and functions are.

Sl No.	Functional requirement	Sub requirement
1.	Data collection and processing	Diverse dataset with both positive and negative x-rays.
2.	Image enhancement tools	Enhancement tools such as CLAHE and contrast to enable better detection.
3.	Feature extraction	Convolution layers of the pre-trained CNN model to extract features and observe anomalies.
4.	Transfer learning model	MobileNetV2 pre-trained CNN model based on which the transfer learning model is deployed.
5.	Prediction and classification	User x-ray uploading provision to classify and diagnose the disease.
6.	User interface to interact with the model	A flask-based python web application to allow users to interact and diagnose disease.

4.2 Non-functional Requirements

Defines how the system should perform and focuses on user expectations.

Sl No.	Non-functional requirement	Description
1.	Accuracy and reliability	The system ensures accurate classification with a low false positive and false negative rate.
2.	Scalability	The model can be re-trained easily with updated datasets and ensures scalability of detection.
3.	Usability and accessibility	The front-end web page is easy to understand and use even for a non-technical persona.
4.	Security and privacy	The code ensures the uploaded chest x-rays are deleted after classification and avoids any sort of data leakage.

5. Project Design

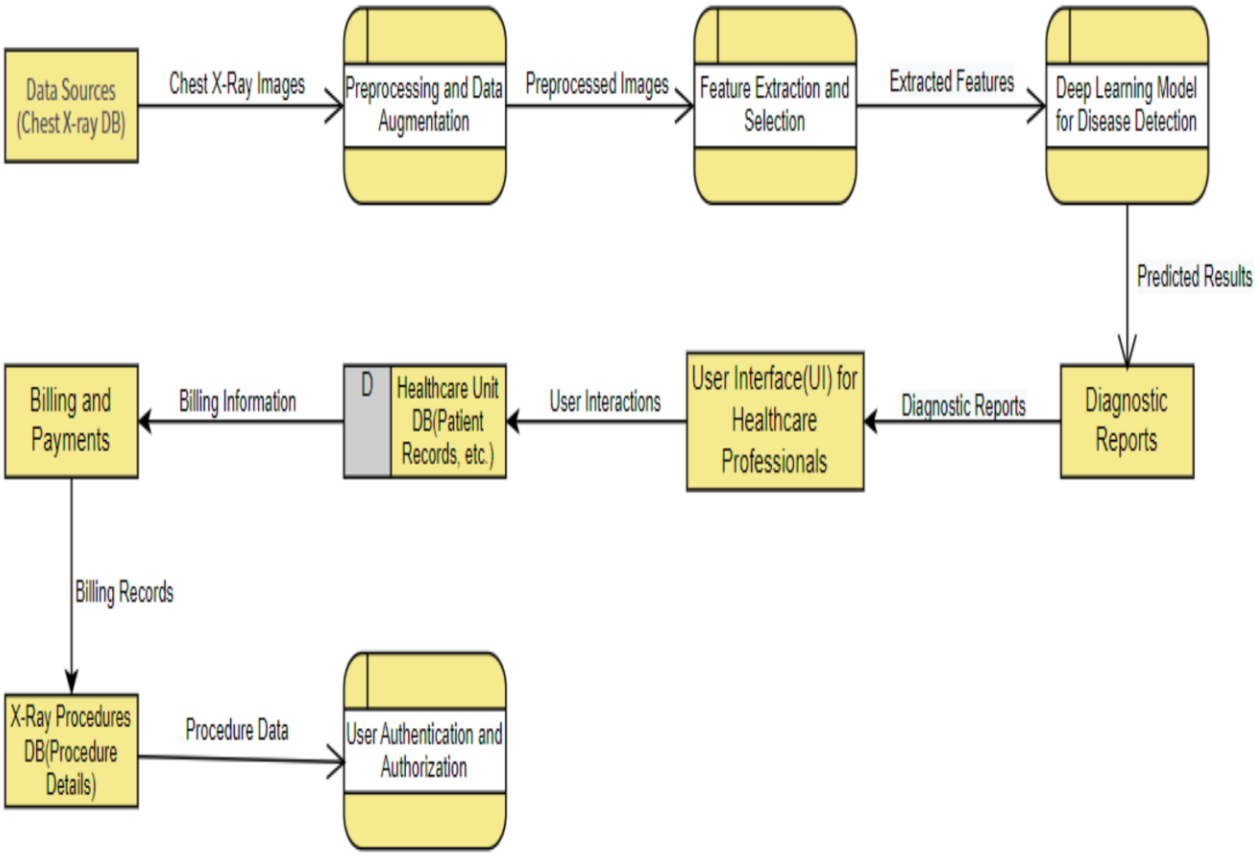
5.1 Data Flow Diagrams And User Stories

A traditional visual representation of the information flows within a system. It shows how data enters and leaves the system, what changes the information, and where data is stored.

DFD Components:

1. **Data Sources:** Chest x-ray images are collected from various sources, such as a database of x-ray images.
2. **Pre-processing & Data Augmentation:** The input x-ray images go through pre-processing and data augmentation to enhance the data quality and prepare it for feature extraction.
3. **Feature Extraction and Selection:** This step involves extracting relevant features from pre-processed images and selecting the most informative ones for the deep learning model.
4. **Deep Learning Model for Disease Detection:** The core of the system, where the deep learning model uses the extracted features to detect and diagnose diseases. It also autonomously updates its learning.
5. **Diagnostic Reports:** The model generates diagnostic reports based on its predictions and stores them for further use.
6. **User Interface (UI) for Healthcare Professionals:** The interface through which healthcare professionals interact with the system, upload images, and access diagnostic reports.
7. **Healthcare Unit Database:** A database where patient records and diagnostic reports are stored, allowing healthcare professionals to access historical data.

Dataflow diagram:

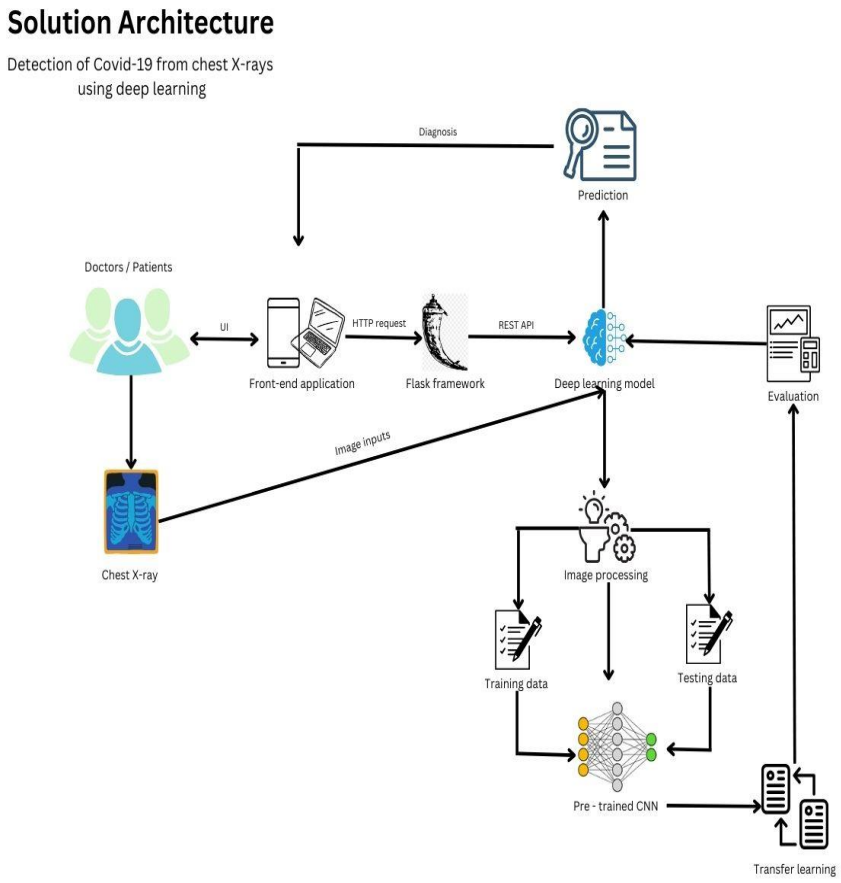


User stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Healthcare Professional	Disease Diagnosis	USN-1	As a healthcare professional, I can upload chest X-ray images to the system for disease diagnosis	The system allows me to upload X-ray images and provides diagnostic reports for the uploaded images	High	Sprint-1
	Access Diagnostic Reports	USN-2	As a healthcare professional, I can access diagnostic reports generated by the system	I can view and access diagnostic reports, including disease diagnosis results, patient information, and dates of diagnosis	High	Sprint-1
Billing Department	Billing and Payments	USN-3	As a billing department staff member, I can generate invoices for diagnostic services and process payments.	The billing department can generate invoices for diagnostic services, including detailed service information and process payments through various methods	High	Sprint-1
System Administrator	User Management	USN-4	As a system administrator, I can manage user accounts and their access to the system.	The system administrator can manage user accounts, assign roles and permissions, and enforce security and access controls	High	Sprint-1
Healthcare Professional	X-Ray Procedure Information	USN-5	As a healthcare professional, I want access to information about X-ray procedures available in the system.	Healthcare professionals can access information about X-ray procedures, including descriptions, costs, and billing codes	Medium	Sprint-1
System Administrator	User Authentication and Authorization	USN-6	As a system administrator, I want to ensure secure user authentication and authorization for system access.	The system administrator can configure secure user authentication methods, define user roles and access rights, and enforce strong password policies.	High	Sprint-1

5.2 Solution Architecture

Refers to the design and structure of a software solution or system. It involves creating a comprehensive plan that outlines the components, modules, interactions, technologies, and overall layout required to meet specific business or technical needs.

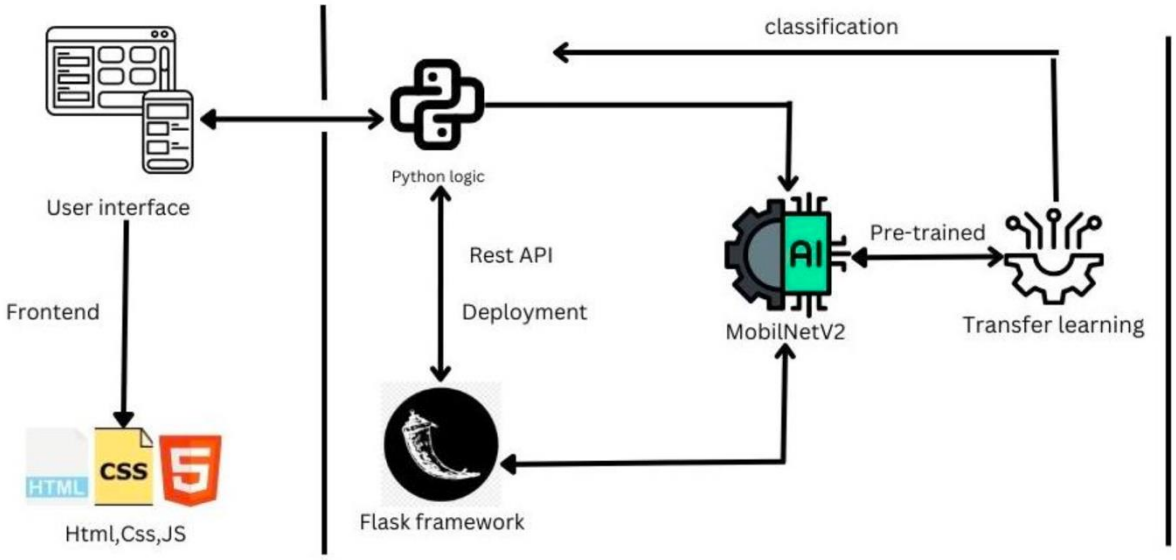


6. Project Planning And Scheduling

6.1 Technical Architecture

Refers to the structure of a technology solution. It encompasses the layout, components, and the way in which these components interact to present a full scale application.

Technical architecture diagram:



Components and technologies:

S.No	Component	Description	Technology
1.	User interface	Web UI that allows users to upload personal chest x-rays and classify them.	HTML, CSS, JS, Bootstrap, Font awesome.
2.	Application logic -1	Logic to receive user input as Images only and store its Features in an array.	HTML, JS, Flask, NumPy, Python.
3.	Image enhancement	Image enhancement tools such as white balancing and CLAHE	OpenCV, Python, NumPy.
4.	Normalization of array	To achieve standard scaling of the data to reduce variations in prediction.	NumPy, Min-Max scaler, Python.
5.	Deep learning model	A transfer learning based model that is defined based on a pre-trained CNN model.	Tensorflow, MobilNetV2, Python, Scikit-learn, NumPy, Pandas, OpenCV, ImageNet.
6.	Infrastructure	The model is deployed using Flask framework and hosted Locally.	Local host, Flask

Application characteristics:

S.No	Characteristics	Description	Technology
1.	Frameworks	Deep learning and web frameworks.	Tensorflow, Flask, Keras.
2.	Scalability	The frameworks used can be easily retrained using to handle updated datasets.	Tensorflow, version Control system.
3.	Security	The user uploaded files are not stored permanently thereby ensuring no data leak.	Python logic, OS package.

6.2 Sprint Planning And Estimation

Sprint planning allows developers complete a set of tasks according to user interests in an organized and systematic manner.

Product backlog, spring schedule and estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection and Preprocessing	USN-1	As a developer, I need to create a script for X-ray data collection and preprocessing to prepare the dataset for training.	3	High	Harish
Sprint-1	Model Development	USN-2	As a data scientist, I need to develop a deep learning model using transfer learning algorithms for chest X-ray image analysis.	5	High	Kanishka
Sprint-2	Model Optimization	USN-3	As a machine learning engineer, I need to optimize the deep learning model to improve accuracy and efficiency.	5	High	Harish
Sprint-3	Real-time diagnosis	USN-4	As a user, I want the diagnostic tool to provide real-time diagnosis from chest X-ray images.	8	High	Harish, Kanishka

6.3 Sprint Delivery Schedule

Project tracker, Velocity and burndown chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2023	29 Oct 2023	20	29 Oct 2023
Sprint-2	20	6 Days	31 Oct 2023	05 Nov 2023	20	05 Nov 2023
Sprint-3	20	6 Days	07 Nov 2023	12 Nov 2023	20	12 Nov 2023

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Sprint Duration=6 days
Velocity of the team=20 (points per sprint)

Average velocity (AV) per iteration unit (story points per day): $AV = \frac{Velocity}{Sprint\ Duration} = \frac{20}{06} = 3.33$

7. Coding And Solutioning

7.1 Importing Necessary Libraries

Initializing python in jupyter notebook and importing all the necessary libraries required to build a full scale model.

```
import numpy as np
import matplotlib.pyplot as plt
import os
import random
import itertools
import cv2
import requests

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split

import ssl
ssl._create_default_https_context = ssl._create_unverified_context

import tensorflow as tf
from tensorflow.keras import *
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
from tensorflow.keras.callbacks import *
from tensorflow.keras.optimizers import *
from keras.utils import to_categorical
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.layers.experimental.preprocessing import RandomFlip, RandomRotation

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

7.2 Importing Dataset

Importing dataset picked up from Kaggle which contains directories for 3 classes of X rays : Covid-19, Normal, Viral pneumonia.

```
#Defining a function to import sample images from directory
def import_image_to_array(directory_path, size):
    # Define empty lists to store the images array and their respective labels
    image_arrays = []

    directory = os.listdir(directory_path)

    for image_name in directory:

        #Load an image from the specified directory
        image = cv2.imread(directory_path + image_name)

        #Resize image to the specified size
        image = cv2.resize(image, size)

        #Update the image dataset and labels lists respectively
        image_arrays.append(image)

    return image_arrays
```

```
#Defining directories and calling above function
NORMAL_DIR = "/Users/casarulez/Projects/Covid-19/COVID_IEEE/normal/"  # Normal CXR Image Dataset
VIRAL_DIR = "/Users/casarulez/Projects/Covid-19/COVID_IEEE/virus/"    # Viral Pneumonia CXR Image Dataset
COVID_DIR = "/Users/casarulez/Projects/Covid-19/COVID_IEEE/covid/"    # COVID-19 CXR Image Dataset

normal_dataset = import_image_to_array(NORMAL_DIR, (224,224))
viral_dataset = import_image_to_array(VIRAL_DIR, (224,224))
covid_dataset = import_image_to_array(COVID_DIR, (224,224))
```

```
#Printing dataset organization
print("Dataset architecture:")
print("Normal X-ray images:",len(normal_dataset))
print("Viral X-ray images:",len(viral_dataset))
print("Covid X-ray images:",len(covid_dataset))
```

```
Dataset architecture:
Normal X-ray images: 668
Viral X-ray images: 619
Covid X-ray images: 536
```

7.3 Image Processing

Since, medical images are underexposed which leads to crucial parts of the image appearing dark. To resolve this, white balancing and CLAHE is applied which enhances colour fidelity.

```
#Function to enhance images using CLAHE and white balancing.
def image_enhancer(image_arrays):
    enhanced_images = []

    for image in image_arrays:

        #White Balance
        image_WB  = np.dstack([white_balance(channel, 0.05) for channel in cv2.split(image)] )
        gray_image = cv2.cvtColor(image_WB, cv2.COLOR_RGB2GRAY)

        #CLAHE
        clahe_function = clahe()
        image_clahe = clahe_function.apply(gray_image)
        image = cv2.cvtColor(image_clahe, cv2.COLOR_GRAY2RGB)

        enhanced_images.append(image)

    return enhanced_images
```

```
normal_dataset_enhanced = image_enhancer(normal_dataset)
viral_dataset_enhanced = image_enhancer(viral_dataset)
covid_dataset_enhanced = image_enhancer(covid_dataset)
```

7.4 Normalizing Dataset

The dataset is normalized to bring down scale and reduce variations which ensures accurate classification

```
#Function to normalize dataset
def normalizer(image_arrays):
    # Create an empty list to store normalized arrays
    norm_image_arrays = []

    # Iterate over all the image arrays and normalize them before storing them into our predefined list
    for image_array in image_arrays:
        norm_image_array = image_array / 255.0
        norm_image_arrays.append(norm_image_array)

    return norm_image_arrays
```

```
normal_dataset_normalized = normalizer(normal_dataset_enhanced)
viral_dataset_normalized = normalizer(viral_dataset_enhanced)
covid_dataset_normalized = normalizer(covid_dataset_enhanced)
```

7.5 Train Test Validation Split

The data is split into training validation and testing for fitting in the model

```
#Combine the three classes into a single dataset
combined_dataset = np.concatenate((normal_dataset_normalized, viral_dataset_normalized, covid_dataset_normalized), a

#Generate labels for the three classes
normal_labels = np.zeros(len(normal_dataset_normalized))  # Label 0 for normal
viral_labels = np.ones(len(viral_dataset_normalized))      # Label 1 for viral
covid_labels = 2 * np.ones(len(covid_dataset_normalized))  # Label 2 for COVID

#Combine the labels into a single array
combined_labels = np.concatenate((normal_labels, viral_labels, covid_labels), axis=0)

#One hot encoding the labels
combined_labels = to_categorical(combined_labels, num_classes=3)

#Split the combined dataset and labels into training, validation, and test sets
train_data, test_data, train_labels, test_labels = train_test_split(combined_dataset, combined_labels, test_size=0.3
validation_data, test_data, validation_labels, test_labels = train_test_split(test_data, test_labels, test_size=0.5,
```

7.6 Defining A Transfer Learning Model

MobilNetV2(google) is used as pre-trained CNN model to facilitate transfer learning. The model is fit after defining custom parameters and saved.

```
def make_mobilenet_model(image_size, num_classes, data_augmentation = data_augmenter()):

    input_shape = image_size + (3,)

    base_model = tf.keras.applications.MobileNetV2(input_shape=input_shape,
                                                    include_top=False, # Do not include the dense prediction layer
                                                    weights="imagenet") # Load imageNet parameters

    #Freeze the base model by making it non trainable
    base_model.trainable = False

    #Create the input layer (Same as the imageNetv2 input size)
    inputs = tf.keras.Input(shape=input_shape)

    #Apply data augmentation to the inputs
    x = data_augmentation(inputs)

    #Set training to False to avoid keeping track of statistics in the batch norm layer
    x = base_model(x, training=False)

    #Add the new Binary classification layers
    #use global avg pooling to summarize the info in each channel
    x = GlobalAveragePooling2D()(x)
    #include dropout with probability of 0.2 to avoid overfitting
    x = Dropout(0.2)(x)

    #Create a prediction layer
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes

    x = layers.Dropout(0.5)(x)

    prediction_layer = Dense(units, activation=activation)

    outputs = prediction_layer(x)

    model = Model(inputs, outputs)

    return model
```

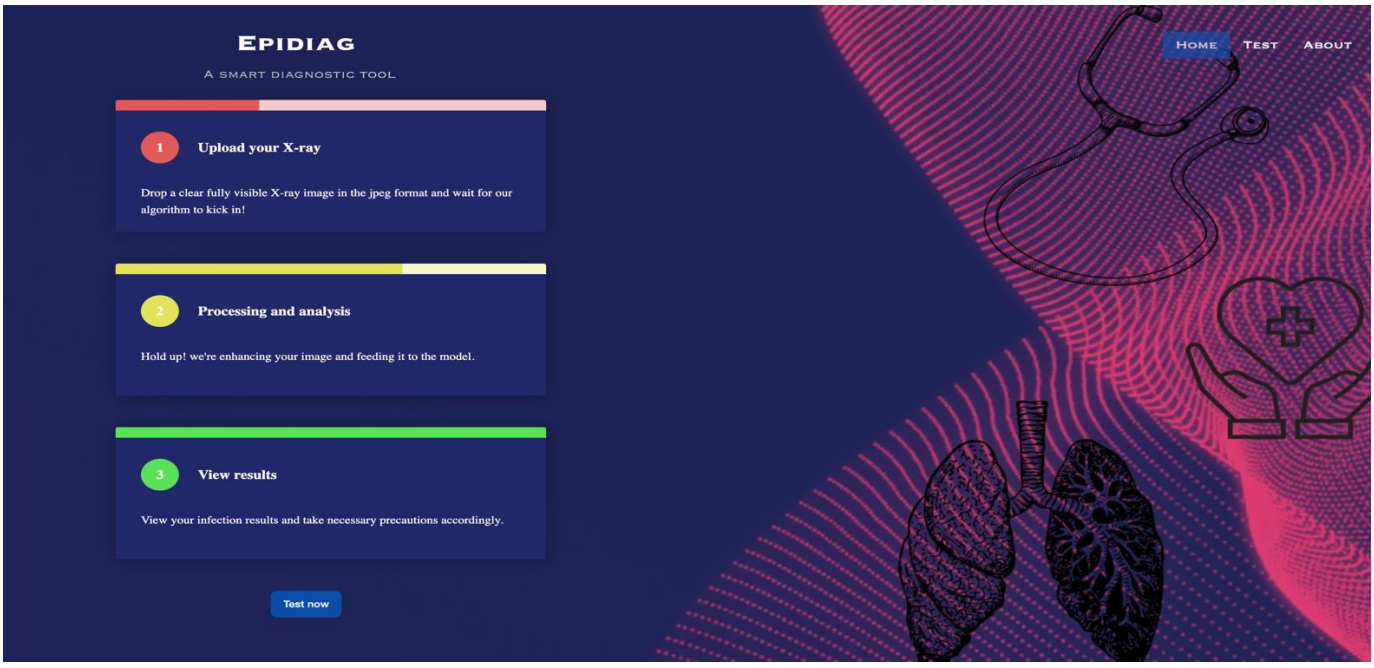
7.7 Flask Deployment

The above model is passed to the flask web framework and deployed to a front end website

```python
@app.route('/diagnose', methods=['POST'])
def diagnose():
    file = request.files['image']
    if file:
        # Define the directory where you want to save the uploaded audio files
        upload_folder = 'uploads'

        # Ensure the 'uploads' directory exists
        os.makedirs(upload_folder, exist_ok=True)

        # Save the uploaded audio file to the specified directory
        file.save(os.path.join(upload_folder, file.filename))
        temp_path=os.path.join(upload_folder,file.filename)
        output=upload(temp_path)

        return render_template('output.html', text="{}".format(output))
```

7.8 Frontend Website

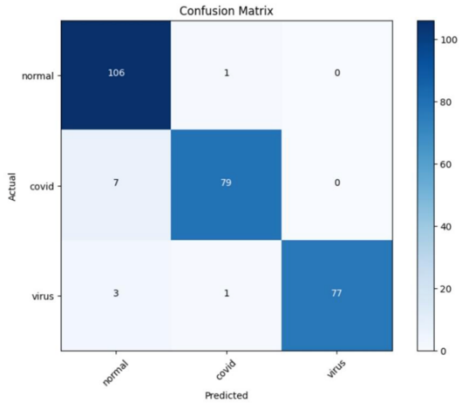
The website is built using HTML,CSS and JS. The interface allows users to upload their personal chest X-rays and diagnose the infection.



8. Performance testing

8.1 Performance Metrics and Tuning

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** Confusion Matrix, Accuracy Score & Classification Report	**Confusion Matrix:**  **Accuracy Score:**

	precision	recall	f1-score	support
normal	0.94	0.95	0.94	107
covid	0.94	0.95	0.95	86
virus	1.00	0.96	0.98	81
accuracy			0.96	274
macro avg	0.96	0.96	0.96	274
weighted avg	0.96	0.96	0.96	274

			Classification Report:
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	**Metrics Used:**

```
base_mobilenet_model = mobilenet_model.layers[2] #MobileNetV2 Architecture
base_mobilenet_model.trainable = True

#The MobileNet Model has 155 layers (the prediction layer inclusive)
#Fine-tune from this layer onwards
fine_tune_at = 120

#Freeze all the layers before the `fine_tune_at` layer
for layer in base_mobilenet_model.layers[:fine_tune_at]:
    layer.trainable = True


optimizer = Adam(learning_rate = 0.1 * base_learning_rate)
batch_size = 64
loss = 'categorical_crossentropy'
metrics = ['accuracy']
callback = EarlyStopping(monitor='val_accuracy', patience=20, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy',factor=1e-1, patience=8, verbose=1, min_lr = 2e-6)

mobilenet_model.compile(optimizer = optimizer,
                        loss = loss,
                        metrics = metrics)
```

- MobileNetV2
- Loss Function
- Adam Optimizer
- EarlyStopping Callback
- ReduceLROnPlateau class

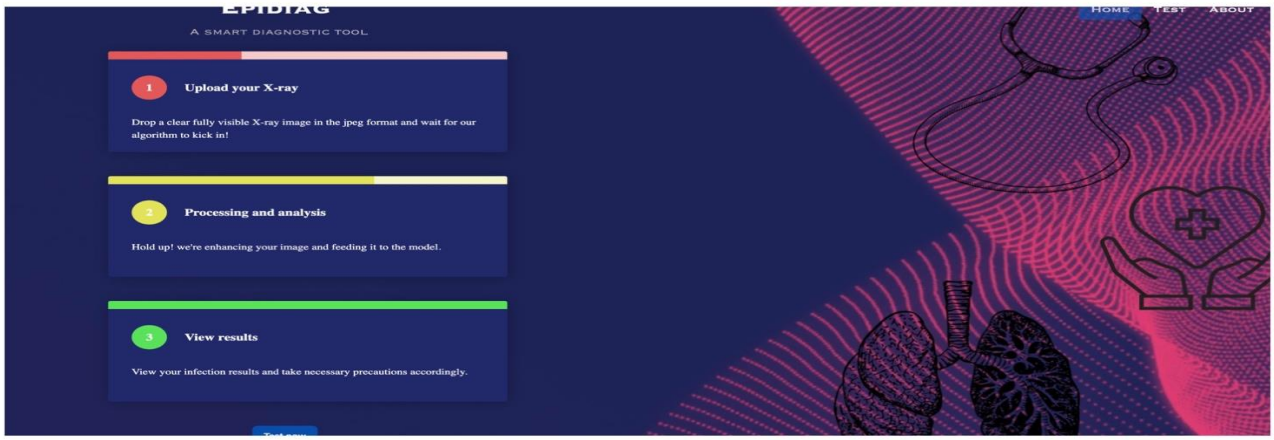
9. Results

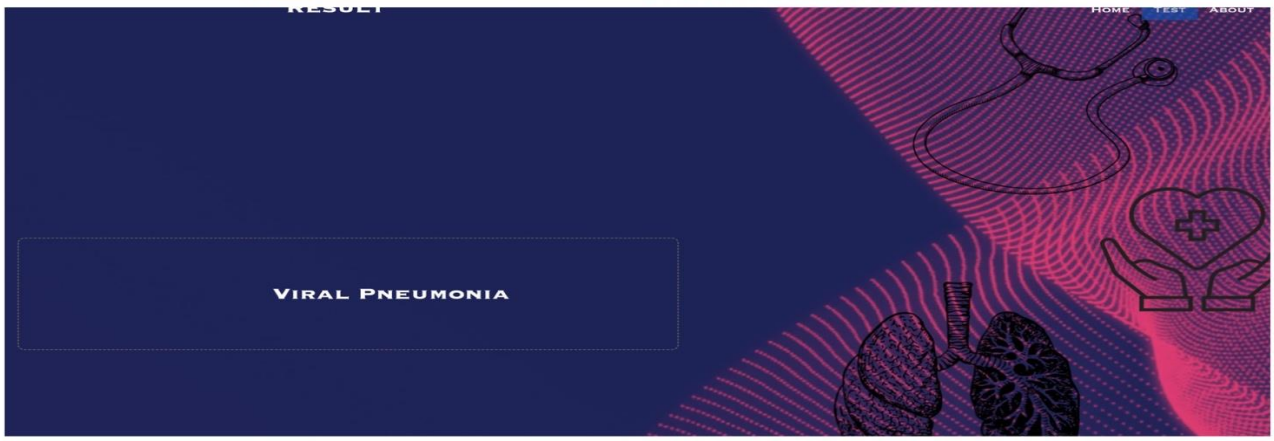
9.1 Output Screenshots:

```
def upload(directory,size):
    image_array=[]
    image = cv2.imread(directory)
    image = cv2.resize(image, size)
    image_array.append(image)
    image_enhanced=image_enhancer(image_array)
    image_normalized=normalizer(image_enhanced)
    image_np=np.array(image_normalized)
    class_names = {0: "Normal", 1 : "Viral Pneumonia", 2: "COVID-19"}
    predictions = mobilenet_model.predict(image_np)
    predicted_labels = np.argmax(predictions, axis=1)
    print(class_names[predicted_labels[0]])
```

```
upload("/Users/casarulez/Downloads/test/image.png",(224,224))
```

```
1/1 [==============================] - 0s 24ms/step
Viral Pneumonia
```





10. Advantages And Disadvantages

10.1 Advantages

1. **Speed and efficiency:** Deep learning algorithms can analyse large volumes of chest x-ray images quickly, allowing for rapid detection and diagnosis of potential COVID-19 cases. The speed is particularly beneficial in handling large numbers of patients during a pandemic.
2. **Potential for early detection:** Deep learning models can potentially detect chest infections in the lungs even before symptoms manifest, aiding in early diagnosis and isolation to prevent further transmission.
3. **Reduced dependency on testing kits:** As an auxiliary tool, it can complement traditional testing method like RT-PCR, especially in situations where testing kits might be unavailable.
4. **Automation and standardization:** Deep learning algorithms can assist in automating the analysis of x-ray images, reducing the dependency on individual expertise and potentially standardizing the interpretation process.

10.2 Disadvantages

1. **Accuracy and reliability:** The performance of deep learning models heavily relies on the quality and quantity of data used for training. If the dataset is biased, limited or not representative, the model's accuracy can be compromised, leading to misdiagnosis.
2. **Interpretability and lack of explainability:** Deep learning models often operate as 'black boxes', making it difficult to understand how they arrive at a particular diagnosis. Lack of interpretability might cause concerns among healthcare professionals and patients about trusting the model's decisions.
3. **Dependency on image quality:** The accuracy of deep learning models can be affected by the quality of the chest x-ray images. Factors like positioning, artefacts and quality of imaging equipment can influence the model's performance.
4. **Risk of over-reliance:** Relying solely on AI-based diagnosis might lead to over-reliance on the technology, potentially undermining the role of healthcare professionals and their expertise.

11. Conclusion

The application was successfully researched, developed and deployed with the help of our mentors and team members through every phase.

We gained insight into the vital role AI technology plays in not only aiding but also pushing beyond human limitations. The healthcare industry is one such place where the deployment of smart machines could bring about major safety revolutions. As developers, we managed to understand the technology stack working behind such applications and build an accurate and scalable application. Finally, we understood the process of user research and implemented the same in our workflow.

12. Future Scope

The future development goals planned are as follows:

1. Implementation of a personal diagnostic chatbot that analysis results and gives immediate home remedies.
2. Personal profile page where users can access old results and maintain their diagnostic portfolio.
3. A much more interactive UI and mobile application for handy use.

The success of this project opens avenues for further research and development in the field of deep learning. Many more such applications are expected to enter every industry and ease the burden on workers and improve overall efficiency.

13. Appendix

1. **GitHub repository :** https://github.com/smartinternz02/SI-GuidedProject-604390-1697705104/tree/main

2. **Directory structure:**

```
Repository/
|
|-Ideation phase/
|           |-Empathy map
|           |-Brainstorming
|-Design phase/
|           |-Proposed solution
|           |-Solution architecture
|           |-Data flow diagram and user stories
|-Planning phase/
|           |-Project planning
|           |-Tech stack
|-Development phase/
|           |-app (Flask app)/
|           |       |-app.py
|           |       |-static
|           |       |-templates
|           |-Covid-19 – classification.ipynb (Source code)
|           |-requirements.txt (dependencies required)
|- Submission phase/
|           |-Performance testing
|           |-Final report
|-Readme.md
```

3. Refer to readme.md to run the application on your local system.

4. **Video presentation :**
   https://drive.google.com/file/d/1nYeGNG0KX3BzjMsuDT8e_8iwQ6FrQpqQ/view?usp=sharing