PROFESSIONAL READINESS FOR INNOVATION,

EMPLOYABILITY AND ENRREPRENEURSHIP

A PROJECT REPORT ON

ADVANCED COVID-19 DETECTION FROM LUNG XRAYS WITH MACHINE LEARNING OR DEEP LEARNINGS

Project carried out

By

BHARATHI SHAVAKKAR.K S

ABINAYA.S

LOKESH.V

NITHYA.R

ANNA UNIVERSITY REGIONAL CAMPUS, COIMBATORE

MAY 2023

BONAFIDE CERTIFICATE

Certificate that this project report " **ADVANCED COVID-19 DETECTION FROM LUNG XRAYS WITH MACHINE LEARNING OR DEEP LEARNINGS** " is the bonafide work of "**BHARATHI SHAVAKKAR K S, (710020104005), ABINAYA S (710020104002), LOKESH V (710020104017), NITHYA R (710020104310)**" who carried out the project work under my supervision.

SIGNATURE
STAFF IN-CHARGE
(Naan Mudhalvan)

Submitted for the project held at ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE on ___________

ABSTRACT

The COVID-19 pandemic has posed unprecedented challenges to global healthcare systems, emphasizing the need for accurate and efficient diagnostic tools. The interpretation of medical images, such as X-rays, is subjective and can vary among different healthcare professionals. This variability can lead to inconsistencies and discrepancies in the diagnosis of COVID-19, potentially impacting patient care.COVID-19 diagnosis based on X-rays requires the expertise of skilled radiologists who can accurately analyze and interpret the images. However, this process can be time-consuming, especially in scenarios where there is a high volume of cases and limited availability of specialized personnel.The manual analysis of X-rays for COVID-19 detection has limitations in terms of scalability. As the number of cases increases, it becomes challenging for healthcare systems to efficiently process and diagnose a large volume of X-ray images within a reasonable timeframe.There may be a lack of standardized guidelines or protocols for COVID-19 diagnosis based on X-rays.X-ray imaging has emerged as a valuable modality for detecting COVID-19-related lung abnormalities. In this project, we aim to develop a deep learning-based system for COVID-19 detection using X-ray images. By harnessing the power of artificial intelligence, CovidVision aims to provide a reliable and automated solution to assist medical professionals in efficiently diagnosing COVID-19 cases.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

1.1.1 OBJECTIVE

The project aims to develop a deep learning-based system for the detection of COVID-19 from X-ray images. By leveraging the power of deep learning algorithms, the objective is to improve the accuracy and efficiency of COVID-19 diagnosis, assisting healthcare professionals in making informed decisions.

1.1.2 OVERVIEW

The project involves several key steps. First, a diverse dataset of chest X-ray images, consisting of both COVID-19 positive and negative cases, is collected and curated. Preprocessing techniques are applied to enhance the quality and standardization of the images. Next, a suitable deep learning architecture is chosen, such as Convolutional Neural Networks (CNNs), that can effectively extract features from the X-ray images. The selected model is trained on the preprocessed dataset, with the aim of optimizing its parameters for optimal performance.

Once trained, the model is evaluated using various performance metrics, such as accuracy, precision, recall, and F1-score. The results are validated using a separate test dataset to ensure the robustness and generalization of the model. Further improvements are made through fine-tuning and hyperparameter optimization, adjusting parameters like learning rate, batch size, and regularization techniques. This iterative process aims to enhance the model's performance and optimize its ability to detect COVID-19 accurately.

The developed deep learning model is then deployed through a user-friendly interface or application, allowing healthcare professionals to input X-ray images and obtain COVID-19 detection results. Interpretability and explainability techniques are also explored to provide insights into the model's decision-making process. Overall, this project aims to contribute to the fight against the COVID-19 pandemic by developing an advanced deep learning-based system for the detection of COVID-19 from X-ray images. The goal is to improve diagnostic accuracy, support healthcare professionals, and enhance patient care during these challenging times.

1.2 PROJECT PURPOSE

The purpose of the project is to develop a deep learning-based system for COVID-19 detection from X-ray images. The project serves several key purposes:

1. Accurate Diagnosis: The primary purpose is to improve the accuracy of COVID-19 diagnosis from X-ray images. By leveraging deep learning algorithms, the system aims to detect patterns and features indicative of COVID-19 with high precision, minimizing false positives and false negatives.

2. Timely Detection: Another purpose is to enable prompt COVID-19 detection. Deep learning models have the potential to analyze X-ray images rapidly, allowing for quick turnaround times in diagnosing suspected cases. This can aid in early identification and subsequent management of infected individuals.

3. Support Healthcare Professionals: The project aims to assist healthcare professionals in their decision-making process. By providing an automated system for COVID-19 detection, it can act as a valuable tool for radiologists and clinicians, offering reliable insights and enhancing their diagnostic capabilities.

4. Resource Optimization: The project can contribute to optimizing healthcare resources. By automating the COVID-19 detection process, the system can help alleviate the burden on medical professionals, allowing them to focus on

critical tasks while reducing the time and effort required for manual analysis of X-ray images.

5. Public Health Management: The development of an accurate and efficient COVID-19 detection system can aid in effective public health management. By providing a reliable tool for identifying COVID-19 cases, the project can support surveillance efforts, facilitate targeted interventions, and contribute to overall pandemic control and mitigation strategies.

Overall, the purpose of the project is to leverage deep learning techniques to improve COVID-19 detection accuracy, enable timely diagnosis, support healthcare professionals, optimize resources, and contribute to effective public health management during the ongoing COVID-19 pandemic.

CHAPTER 2

IDEATION AND PROPOSED SOLUTION

2.1 PROBLEM STATEMENT

There are several problems that researchers have encountered when using deep learning models for COVID-19 prediction. One of the main challenges is the lack of large-scale datasets. Another challenge is the lack of interpretability of deep learning models. Additionally, the performance of deep learning models can be affected by the quality of the data. Despite these challenges, deep learning models have shown promising results in predicting COVID-19 outcomes.

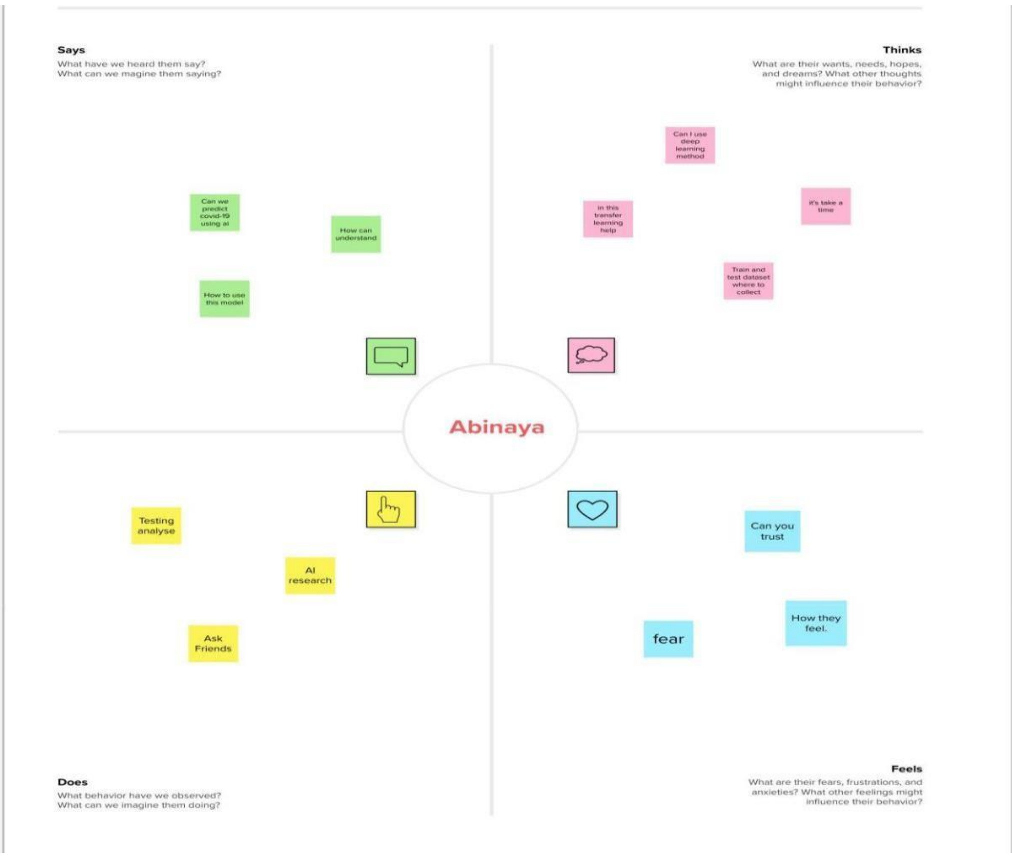
2.1.1 PROBLEM STATEMENTS FOR COVID -19 PROCESS:

2.1.2 TABLE FOR PROBLEM STATEMENTS COVID -19 :

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Patient	test a covid-19	It takes long time	to see a result	Fear
PS-2	Doctor	Predict a covid -19	It takes long time to analyse	To test all test to give result	Frustrated

2.2 EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges. Most prior studies focused on developing models for the severity or mortality prediction of COVID-19 patients. However, effective models for recovery-time prediction are still lacking. Here, we present a deep learning solution named iCOVID that can successfully predict the recovery-time of COVID-19 patients based on predefined treatment schemes and heterogeneous multimodal patient information collected within 48 hours after admission. Meanwhile, an interpretable mechanism termed FSR is integrated into iCOVID to reveal the features greatly affecting the prediction of each patient. Data from a total of 3008 patients were collected from three hospitals in Wuhan, China, for large-scale verification. The experiments demonstrate that iCOVID can achieve a time-dependent concordance index of 74.9% (95% CI: 73.6-76.3%) and an average day error of 4.4 days (95% CI: 4.2-4.6 days). Our study reveals that treatment schemes, age, symptoms, comorbidities, and biomarkers are highly related to recovery-time predictions.

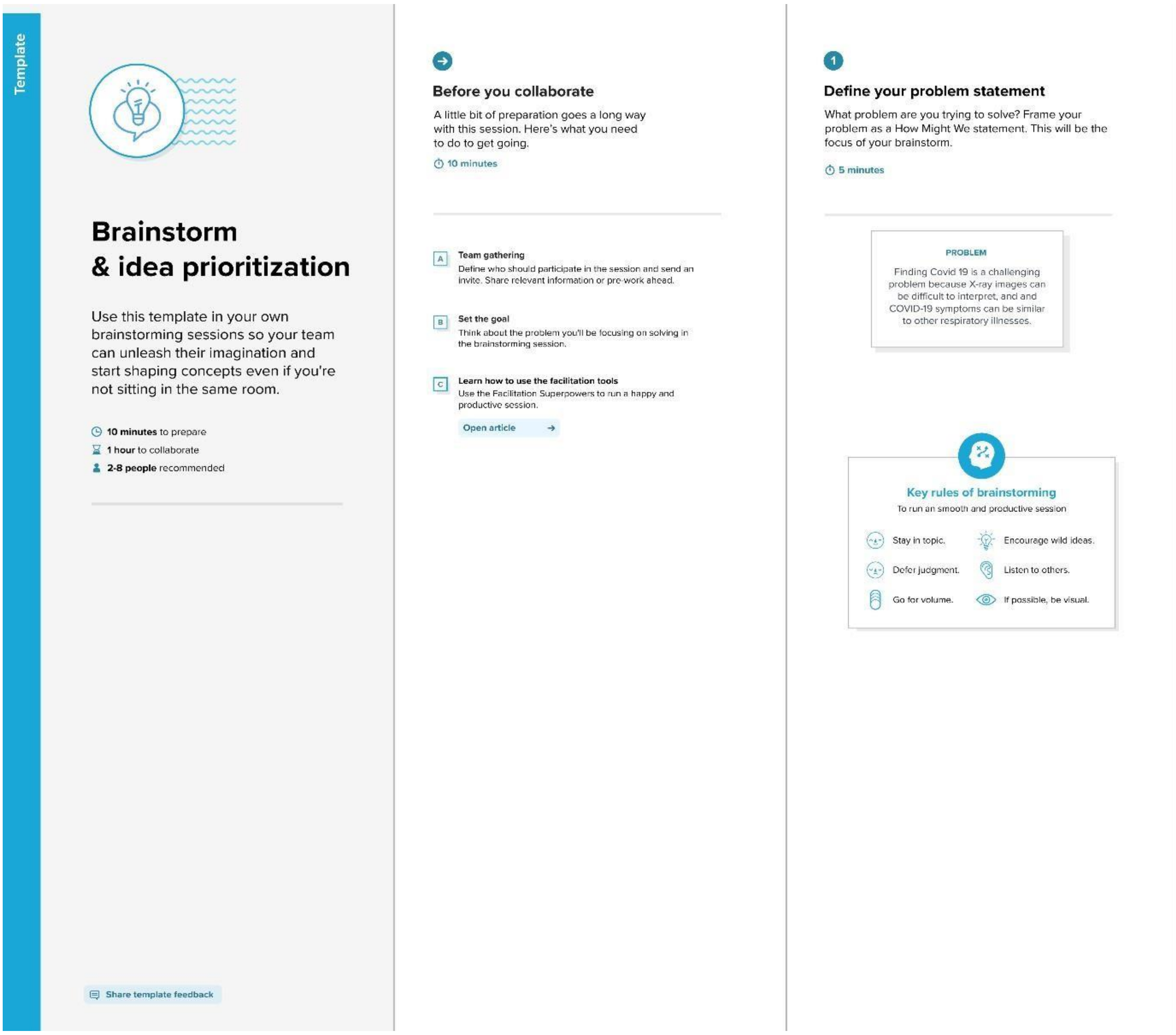


2.3 IDEATION AND BRAINSTORMING

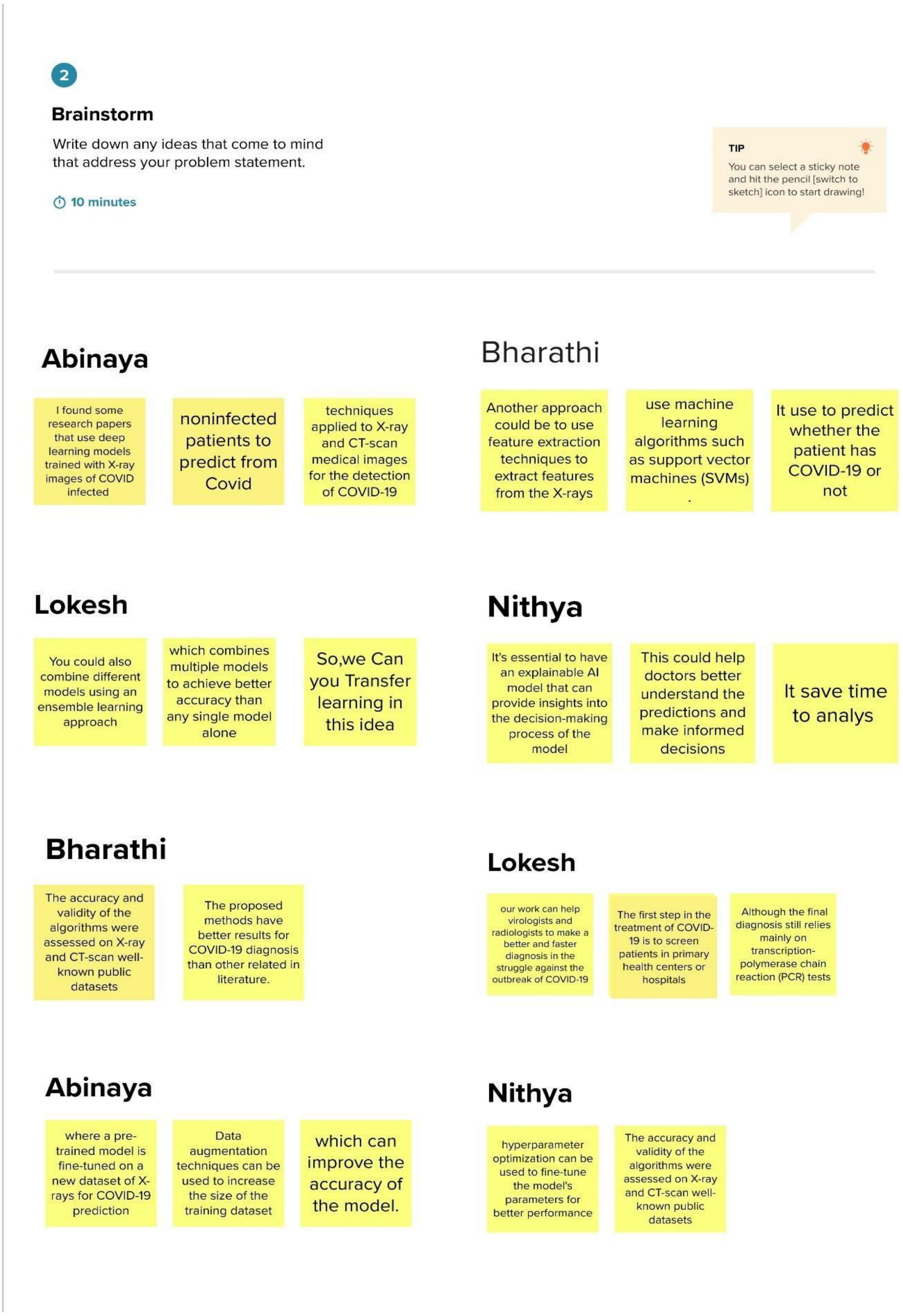
2.3.1 BRAINSTORM & IDEA PRIORITIZATION TEMPLATE

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 volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

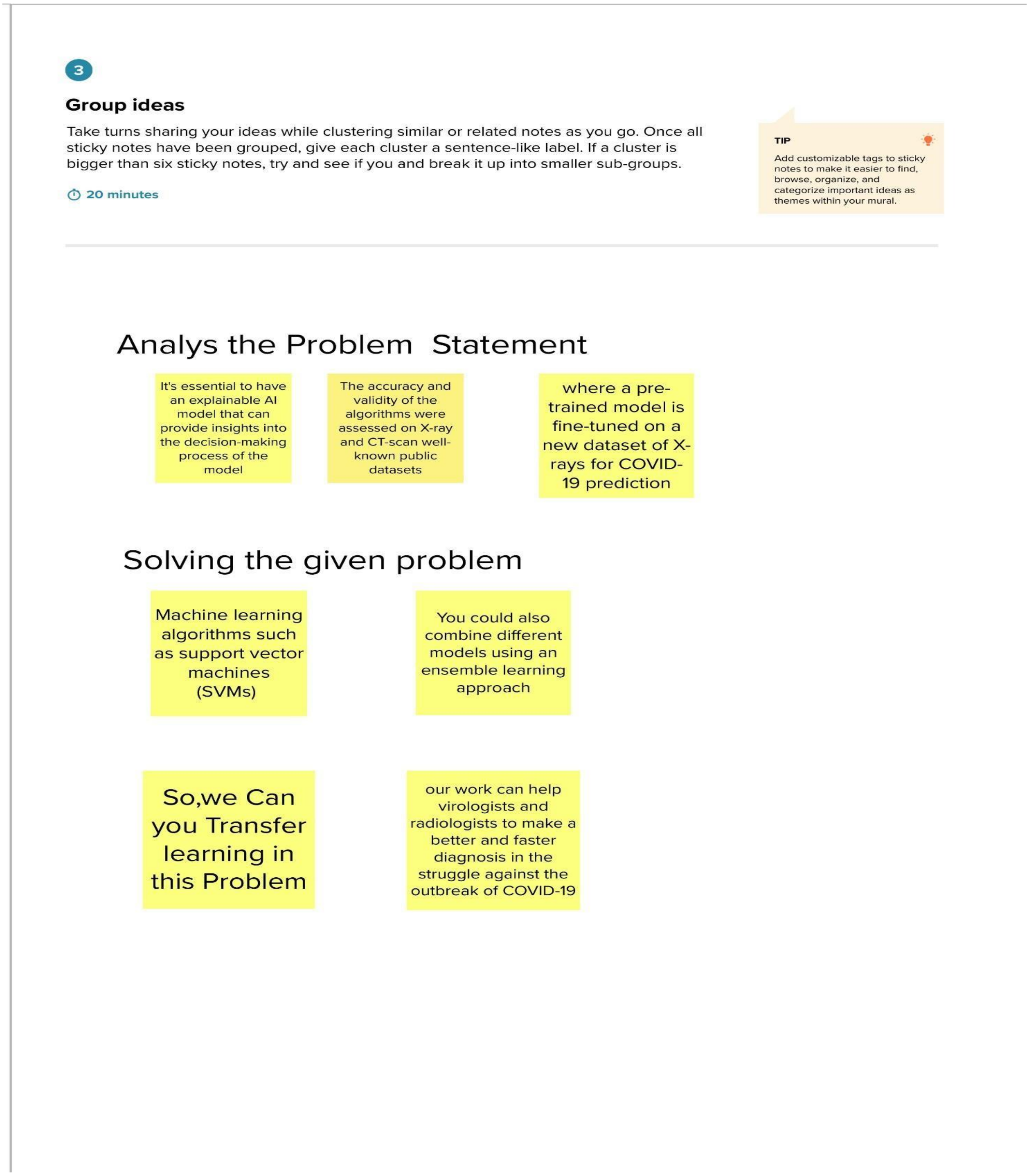
STEP-1: TEAM GATHERING, COLLABORATION AND SELECT THE PROBLEM STATEMENT



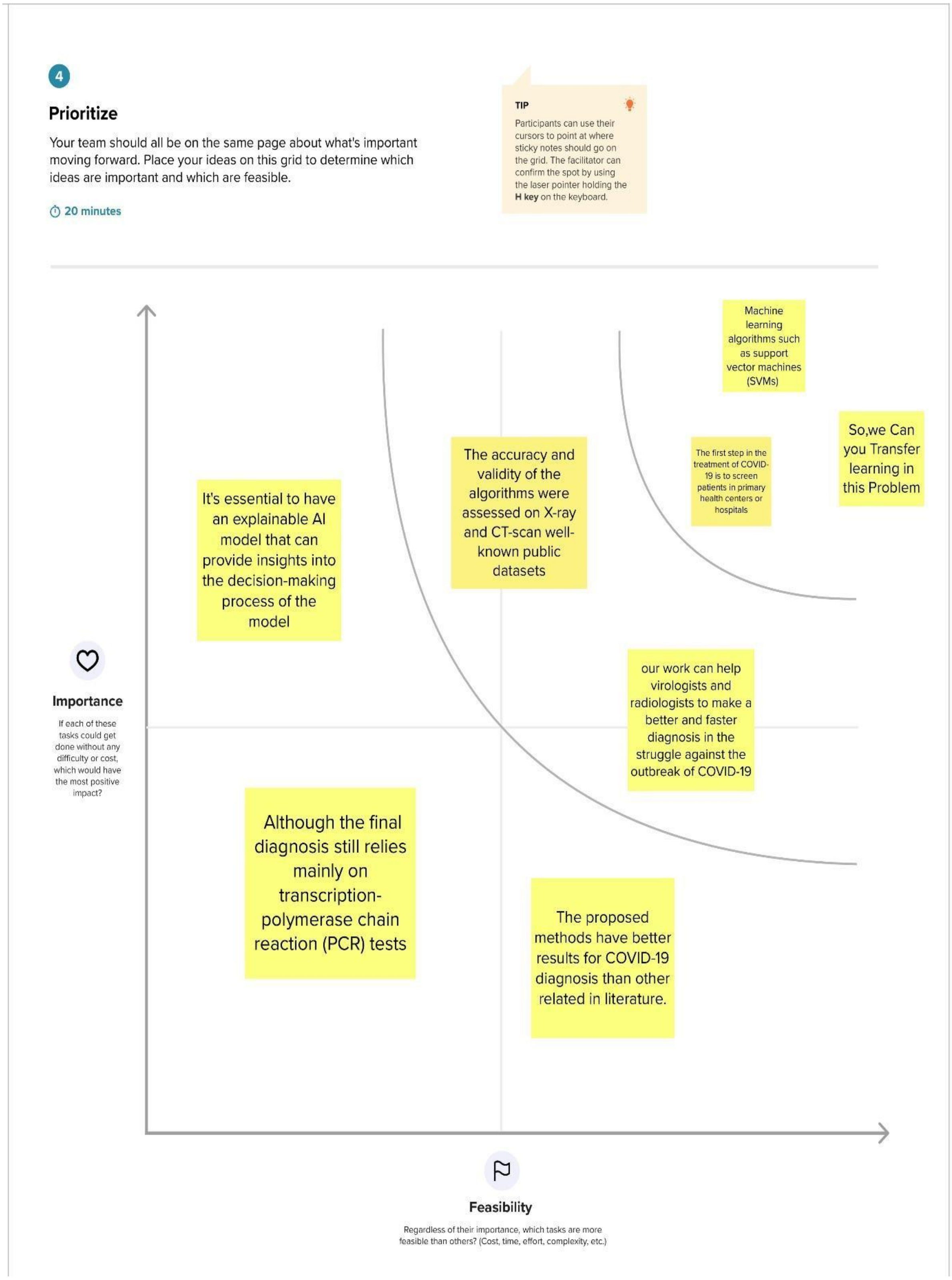
STEP-2: BRAINSTORM, IDEA LISTING



STEP-3: BRAINSTORM IDEAS GROUPING



STEP-4: IDEA PRIORITIZATION



2.4 PROPOSED SOLUTION

The proposed solution is to develop a deep learning-based system for COVID-19 detection from X-ray images. The system will leverage the power of deep learning algorithms to analyze X-ray images and accurately identify potential COVID-19 cases. Develop a deep learning model specifically designed for COVID-19 detection from X-ray images. The model will be trained on a diverse dataset of labeled X-ray images, including both COVID-19 positive and negative cases. The model architecture will be optimized to effectively capture patterns and features indicative of COVID-19.Develop a user-friendly interface or application that allows healthcare professionals to upload X-ray images and obtain COVID-19 detection results generated by the trained model. The interface should provide clear and interpretable results to aid in decision-making.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

1. User Registration and Authentication: Users should be able to register and create accounts. Implement a secure authentication mechanism to verify user identity.

2. Upload and Processing of X-ray Images: Allow users to upload X-ray images for COVID-19 detection. Validate and preprocess the uploaded images for analysis.

3. Deep Learning Model Integration: Integrate the trained deep learning model for COVID-19 detection. Apply the model to the uploaded X-ray images to generate predictions.

4. Result Display: Display the COVID-19 detection results to the users. Clearly indicate whether the image indicates a COVID-19 positive or negative case. Provide additional information, such as probability scores or confidence levels.

5. System Administration: Implement an administration interface to manage user accounts and access privileges. Allow administrators to monitor system performance and handle any technical issues.

3.2 NON-FUNCTIONAL REQUIREMENTS

1. Performance:The system should provide fast and responsive performance to ensure quick detection results.The response time for uploading images and generating predictions should be minimal.

2. Accuracy:The system should strive for high accuracy in COVID-19 detection from X-ray images.The deep learning model should be trained and optimized to achieve reliable and precise results.
3. Scalability:The system should be designed to handle an increasing number of users and image uploads.It should be able to scale seamlessly to accommodate growing demand without compromising performance.

4. Reliability:The system should be robust and reliable, minimizing downtime and ensuring continuous availability.It should have mechanisms in place to handle unexpected errors or failures gracefully.

5. Security:The system should implement strong security measures to protect user data and ensure privacy.User authentication, data encryption, and secure communication protocols should be employed.

6. Usability:The user interface should be intuitive, user-friendly, and easy to navigate.Users should be able to upload images and understand the detection results without technical expertise.

7. Accessibility:The system should be designed to be accessible to individuals with disabilities.Compliance with accessibility guidelines and standards should be ensured.

8. Compatibility: The system should be compatible with different web browsers, operating systems, and devices.It should be responsive and adaptable to various screen sizes and resolutions.

9. Compliance:The system should comply with relevant regulations and guidelines for medical data handling, such as HIPAA.It should adhere to ethical considerations and best practices in AI and healthcare.

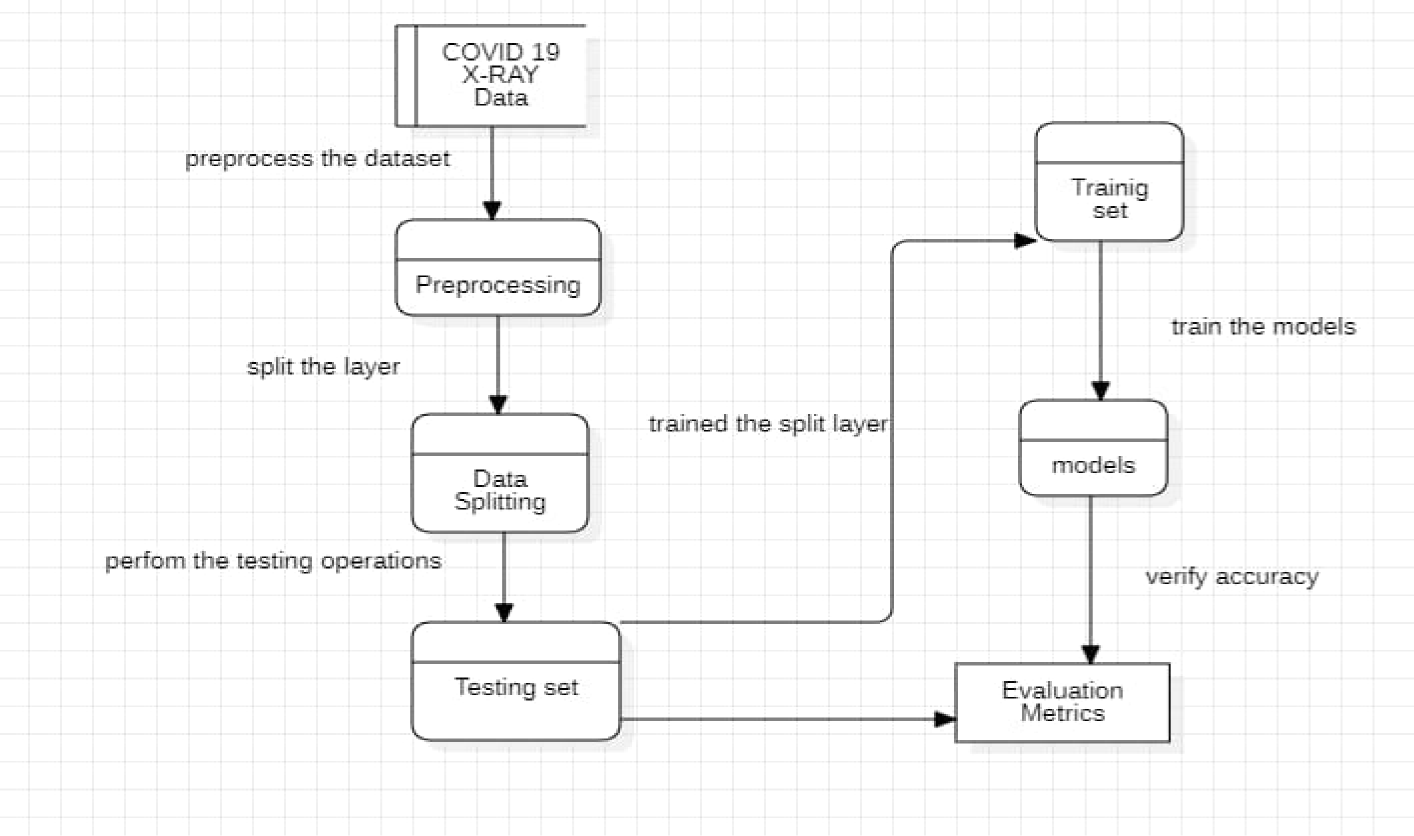
10. Maintainability: The system should be designed and documented in a way that facilitates easy maintenance and updates. Code modularity, version control, and clear documentation should be emphasized.

CHAPTER 4

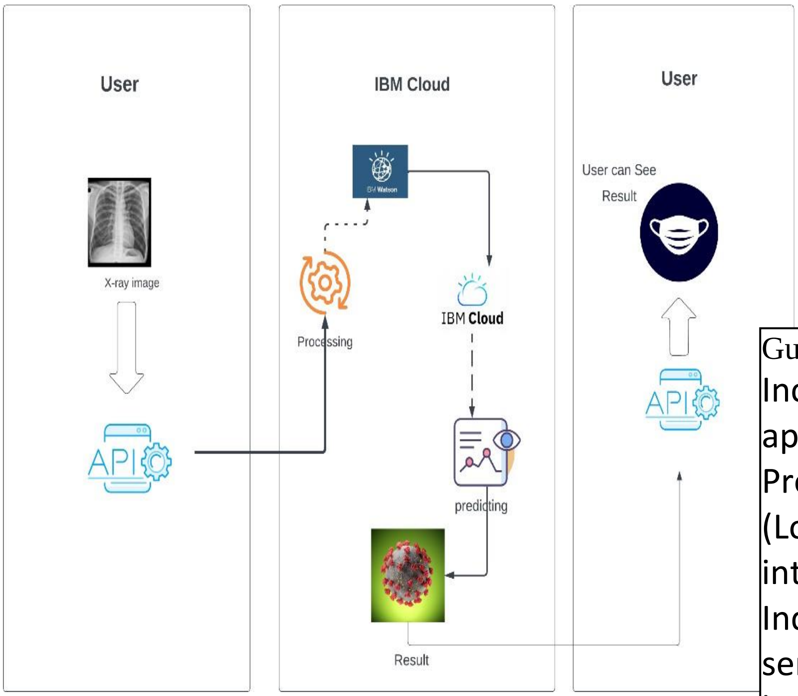
PROJECT DESIGN

4.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM).



4.2 SOLUTION AND TECHNICAL ARCHITECTURE



Guidelines:
Include all the processes (Asan application logic / Technology Block) Provide infrastructural demarcation (Local / Cloud) Indicate external interfaces (third party API's etc.) Indicate Data Storage components / services Indicate interface to machine learning models (if applicable)

4.3 USER STORIES

4.3.1 COMPONENTS & TECHNOLOGIES

S.No	Component	Description	Technology
1.	User Interface	Web UI, Mobile App.	HTML, CSS, JavaScript
2.	Application Logic-1	User can only Load the x-ray scan	image
3.	Application Logic-2	It will process the image	IBM Watson STT service
4.	Application Logic-3	It's give result to the x-ray scan	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	IBM Cloud

6.	Cloud Database	Database Service on Cloud	IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem

CHAPTER 5

CODING AND SOLUTION

5.1 FEATURE-1

AUTOMATED X-RAY IMAGE PREPROCESSING

Description: This feature focuses on implementing automated preprocessing techniques for X-ray images before they are fed into the deep learning model for COVID-19 detection. Preprocessing aims to enhance the quality and consistency of the images, ensuring optimal performance of the detection system.

5.1.1 STEPS FOR IMPLEMENTATION:

1. Image Resizing: Develop a function to resize the uploaded X-ray images to a standard resolution suitable for analysis. This step ensures that all images have consistent dimensions, which is crucial for the deep learning model.

2. Noise Reduction: Implement noise reduction techniques, such as Gaussian blurring or median filtering, to reduce noise or artifacts present in the images. This step improves image clarity and reduces the impact of irrelevant details during analysis.

3. Image Normalization: Apply image normalization methods, such as contrast stretching or histogram equalization, to enhance the image's contrast and improve the visibility of relevant features. Normalization helps to standardize the image intensities across different samples.

4. Region of Interest Extraction: Develop algorithms to automatically identify and extract the lung region from the X-ray images. This step helps in focusing the analysis on the relevant area and removing potential distractions or irrelevant information.

5. Data Augmentation: Implement data augmentation techniques, such as rotation,flipping, or random cropping, to increase the diversity of the dataset.

6. Augmentation helps in training the deep learning model to be more robust and generalize better to unseen X-ray images.

5.1.2 BENEFITS:

- Consistent and standardized X-ray images: The automated preprocessing ensures that all input images have consistent dimensions and improved quality, reducing variability and enhancing the accuracy of the subsequent detection process.
- Noise reduction and contrast enhancement: Preprocessing techniques enhance the visibility of important features in X-ray images, enabling the deep learning model to extract meaningful patterns associated with COVID-19.
- Time-saving and efficiency: Automating the preprocessing steps reduces the manual effort required for image preparation, allowing for a faster and more streamlined COVID-19 detection process.

5.1.3 CODING

Load the image

```
from tensorflow.keras.preprocessing import image
import numpy as np

image_path = 'COVID-46.png'
img = image.load_img(image_path, target_size=(224, 224))
img_array = image.img_to_array(img)
# Expand the dimensions to match the expected input shape for Xception (4 dimen-
sions)
img_array = np.expand_dims(img_array, axis=0)
```

Preprocess the image

```
img_array = preprocess_input(img_array)
images_data = tf.image.resize(img_array, [224, 224])
```

5.2 FEATURE-2

DEEP LEARNING MODEL TRAINING AND EVALUATION

Description: This feature focuses on training and evaluating the deep learning model for COVID-19 detection from X-ray images. The model will be trained on a labeled dataset and evaluated to assess its performance and accuracy in detecting COVID-19 cases.

5.2.1 STEPS FOR IMPLEMENTATION:

1. Dataset Preparation: Prepare a labeled dataset of X-ray images, including COVID-19 positive and negative cases. Ensure that the dataset is diverse, representative, and properly balanced to avoid bias.
2. Model Architecture Selection: Choose an appropriate deep learning architecture for COVID-19 detection, such as Convolutional Neural Networks (CNNs). Consider popular architectures like ResNet, VGGNet, or DenseNet, or customize the architecture based on the specific requirements.
3. Model Training: Implement the deep learning model using a suitable framework like TensorFlow or PyTorch. Train the model using the prepared dataset, employing techniques such as mini-batch training and backpropagation to optimize the model's parameters.
4. Hyperparameter Tuning: Fine-tune the model's hyperparameters, such as learning rate, batch size, and regularization techniques, to optimize its performance. Utilize techniques like cross-validation or grid search to identify the best combination of hyperparameters.
5. Model Evaluation: Evaluate the trained model using evaluation metrics like accuracy, precision, recall, and F1-score. Split the dataset into training, validation, and test sets to assess the model's generalization and performance on unseen data.
6. Performance Analysis: Analyze the model's performance using techniques like confusion matrix and ROC curves. Identify any potential biases or limitations of the model and adjust the training process or dataset accordingly.

5.2.2 BENEFITS:

- Accurate COVID-19 Detection: The deep learning model, trained on a diverse dataset, has the potential to accurately detect COVID-19 cases from X-ray images, aiding in early identification and timely intervention.
- Performance Optimization: Through hyperparameter tuning and performance analysis, the model's accuracy and robustness can be improved, ensuring reliable detection results.
- Generalization Ability: Evaluating the model on unseen data helps assess its generalization ability and provides insights into its performance in real-world scenarios.
- Model Iteration and Improvement: The evaluation results can guide further iterations and improvements in the model, enhancing its accuracy and addressing any limitations.

5.2.3 CODING

5.3 DATABASE SCHEMA

1. Users:

    i. user_id (Primary Key)
    ii. username
    iii. password
    iv. email
    v. role (e.g., admin, healthcare professional, patient)

2. XRayImages:

    i. image_id (Primary Key)
    ii. user_id (Foreign Key referencing Users table)
    iii. image_path
    iv. upload_date

3. Results:

    i. result_id (Primary Key)
    ii. image_id (Foreign Key referencing XRayImages table)
    iii. prediction (e.g., COVID-19 positive, COVID-19 negative)
    iv. confidence_score
    v. detection_date

CHAPTER 6
RESULTS

6.1 PERFORMANCE METRICS

Performance metrics for the COVID-19 detection system can be categorized into two main areas: model evaluation metrics and system performance metrics. Here are some commonly used metrics in each category:

6.1.1 MODEL EVALUATION METRICS:

1. Accuracy: The percentage of correctly predicted COVID-19 cases out of the total number of cases. It provides an overall measure of the model's correctness.
2. Precision: The proportion of true positive predictions (COVID-19 positive) out of all positive predictions. It measures the model's ability to correctly identify COVID-19 cases.
3. Recall (Sensitivity or True Positive Rate): The proportion of true positive predictions out of all actual positive cases. It measures the model's ability to identify all COVID-19 cases, minimizing false negatives.
4. Specificity (True Negative Rate): The proportion of true negative predictions (COVID-19 negative) out of all actual negative cases. It measures the model's ability to correctly identify non-COVID-19 cases.
5. F1-Score: The harmonic mean of precision and recall, providing a balanced measure of the model's performance. It is useful when dealing with imbalanced datasets.

6.1.2 SYSTEM PERFORMANCE METRICS:

1. Response Time: The time taken by the system to process an uploaded X-ray image and generate the COVID-19 detection result. It measures the system's speed and responsiveness.
2. Throughput: The number of X-ray images processed by the system within a given time period. It represents the system's capacity to handle a certain workload.
3. Scalability: The ability of the system to handle an increasing number of concurrent users and image uploads without significant degradation in performance.

4. Availability: The percentage of time the system is operational and accessible to users. It measures the system's reliability and uptime.

5. Error Rate: The percentage of erroneous COVID-19 detection results generated by the system. It reflects the accuracy and reliability of the system's predictions.

CHAPTER 7

ADVANTAGES AND DISADVANTAGES

7.1 ADVANTAGES

Advantages of using deep learning for COVID-19 detection from X-ray images:

1. Accurate Detection: Deep learning models have shown promising results in accurately identifying COVID-19 cases from X-ray images, potentially aiding in early diagnosis and treatment.
2. Efficient Screening: Automated deep learning systems can analyze large volumes of X-ray images quickly, allowing for efficient screening of suspected COVID-19 cases.
3. Objective Analysis: Deep learning models provide an objective analysis of X-ray images, reducing the potential for human bias or error in interpretation.
4. Scalability: Once trained, deep learning models can be scaled to handle a large number of X-ray images, making them suitable for high-throughput screening in busy healthcare settings.
5. Continuous Improvement: Deep learning models can be continually trained and improved using new data, leading to enhanced performance over time.

7.2 DISADVANTAGES

Disadvantages and challenges of using deep learning for COVID-19 detection:

1. Data Limitations: Deep learning models require large amounts of labeled training data, which may be challenging to obtain, especially for rare cases or specific patient populations.
2. Generalization to New Cases: Deep learning models may struggle to generalize well to cases that differ significantly from the training data, potentially leading to false positives or false negatives.

3. Interpretability: Deep learning models are often considered black boxes, making it challenging to interpret and explain the reasoning behind their predictions. This lack of interpretability can be a concern in medical settings where explainability is crucial.

4. Ethical Considerations: The deployment of deep learning models for COVID-19 detection raises ethical concerns related to patient privacy, data security, and the potential for biased or discriminatory outcomes.

5. Validation and Regulation: Deep learning models need rigorous validation and regulatory oversight to ensure their safety, efficacy, and compliance with medical standards.

CHAPTER 8
CONCLUSION

In conclusion, the application of deep learning for COVID-19 detection from X-ray images offers several advantages and opportunities in the field of medical diagnosis. The accuracy and efficiency of deep learning models enable automated screening and identification of COVID-19 cases, potentially aiding in early detection and timely intervention. The scalability of these models allows for processing large volumes of X-ray images, making them suitable for high-throughput screening in healthcare settings.

However, there are challenges and limitations that need to be addressed. Obtaining labeled training data can be challenging, and the generalization of deep learning models to new cases remains a concern. The lack of interpretability of deep learning models raises questions about their transparency and trustworthiness. Ethical considerations, such as patient privacy and data security, must be carefully addressed. Additionally, rigorous validation and regulatory oversight are necessary to ensure the safety, efficacy, and compliance of these models in medical applications.

Despite these challenges, deep learning holds great promise in the field of COVID-19 detection. Ongoing research and advancements in deep learning techniques, along with collaborations between the medical and AI communities, can help address the limitations and drive the development of accurate, reliable, and interpretable models. With further improvements, deep learning has the potential to play a significant role in the fight against COVID-19 by aiding healthcare professionals in making informed decisions and improving patient outcomes.

CHAPTER 9
FUTURE SCOPE

The future scope of using deep learning for COVID-19 detection from X-ray images is promising and presents several potential avenues for further research and development. Here are some future directions:

1. Improvement of Model Performance: Future research can focus on refining deep learning models to achieve even higher accuracy and reliability in COVID-19 detection. This can involve exploring advanced architectures, such as attention mechanisms or transformer-based models, and incorporating additional data modalities, such as clinical information or patient history, to enhance the predictive power of the models.

2. Transfer Learning and Pretraining: Transfer learning techniques can be leveraged to adapt pretrained models from related domains, such as general medical imaging or lung disease detection, to COVID-19 detection. This approach can help overcome data limitations and improve the generalization capabilities of the models.

3. Explainable AI for Interpretability: Addressing the lack of interpretability in deep learning models is crucial. Future research can focus on developing explainable AI techniques to provide insights into the model's decision-making process. This will enhance trust, facilitate clinical acceptance, and enable better integration of deep learning models into the healthcare workflow.

4. Real-Time Detection: Developing real-time deep learning models that can provide immediate feedback on COVID-19 detection from X-ray images can greatly assist healthcare professionals in making prompt clinical decisions. This requires optimizing model architectures and deployment strategies to achieve high-speed processing without compromising accuracy.

5. Large-Scale Collaborative Datasets: Building and sharing large-scale, diverse, and annotated datasets specifically for COVID-19 detection will be invaluable for training robust deep learning models. Collaborative efforts involving healthcare institutions, researchers, and data scientists can facilitate the creation of comprehensive datasets that capture the wide range of COVID-19 manifestations across different populations.

6. Integration with Clinical Decision Support Systems: Integrating deep learning models into clinical decision support systems can augment the capabilities of healthcare professionals by providing them with additional insights and recommendations based on the model's predictions. This integration can assist in more accurate diagnosis and treatment planning.

7. Adaptation to New Variants and Diseases: As new variants of COVID-19 emerge and other respiratory diseases manifest, deep learning models can be adapted and trained to detect these variations. This flexibility can enhance the system's utility in diagnosing and monitoring emerging infectious diseases.

CHAPTER 10

APPENDIX

APPENDIX-1

SOURCE CODE:

```
!pip install -q kaggle

import os
import shutil
import zipfile
import random
import csv
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from keras.models import load_model
from sklearn import preprocessing
from tensorflow.keras.preprocessing import image
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tqdm import tqdm
import matplotlib.pyplot as plt
from PIL import Image
import cv2
from tensorflow.keras.applications.xception import preprocess_input
# Create a directory named .kaggle in the home directory
os.makedirs(os.path.expanduser('~/.kaggle'), exist_ok=True)
# Copy the kaggle.json file to the .kaggle directory in the home directory
shutil.copyfile('kaggle.json', os.path.expanduser('~/.kaggle/kaggle.json'))

!kaggle datasets download -d tawsifurrahman/covid19-radiography-database
# Specify the path of the zip file
zip_path = 'covid19_radiography_database.zip'

# Specify the directory where you want to extract the files
extract_path = 'covid19_radiography_database'
```

```
# Open the zip file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    # Extract all the files to the specified directory
    zip_ref.extractall(extract_path)
dataset_path = 
"C:/Users/LOKESH/Advange_Covid_19_prediction/covid19_radiography_database/
covid19_radiography_database/COVID_19_Radiography_Dataset"
data = []
class_name = []
```

Displaying Information of Dataset

```
folders = ["Covid","Lung_Opacity", "Normal", "Viral Pneumonia"]
dataset_pathss = 
"C:/Users/LOKESH/Advange_Covid_19_prediction/covid19_radiography_database/
covid19_radiography_database/COVID_19_Radiography_Dataset"
dataset = []
header = ['Disease_list', 'Image_Count']
dataset.append(header)

for folder in folders:
    image_folder = f'animal/{folder}'
     #= os.path.join(dataset_paths, folder)
    folder_path = os.path.join(dataset_pathss, folder)
    images = os.listdir(folder_path)
    image_count = len(images)  # Count the number of images in the folder
    dataset.append([folder, image_count])

# Write the data to a CSV file
with open('image_counts.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(dataset)
dataset
 [['Disease_list', 'Image_Count'],
 ['Covid', 3616],
 ['Lung_Opacity', 6012],
 ['Normal', 10192],
 ['Viral Pneumonia', 1345]]
```

Graphical representation

```
# Separate the header and the data
header = dataset[0]
rows = dataset[1:]
```

```
# Extract the folder names and image counts
folders = [row[0] for row in rows]
image_counts = [int(row[1]) for row in rows]

# Plot the data as a bar graph
plt.bar(folders, image_counts)
#plt.xlabel(header[0])
plt.ylabel(header[1])
plt.title('Total Number of Images in Each Data')
#plt.xticks(rotation=90)

# Set the y-axis limit to 15000
plt.ylim(0,15000)

plt.show()
# Define the path to the dataset folder
dataset_paths =
"C:/Users/LOKESH/Advange_Covid_19_prediction/covid19_radiography_database/
covid19_radiography_database/COVID_19_Radiography_Dataset"

# Create a figure with 6 subplots
fig, axes = plt.subplots(1, 4, figsize=(10, 6))
axes = axes.flatten()

# Iterate over the folders and select a random image from each folder
folders = ["Covid","Lung_Opacity", "Normal", "Viral Pneumonia"]
for i, folder in enumerate(folders):
    folder_path = os.path.join(dataset_paths, folder)
    images = os.listdir(folder_path)


    # Select a random image from the folder
    random_image = random.choice(images)
    image_path = os.path.join(folder_path, random_image)
    image = plt.imread(image_path)


    # Plot the image in the subplot
    axes[i].imshow(image)
    axes[i].axis("on") # Show axis
    axes[i].set_title(folder)

# Adjust the layout and spacing
plt.tight_layout()
```

```
plt.show()
```

Data Augmentation

```
for covid in tqdm(os.listdir(dataset_path)):
    for i in range(len(os.listdir(dataset_path+ '/' + covid))):
        if i < 40:
            img = cv2.imread(dataset_path + '/' + covid + '/' + os.listdir(dataset_path+ '/' +
covid)[i])
            resized_img = cv2.resize(img,(224,224))
            resized_img = resized_img / 255.0
            data.append(resized_img)
            class_name.append(covid)

data = np.array(data,dtype = 'float32')
le = preprocessing.LabelEncoder()
le.fit(class_name)
class_names = le.classes_
class_name = le.transform(class_name)

class_name = np.array(class_name, dtype = 'uint8')
class_name = np.resize(class_name, (len(class_name),1))
```

Split the dataset into train and testing part

```
train_images, test_images, train_labels, test_labels = train_test_split(
    data, class_name, test_size=0.25, stratify = class_name
)
input_shape = (224, 224, 3)
```

Xception Model

```
from keras.applications.xception import Xception
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool-
ing2D,GlobalAveragePooling2D, Flatten
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array

xception_model = Sequential()
```

```
xception_base_model = Xception(include_top = False, weights = "imagenet", in-
put_shape = (224, 224, 3))
print(f'Number of layers in Xception model : {len(xception_base_model.layers)}')
Number of layers in Xception model : 132



for layer in xception_base_model.layers[:]:
    layer.trainable = False

for layer in xception_base_model.layers[4:]:
    layer.trainable = True

 # adding AveragePooling2D and Output layer
xception_model.add(xception_base_model)
xception_model.add(GlobalAveragePooling2D())
xception_model.add(Dense(units = 4, activation = 'softmax'))



xception_model.summary()
# callback processing statement
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping( monitor = 'val_accuracy',
                              mode = 'max', min_delta = 1e-5, patience = 5, re-
store_best_weights = True, verbose = 0)

# Compile process
xception_model.compile(optimizer = keras.optimizers.RMSprop(learning_rate =
0.0001) , loss = 'sparse_categorical_crossentropy' , metrics = ['accuracy'])

pwd


# fit the model
xception = xception_model.fit(train_images, train_labels, batch_size = 32, epochs
=50,
                             callbacks = [early_stopping], validation_split = 0.2)
xception_model.summary()
# Load the saved model
xception_model.save('xception_covid.h5')


loaded_model = load_model('xception_covid.h5')
print("Ya,It nearly over")
```

```
loaded_model.summary
```

<bound method Model.summary of <keras.engine.sequential.Sequential object at 0x000002E8217AD6A0>>

```
import matplotlib.pyplot as plt
plt.figure(figsize = (30,10))

plt.subplot(2,2,1)
i_train_acc = xception.history['accuracy']
i_val_acc = xception.history['val_accuracy']
i_epoch = [i for i in range(len(i_val_acc))]
plt.plot(i_epoch , i_train_acc , 'go-' , label = 'Training Accuracy')
plt.plot(i_epoch, i_val_acc , 'ro-' , label = 'Validation Accuracy')
plt.title('Training & Validation Accuracy for Xception Net')
plt.legend()
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
```

Text(0, 0.5, 'Accuracy')

```
#Testing for test data
xception_predictions = xception_model.predict(test_images)
xception_predictions = np.argmax(xception_predictions,axis = 1)
#Testing for test data
randomImage = np.random.randint(0,np.shape(test_images)[0])

plt.imshow(test_images[randomImage])
plt.title("Xception Model's Predicted Covid : " +
str(le.inverse_transform([xception_predictions[randomImage]])))
```

Text(0.5, 1.0, "Xception Model's Predicted Covid : ['Viral Pneumonia']")

Testing part-1

```
# Load the image
from tensorflow.keras.preprocessing import image
import numpy as np

image_path = 'COVID-46.png'
img = image.load_img(image_path, target_size=(224, 224))
img_array = image.img_to_array(img)
# Expand the dimensions to match the expected input shape for Xception (4 dimensions)
img_array = np.expand_dims(img_array, axis=0)

# Preprocess the image
```

```
img_array = preprocess_input(img_array)
images_data = tf.image.resize(img_array, [224, 224])
# Make predictions on the image
prediction = loaded_model.predict(images_data)
plt.imshow(img)
r=np.argmax(prediction)
plt.title(f'Prediction: {class_names[r]}')
plt.show()
```

Testing part-2

```
# Load the image

image_path = 'Viral Pneumonia-5.png'
img = image.load_img(image_path, target_size=(224, 224))
img_array = image.img_to_array(img)
# Expand the dimensions to match the expected input shape for Xception (4 dimen-
sions)
img_array = np.expand_dims(img_array, axis=0)

# Preprocess the image

img_array = preprocess_input(img_array)
images_data = tf.image.resize(img_array, [224, 224])
# Make predictions on the image
prediction = loaded_model.predict(images_data)
plt.imshow(img)
r=np.argmax(prediction)
plt.title(f'Prediction: {class_names[r]}')
plt.show()
```

Testing part-3

```
# Load the image

image_path = 'Normal-30.png'
img = image.load_img(image_path, target_size=(224, 224))
img_array = image.img_to_array(img)
# Expand the dimensions to match the expected input shape for Xception (4 dimen-
sions)
img_array = np.expand_dims(img_array, axis=0)

img_array = preprocess_input(img_array)
images_data = tf.image.resize(img_array, [224, 224])
# Make predictions on the image
```

```
prediction = loaded_model.predict(images_data)
1/1 [==============================] - 0s 101ms/step
plt.imshow(img)
r=np.argmax(prediction)
plt.title(f'Prediction: {class_names[r]}')
plt.show()
```

PYTHON CODE:

```
from keras.applications.xception import
preprocess_input
from keras.preprocessing import image
import numpy as np
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
app=Flask(__name__)

model=load_model(r"xception_covid.h5",compile=False)
@app.route('/')
def index():
   return render_template("index.html")
@app.route('/precaution/')
def next1():
   return render_template("precaution.html")
@app.route('/test/')
def next2():
   return render_template("test.html")

@app.route('/predict',methods=['GET','POST'])
def upload():
   if request.method=='POST':
      f=request.files['image']
      basepath=os.path.dirname(__file__)
      print("Hello")
      filepath=os.path.join(basepath,'uploads',f.filename)
      f.save(filepath)
      img=image.load_img(filepath,target_size=(224,224))
      x=image.img_to_array(img)
      x=np.expand_dims(x,axis=0)

  img_data=preprocess_input(x)
  pred=np.argmax(model.predict(img_data),axis=1)

      print(pred)
      index=['COVID','Lung_Opacity','Normal','Viral Pneumonia']
      text="The result of x-ray : " +str(index[pred[0]])
```

```
    return text

if __name__=='__main__':
    app.run(debug=True)
```

APPENDIX-2

LINKS:

Github :
https://github.com/naanmudhalvan-SI/PBL-NT-GP--1701-1680520245/tree/main

Video_link :
https://drive.google.com/file/d/15cQ7LG844fEsbsBxLIzLb4JdqZsayWOm/view?usp=share_link