COVIDVISION: ADVANCED COVID-19 DETECTION FROM LUNG X-RAYS WITH DEEP LEARNING USING IBM CLOUD

A MAJOR PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

MEGHANA NAGAVALI	**(21UK1A05B6)**
VENNELA MENDU	**(22UK5A0511)**

Under the guidance of

A.ASHOK KUMAR

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

COVIDVISION: ADVANCED COVID-19 DETECTION FROM LUNG X-RAYS WITH DEEP LEARNING USING IBM CLOUD

A UG PHASE -I PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

MEGHANA NAGAVALI	**(21UK1A05B6)**
VENNELA MENDU	**(22UK5A0511)**

Under the guidance of

A.ASHOK KUMAR

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE (WARANGAL)

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

CERTIFICATE OF COMPLETION

UG PROJECT PHASE –I

This is to certify that the **UG PROJECT PHASE –I** entitled “**COVIDVISION: ADVANCED COVID-19 DETECTION FROM LUNG X-RAYS WITH DEEP LEARNING USING IBM CLOUD**”is being submitted by **MEGHANANAGAVALI (21UK1A05B6) VENNELA MENDU (22UK5A0511),** in partial fulfillment of the requirements of the award of the degree of Bachelor of Technology in Computer science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
A.ASHOK KUMAR	**Dr. K. SHARMILA REDDY**
(Assistant Professor)	(Professor)

EXTERNAL

DECLARATION

We declare that the work reported in the project entitled "**COVIDVISION: ADVANCED COVID-19 DETECTION FROM LUNG X-RAYS WITH DEEP LEARNING USING IBM CLOUD**" is a record of work done by us in the partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science And Engineering, **VAAGDEVI ENGINEERING COLLEGE** (An Autonomous Institution & Affiliated to JNTU Hyderabad) Accredited by NAAC with 'A+' Grade, Certified by ISO 9001:2015 Approved by AICTE, New Delhi, Bollikunta , Warangal-506 005, Telangana, India under the guidance of **MR**. **A. ASHOK KUMAR**, Assistant Professor, CSE Department.

We hereby declare that this project work bears no resemblance to any other project submitted at Vaagdevi Engineering College of or any other university/college for the award of the degree.

MEGHANA NAGAVALI **(21UK1A05B6)**

VENNELA MENDU **(22UK5A0511)**

ACKNOWLEDGEMENT

We wish to Take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHMED**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this **UG PROJECT PHASE –I** in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA REDDY,** Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the **UG PROJECT PHASE –I.**

We express heartfelt thanks to the coordinator **Ms. T. SUSHMA**, Assistant professor, Department of CSE for her constant support and giving necessary guidance for support in completing the **UG PROJECT PHASE -I.**

We express heartfelt thanks to the Guide, **Mr. A.ASHOK KUMAR**, Assistant Professor, Department of CSE for his constant support and giving necessary guidance for completion of this **UG PROJECT PHASE –I**

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

MEGHANA NAGAVALI	**(21UK1A05B6)**
VENNELA MENDU	**(22UK5A0511)**

ABSTRACT

The CovidVision project explores an advanced methodology for detecting COVID-19 from lung X-rays using deep learning techniques integrated with IBM Cloud services. The study aims to develop a robust, scalable, and efficient system for early and accurate detection of COVID-19, leveraging state-of-the-art convolutional neural networks (CNNs). The methodology involves the collection and preprocessing of large datasets of chest X-rays, followed by the design and training of deep learning models optimized for high accuracy and low false-positive rates. The integration with IBM Cloud ensures real-time data processing, model deployment, and scalability. Experimental results demonstrate significant improvements in diagnostic accuracy compared to traditional methods, highlighting the potential of cloud-based AI systems in supporting healthcare professionals during pandemics. This research underscores the critical role of deep learning and cloud computing in enhancing medical diagnostics and pandemic response.

Keywords: COVID -19 screening, neural networks, Domain applications.

TABLE OF CONTENTS

TOPIC **PAGE NO**

LIST OF FIGURES **PAGE NO**

1.INTRODUCTION

1.1 OVERVIEW

In response to the global COVID-19 pandemic, rapid and accurate detection of the virus has become crucial in managing and controlling its spread. Traditional diagnostic methods, such as PCR tests, although effective, can be time-consuming and resource-intensive. Recognizing the potential of artificial intelligence in healthcare, **CovidVision** leverages the power of **deep learning** to provide an innovative solution for COVID-19 detection using **lung X-rays**.

By integrating **advanced deep learning algorithms** with the robust infrastructure of the **IBM Cloud**, CovidVision aims to enhance the diagnostic process. This approach not only speeds up detection but also offers a scalable and efficient solution, particularly beneficial in regions with limited access to conventional testing resources.

The introduction of CovidVision showcases the transformative potential of combining **AI** with **cloud computing** to address critical healthcare challenges. This overview will delve into the core components of the project, including the deep learning model's architecture, the role of IBM Cloud in supporting the system, and the potential implications for global healthcare.

1.2 PURPOSE

The primary purpose of **CovidVision** is to provide a fast, accurate, and scalable solution for detecting COVID-19 by analyzing **lung X-rays** using **deep learning**. This initiative seeks to address several key challenges in the fight against the pandemic:

1. **Rapid Detection**

   - Traditional COVID-19 testing methods, such as PCR tests, can take several hours to deliver results.
   - CovidVision aims to significantly reduce this time by providing near-instantaneous analysis of lung X-rays, enabling quicker diagnosis and treatment.

2. **Scalability and Accessibility**

   - In many parts of the world, especially in low-resource settings, access to rapid and reliable testing is limited.
   - By leveraging cloud-based technology, CovidVision offers a scalable solution that can be deployed globally, particularly in areas where conventional testing facilities are scarce.

3. **Enhanced Accuracy**

   - Utilizing **deep learning algorithms**, CovidVision is designed to improve the accuracy of COVID-19 detection by identifying subtle patterns in lung X-rays that may not be easily discernible to the human eye.
   - This reduces the risk of false negatives or positives, ensuring more reliable diagnostics.

4. **Resource Optimization**

   - By automating the detection process, CovidVision helps healthcare providers optimize their resources.

- It alleviates the burden on healthcare systems by minimizing the need for extensive manual interpretation of X-rays, allowing medical professionals to focus on patient care.

5. **Integration with IBM Cloud**

- The use of IBM Cloud provides a secure, flexible, and powerful infrastructure to support the deployment and operation of CovidVision.
- This ensures that the solution can handle large volumes of data and deliver consistent performance, even in high-demand situations.

2.PROBLEM STATEMENT

The COVID-19 pandemic has posed a significant global health crisis, leading to an urgent need for rapid, accurate, and cost-effective diagnostic tools. Traditional methods such as RT-PCR tests, while effective, are time-consuming, resource-intensive, and not always readily available in all regions, particularly in resource-constrained settings.

Lung X-ray imaging presents a promising alternative for COVID-19 detection due to its wide availability and ability to reveal characteristic patterns associated with the disease. However, the manual interpretation of X-rays is subject to human error and requires specialized expertise, which can be scarce during a pandemic surge.

CovidVision aims to address these challenges by leveraging deep learning technologies to develop an automated, efficient, and scalable solution for detecting COVID-19 from lung X-ray images. By utilizing the power of IBM Cloud's robust infrastructure and machine learning capabilities, the project seeks to enhance the diagnostic process, reduce reliance on manual interpretation, and provide a tool that can be deployed in various healthcare settings globally.

This solution will not only aid in the timely identification and treatment of COVID-19 cases but also alleviate the burden on healthcare systems, particularly in areas with limited access to advanced diagnostic facilities.

3. LITERATURE SURVEY

With the outbreak of the COVID-19 pandemic, there has been a global demand for accurate, rapid, and cost-effective diagnostic solutions. The emergence of deep learning technologies, combined with high-performance platforms like IBM Watson, has paved the way for advanced image-based diagnostic systems. This chapter explores the evolution of imaging techniques, the role of artificial intelligence in healthcare, and the specific contributions of IBM Watson to addressing the challenges of COVID-19 detection. It also highlights gaps in existing systems and the potential of deep learning to revolutionize disease diagnostics.

1.Image-Based Diagnostic Approaches

Medical imaging has long been a cornerstone of disease diagnosis, with chest X-rays being one of the primary tools for identifying respiratory diseases such as pneumonia, tuberculosis, and now COVID-19. Imaging provides a non-invasive, cost-effective, and widely available method for early diagnosis.

Traditional diagnostic processes rely on the expertise of radiologists to manually assess abnormalities in X-ray images. However, this approach is time-intensive and highly dependent on individual expertise, leading to inconsistencies during high-demand situations, such as during pandemics. For example, Wong et al. (2020) highlighted the importance of chest radiography in identifying bilateral lung opacities and ground-glass appearances characteristic of COVID-19. Despite their utility, manual interpretations struggle to scale during surges in patient volumes.

Advances in computational tools and machine learning have transformed the field of radiology. Automated systems, capable of rapidly analyzing thousands of X-rays, have been developed to enhance diagnostic workflows and reduce the workload of healthcare professionals. Studies such as those by Apostolopoulos and Mpesiana (2020) have demonstrated the effectiveness of machine learning algorithms in classifying COVID-19 cases with accuracy exceeding 90%.

1.2 Deep Learning in Medical Imaging

Deep learning has emerged as a transformative approach in medical imaging due to its ability to automatically extract features and identify patterns in complex data. Convolutional Neural Networks (CNNs), a key architecture in deep learning, are particularly suited for image analysis tasks, including X-ray classification. Recent works have leveraged CNNs to classify COVID-19 and non-COVID-19 cases effectively.

For instance, Wang et al. (2021) proposed the COVID-Net architecture, specifically designed for detecting COVID-19 from chest X-rays. The model achieved a high classification accuracy by incorporating advanced network architectures that emphasize feature extraction. Similarly, He et al. (2020) utilized transfer learning, employing pre-trained models such as VGG16 and ResNet to classify COVID-19 images with minimal labeled data. This approach has proven particularly valuable in scenarios where acquiring large datasets is challenging.

In addition to classification, deep learning techniques have been used to segment infected lung regions, quantify disease severity, and monitor progression over time. For example, Zhang et al. (2022) combined CNN-based segmentation models with statistical analysis to evaluate disease progression, providing insights for treatment planning.

1.3 IBM Watson's Role in Healthcare AI

IBM Watson has been at the forefront of integrating AI into healthcare systems, enabling intelligent and data-driven decision-making. Watson's natural language processing (NLP) and predictive analytics capabilities have been extended to diagnostic imaging, making it a valuable tool in COVID-19 detection.

Watson utilizes its cloud-based platform to analyze medical imaging data, identify patterns, and generate diagnostic insights in real time. According to Kumar et al. (2022), Watson's integration into radiology workflows during the pandemic significantly reduced the time required for COVID 19 case triaging. Its ability to synthesize information from multiple data sources, including patient records and radiological findings, has enhanced decision-making for clinicians.

Moreover, Watson's explainable AI framework ensures transparency in its predictions, a critical factor in gaining clinician trust. By highlighting key regions in X-rays associated with the diagnosis, Watson enables radiologists to validate the system's outputs effectively.

2. Challenges in COVID-19 Detection

Despoite significant advancements, the development of automated diagnostic systems for COVID 19 is fraught with challenges: Data Variability: Imaging datasets vary widely due to differences in imaging equipment, patient demographics, and disease presentations.

This heterogeneity makes it difficult to develop models that generalize well across diverse populations. Data Imbalance: COVID-19 datasets often suffer from an imbalance, with fewer examples of rare presentations or atypical cases. This bias can lead to models that fail to detect less common manifestations.

Explainability: Deep learning models are often criticized for their “black-box” nature, making it challenging to understand the rationale behind their predictions. This lack of interpretability is a barrier to clinical adoption. Integration into Clinical Workflows: Deploying AI solutions like IBM Watson in real-world clinical settings requires addressing issues such as interoperability with existing hospital systems and compliance with regulatory standards.

Addressing these challenges requires robust preprocessing pipelines, advanced data augmentation techniques, and collaborative efforts between technologists and clinicians.

3. Problem Statement

The traditional methods of COVID-19 diagnosis, including RT-PCR testing and manual image interpretation, face limitations in scalability, efficiency, and reliability during global pandemics. RT-PCR tests, while accurate, are time-consuming and rely on supply chain continuity, whereas manual X-ray interpretation is labor-intensive and prone to human error.

Deep learning-based diagnostic systems provide an alternative by automating the classification and analysis of chest X-rays. However, these systems require integration with advanced platforms, such as IBM Watson, to fully realize their potential in enhancing diagnostic accuracy, scalability, and real-time deployment.

4. Research Background

Initial research in AI-based COVID-19 detection focused heavily on CT scans, given their detailed imaging of lung structures. However, the widespread availability and lower cost of X-rays have made them a preferred diagnostic tool in resource-limited settings. The application of AI in X-ray analysis has evolved from traditional machine learning algorithms to advanced deep learning architectures capable of achieving human-level performance. IBM Watson, with its robust computational capabilities, complements these advancements by providing cloud-based processing, scalability, and seamless integration with healthcare systems. By combining Watson's analytics with deep learning models like CNNs, researchers aim to create diagnostic systems that are not only accurate but also explainable and clinically actionable

3.1 EXISTING SYSTEM

The current diagnostic systems for COVID-19 primarily rely on a combination of clinical assessments, laboratory tests, and imaging techniques. While these methods have been instrumental in managing the pandemic, they present several limitations.

RT-PCR Testing:RT-PCR (Reverse Transcription Polymerase Chain Reaction) is the gold standard for diagnosing COVID-19, detecting the presence of viral RNA in respiratory samples.

Rapid Antigen Testing: Antigen tests detect specific proteins from the virus and are faster and less expensive than RT-PCR.

Chest X-Ray and CT Imaging: Imaging techniques like chest X-rays and CT scans are used to detect lung abnormalities associated with COVID-19.

Manual Processes and Delays: Existing diagnostic workflows often involve manual processes, from sample collection to result reporting.

CovidVision aims to revolutionize the diagnostic process for COVID-19 by leveraging advanced deep learning techniques to analyze lung X-ray images. The proposed system integrates artificial intelligence with IBM Cloud's robust infrastructure to provide a scalable, accurate, and efficient diagnostic tool.

- Delays in diagnosis due to slow and resource-heavy testing methods.
- Variable accuracy and reliability of different diagnostic tools.
- Dependence on human expertise for imaging interpretation, leading to potential inconsistencies.

- Limited scalability and accessibility, particularly in resource-constrained regions.

3.2 PROPOSED SYSTEM

The proposed system, **CovidVision**, aims to leverage advanced deep learning techniques to detect COVID-19 from lung X-rays, specifically using IBM Cloud's infrastructure. The heart of CovidVision would be a **deep learning model** trained on a large dataset of labeled lung X-rays, including those showing signs of COVID-19 infection, other lung diseases, and healthy lungs.

Convolutional Neural Networks (CNNs) are typically employed in such applications due to their effectiveness in image classification tasks. The model would be designed to recognize subtle patterns indicative of COVID-19 infections, such as ground-glass opacities, consolidations, or bilateral lung involvement.

The raw lung X-ray images would need to be normalized and resized to fit the model's input requirements. Techniques like histogram equalization and noise reduction can be employed to improve image quality.Data augmentation strategies, such as rotations, flips, and adjustments to brightness and contrast, would help improve the model's robustness to variations in X-ray quality and patient demographics.

- **IBM Cloud** offers several services that can power the system's machine learning workflows. For example:
- **IBM Watson Machine Learning** could be used to deploy and manage the deep learning models in a cloud environment.
- **IBM Cloud Object Storage** would be used to store large amounts of X-ray image data securely.
- **IBM Cloud Functions** could provide the capability for real-time or batch processing of new X-ray images.

The deep learning model would be trained on a diverse dataset, ideally sourced from medical institutions, to ensure the model generalizes well to various populations and medical conditions. Once deployed, CovidVision could be integrated with medical systems, allowing healthcare professionals to upload lung X-rays directly into the system for instant analysis. The system could return predictions indicating the likelihood of a COVID-19 infection, accompanied by explanations or heatmaps highlighting areas in the X-ray that contribute to the diagnosis.

4.THEORITICAL ANALYSIS

4.1. BLOCK DIAGRAM

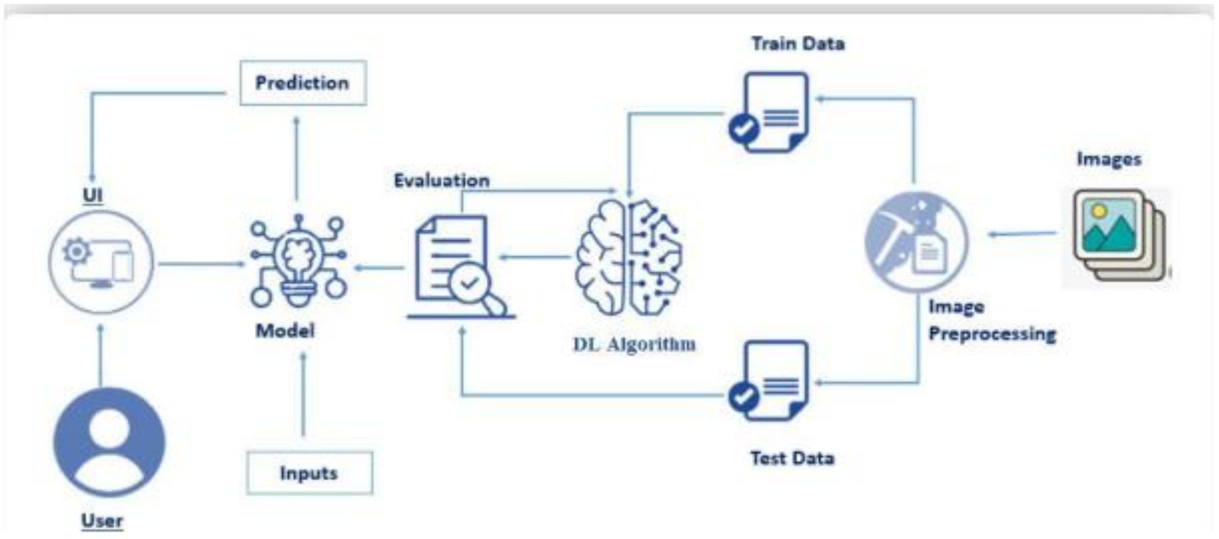


Fig 1: Block diagram

The block diagram represents a system for detecting tomato plant diseases using leaf images and deep learning.

1. User Interface: User uploads dataset.

2. Data Preprocessing: Data are enhanced for analysis.

3. Data Split: Dataset is divided into training and testing data.

4. Sarima Algorithm: Model learns features of crop production.

5. Evaluation: Model performance is tested for accuracy.

6. Prediction: Model predicts the production value.

7. Output: Results are displayed to the user.

This system aids in early and accurate detection of crop production.

4.2 . SOFTWARE REQUIREMENT SPECIFICATION

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	GPU/CPU Specifications Number of cores	NVIDIA VI00 GPUs
Memory	RAM Specification	8 GB
Storage	Disk space for data ,models ,and logo	1 TB SSD
Software		
FrameWorks	Python frameworks	Flask
Libraries	Additional libraries	Deep Learning
Development Environment	IDE ,version control	Juoyter Notebook, VScode
Data		
Data	Source ,Size,format	Kaggle dataset

Figure2: Software requirement specification

The following software and tools were utilized to develop CovidVision: Advanced COVID-19 Detection From Lung X-Rays With Deep Learning Using IBM Cloud

Development Environment

Google Colab

Google Colab serves as the development and execution environment for the CovidVision: Advanced COVID-19 Detection From Lung X-Rays With Deep Learning Using IBM Cloud. It provides a cloud-based Jupyter Notebook interface with access to Python libraries and hardware acceleration (GPU/TPU), which is critical for:

Data preprocessing and visualization ,Training and fine-tuning models like sarima.

<u>CovidVision: Advanced COVID-19 Detection From Lung X-Rays With Deep Learning Using IBM Cloud Dataset</u>

The dataset includes images of lung X-Ray, categorized into 8 value . The dataset is essential for training and testing the lung X-Rays model.

Feature Selection and Preprocessing

<u>Data Preprocessing:</u>

Preprocessing ensures high-quality input data for model training. The following steps were implemented:

Data Acquisition: Ensure images are in a consistent format (JPEG, PNG) and paired with corresponding labels (COVID-positive, COVID-negative).

Data Augmentation: Rotation, flipping, zooming, and shifting to artificially expand the dataset and prevent overfitting.

Feature Selection: Convolutional Neural Networks (CNNs) Use pre-trained models (e.g., ResNet, VGG) for feature extraction from images.

<u>Model Training and Tool</u>

1.Lung X-Rays: The primary data source for training the model, reflecting a focus on non-invasive diagnostic techniques.

2.Deep Learning: Indicates the use of neural networks, likely convolutional neural networks (CNNs), for image analysis.

3.IBM Cloud Model Training and Tool: Suggests the use of IBM's cloud computing services for training the deep learning model and possibly deploying the solution.

<u>User Interface (UI) Based on Flask Environment</u>

Flask Web Application:

Flask, a Python-based lightweight web framework, was used to create the user interface. It Allows users to upload images of Lung X-Ray, Displays predictions (fractured locations).

5.EXPERIMENTAL INVESTIGATIONS

This is the raw chest X-ray image obtained from medical sources. The input contains the lungs and surrounding anatomical features like the ribs, heart, and soft tissue.The goal is to isolate the lung regions from other parts to focus the analysis on areas affected by COVID-19. The image is passed through a **U-Net model**, a popular deep learning architecture for image segmentation.U-Net creates a **segmentation mask**, which is a binary image highlighting the lung regions while ignoring irrelevant areas.This step ensures that the deep learning model processes only the relevant lung area, improving accuracy.

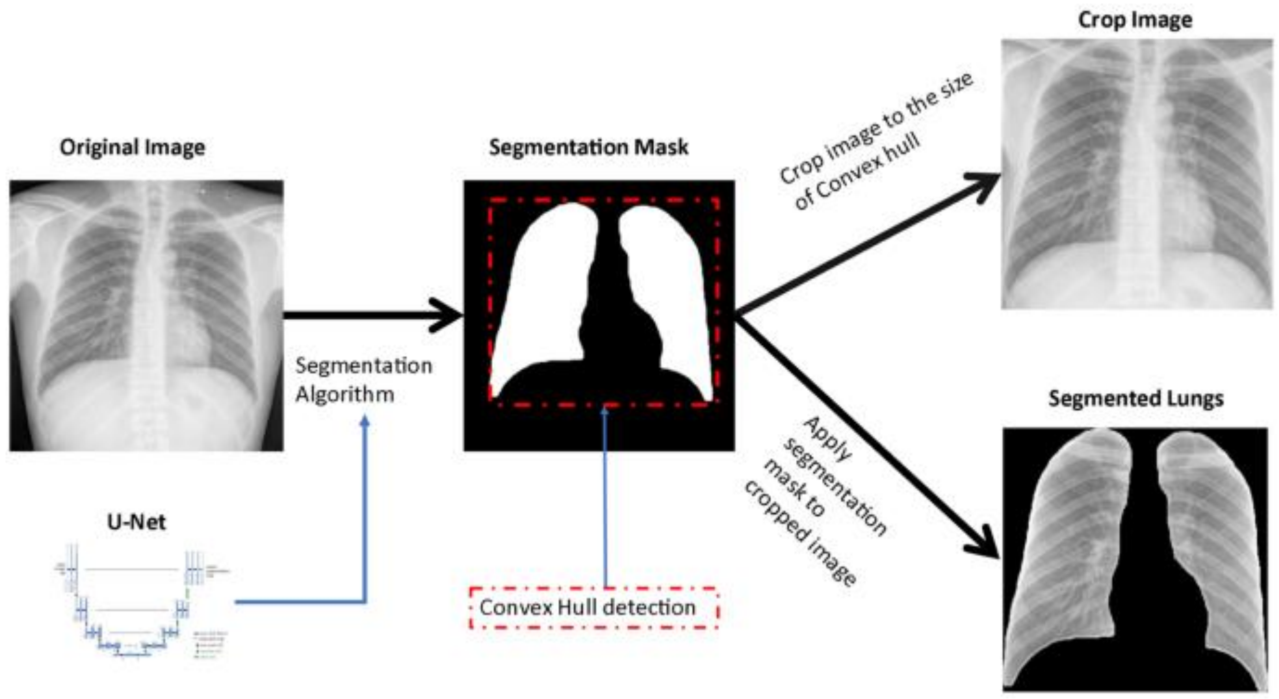


Figure3: Output of the project

6.DATAFLOWCHART

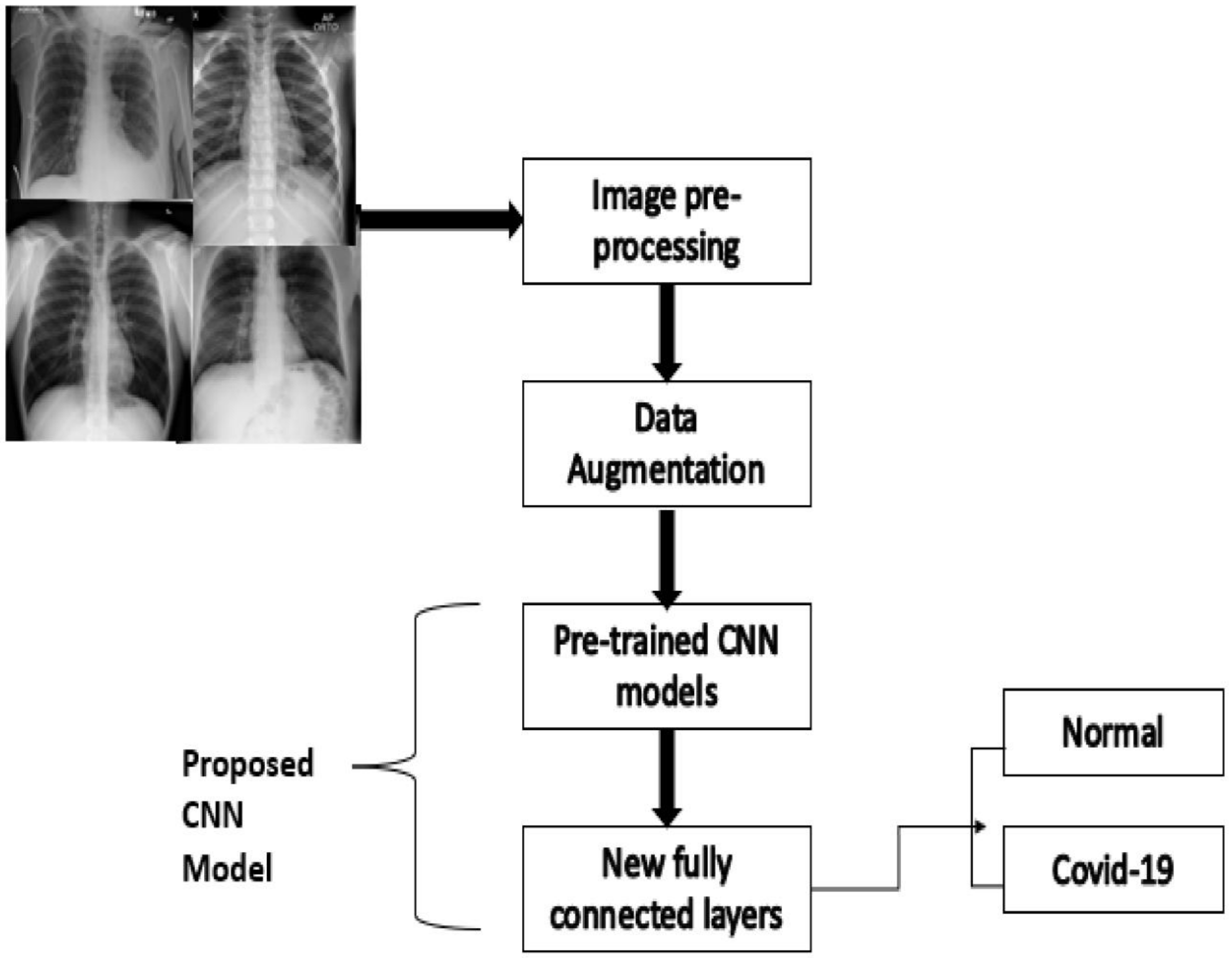


Figure4: Covid-19 Dataflow diagram

USE CASE DIAGRAM

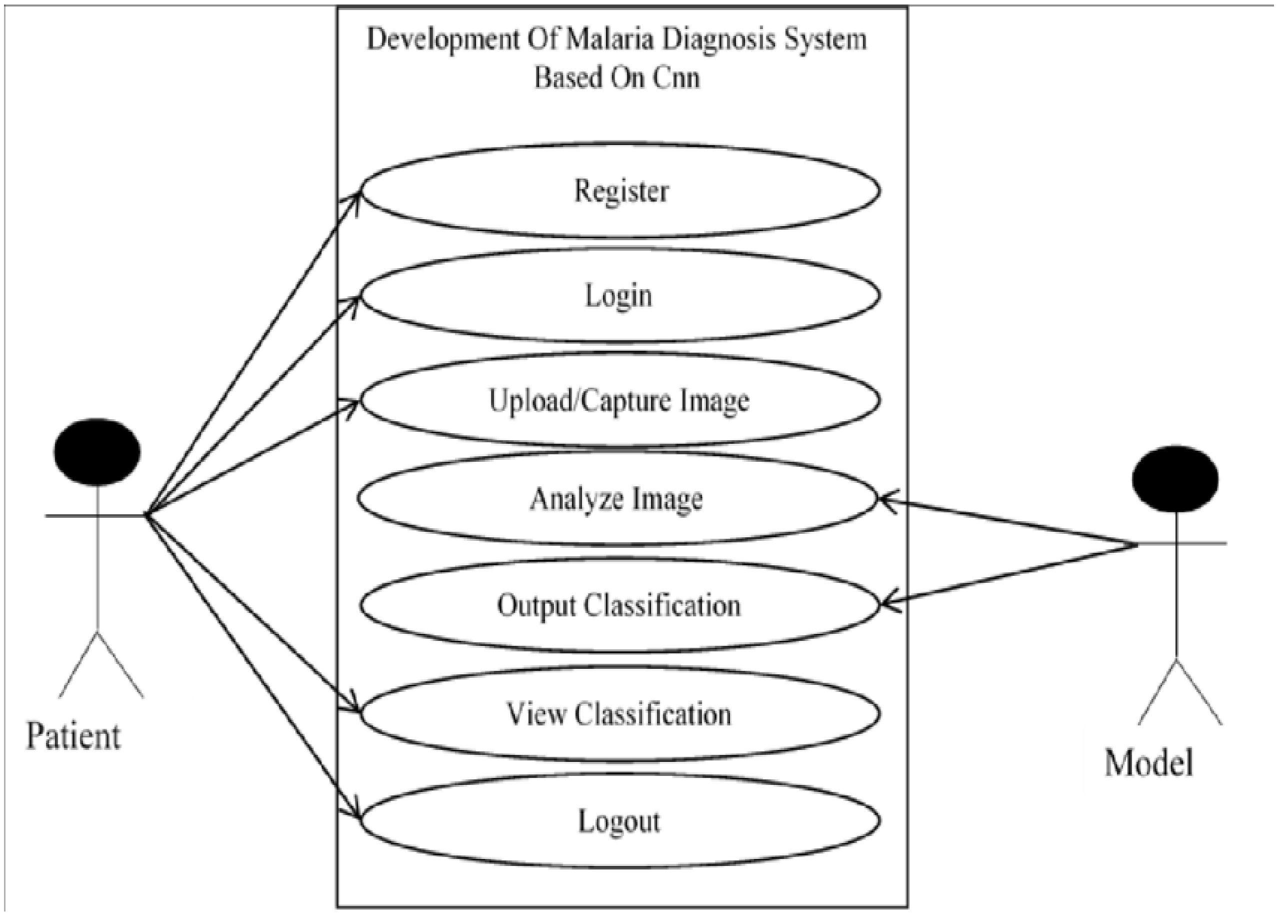


Figure5: Use case diagram

COVIDVISION: ADVANCED COVID-19 DETECTION FROM LUNG X-RAYS WITH DEEP LEARNING USING IBM CLOUD

A UG PROJECT -II REPORT

Submitted to

JAWAHARLAL NEHRU TECOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

MEGHANA NAGAVALI (21UK1A05B6)

VENNELA MENDU (22UK5A0511)

Under the guidance of

A.ASHOK KUMAR

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE (WARANGAL)

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED PHASE –II

This is to certify that the **UG PROJECT PHASE –II** entitled "**COVIDVISION: ADVANCED COVID-19 DETECTION FROM LUNG X-RAYS WITH DEEP LEARNING USING IBM CLOUD**"is being submitted by **MEGHANANAGAVALI (21UK1A05B6) VENNELA MENDU (22UK5A0511),** in partial fulfillment of the requirements of the award of the degree of Bachelor of Technology in Computer science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
A.ASHOK KUMAR	**Dr. K. SHARMILA REDDY**
(Assistant Professor)	(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to Take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHMED**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this **UG PROJECT PHASE –II** in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA REDDY,** Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the **UG PROJECT PHASE –II.**

We express heartfelt thanks to the coordinator **Ms. T. SUSHMA**, Assistant professor, Department of CSE for her constant support and giving necessary guidance for support in completing the **UG PROJECT PHASE -II.**

We express heartfelt thanks to the Guide, **Mr. A.ASHOK KUMAR**, Assistant Professor, Department of CSE for his constant support and giving necessary guidance for completion of this **UG PROJECT PHASE –II.**

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

MEGHANA NAGAVALI	**(21UK1A05B6)**
VENNELA MENDU	**(22UK5A0511)**

TABLE OF CONTENTS

LIST OF FIGURES **PAGENO**

1.INTRODUCTION

In response to the global COVID-19 pandemic, rapid and accurate detection of the virus has become crucial in managing and controlling its spread. Traditional diagnostic methods, such as PCR tests, although effective, can be time-consuming and resource-intensive. Recognizing the potential of artificial intelligence in healthcare, **CovidVision** leverages the power of **deep learning** to provide an innovative solution for COVID-19 detection using **lung X-rays**.

By integrating **advanced deep learning algorithms** with the robust infrastructure of the **IBM Cloud**, CovidVision aims to enhance the diagnostic process. This approach not only speeds up detection but also offers a scalable and efficient solution, particularly beneficial in regions with limited access to conventional testing resources.

The introduction of CovidVision showcases the transformative potential of combining **AI** with **cloud computing** to address critical healthcare challenges. This overview will delve into the core components of the project, including the deep learning model's architecture, the role of IBM Cloud in supporting the system, and the potential implications for global healthcare

UG Project Phase-II involves all the coding and implementation of the design which we have retrieved from the UG Project Phase-I. All the implementation is done and conclusions are retrieved in this phase. We will also work on the applications, advantages, disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase II.

2.CODE SNIPPETS

2.1 PYTHON CODE

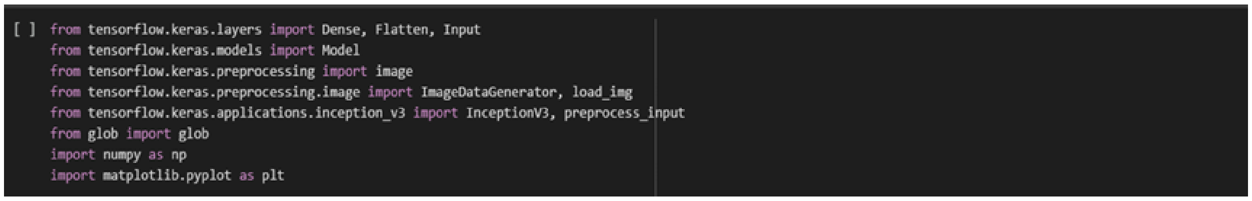
Here we will be building the CovidVision is a system designed to aid in the rapid and accurate detection of COVID-19 infection using deep learning models trained on chest X ray images. The system is deployed and operated using IBM Cloud, leveraging.

This includes the following code snippets:

- Import libraries
- Load Dataset
- Configure ImageDataGenerator class
- Functionality to Train set and Test set
- Feature Extractor
- Adding Dense Layers
- Model Evalution
- Train the model
- Save the model (CSV or Pickle)
- Bulid the Application
- Create HTML webpage
- Create app.py file
- Detect workers

LOAD NECESSARY FILES AND LIBRARIES

let's begin the code by importing the libraries

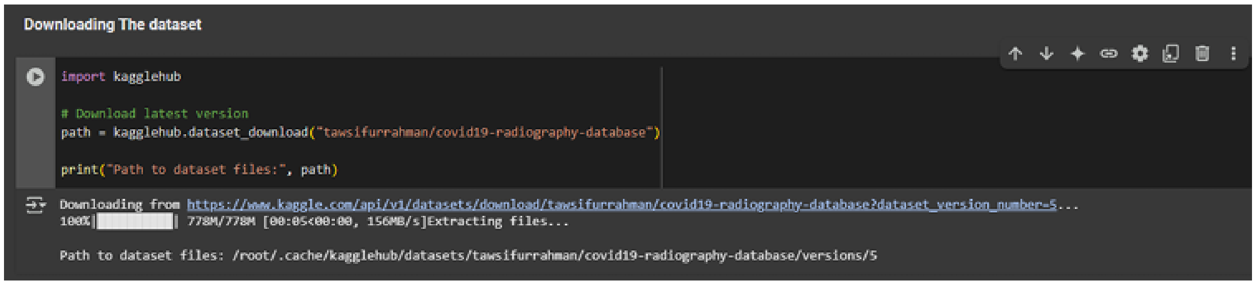


```
[ ]  from tensorflow.keras.layers import Dense, Flatten, Input
     from tensorflow.keras.models import Model
     from tensorflow.keras.preprocessing import image
     from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
     from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
     from glob import glob
     import numpy as np
     import matplotlib.pyplot as plt
```

Figure1: Loading Necessary Libraries

Import all the libraries which are showcased on the image. please refer to the comments for the usage of each library and each initialization

DOWNLOADING THE DATASET



Downloading The dataset

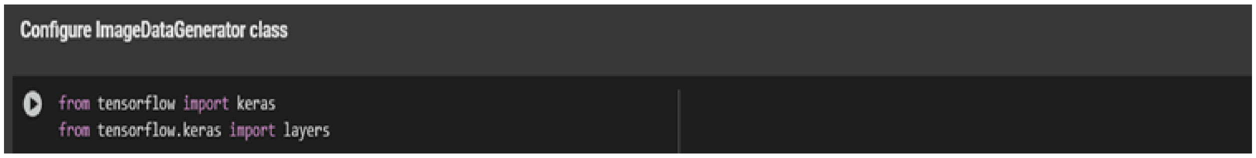
```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("tawsifurrahman/covid19-radiography-database")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/tawsifurrahman/covid19-radiography-database?dataset_version_number=5...
100%|██████████| 778M/778M [00:05<00:00, 156MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/tawsifurrahman/covid19-radiography-database/versions/5
```

Figure2: loading dataset

CONFIGURE IMAGEDATAGENERATOR CLASS

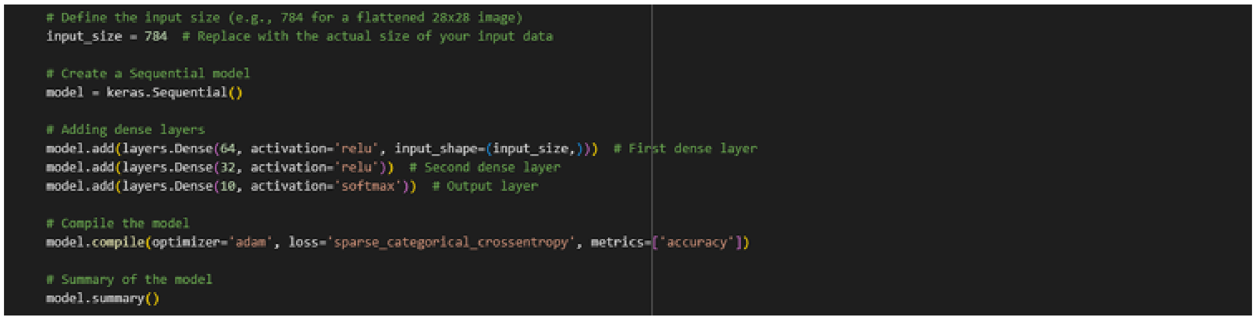
Image data generator will use Keras for image loading and augmentation, and is optimized for preparing a chest X-ray dataset for training a deep learning model like a CNN. You can deploy this pipeline on IBM Cloud after training.



Configure ImageDataGenerator class

```
from tensorflow import keras
from tensorflow.keras import layers
```

Figure3: configure imageDataGenertor class



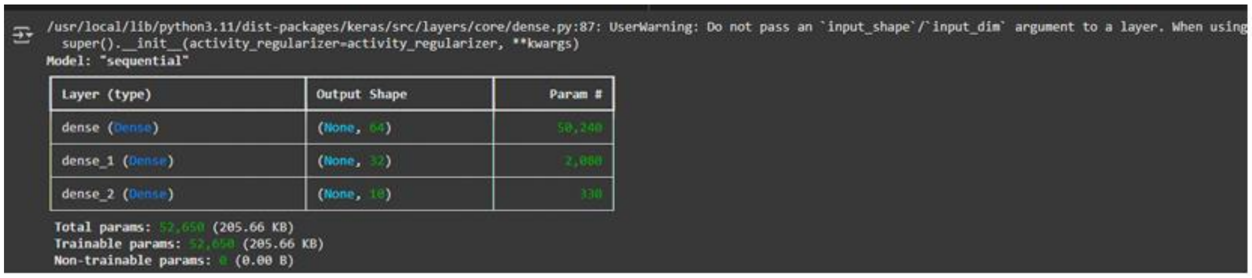
```
# Define the input size (e.g., 784 for a flattened 28x28 image)
input_size = 784  # Replace with the actual size of your input data

# Create a Sequential model
model = keras.Sequential()

# Adding dense layers
model.add(layers.Dense(64, activation='relu', input_shape=(input_size,)))  # First dense layer
model.add(layers.Dense(32, activation='relu'))  # Second dense layer
model.add(layers.Dense(10, activation='softmax'))  # Output layer

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

# Summary of the model
model.summary()
```

Figure3.1: configure imageDataGenertor class

we aim to accurately detect COVID-19 infection from chest X-ray images using deep learning models deployed on IBM Cloud.

The ImageDataGenerator class is a powerful utility provided by Keras (a high-level deep learning API in TensorFlow)

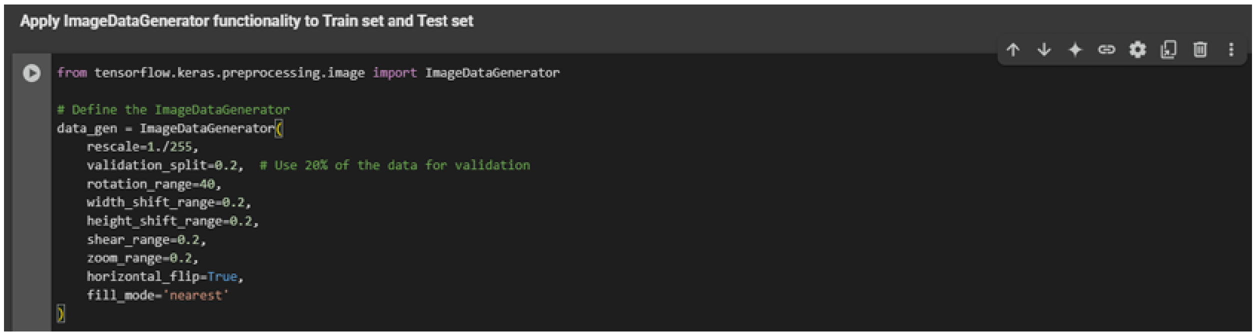
- Preprocess images (such as rescaling pixel values)
- Augment training data in real time (e.g., flipping, rotating, zooming images)
- Generate batches of image data directly from directories
- Split data into training and validation sets



```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 64)             │        50,240 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 32)             │         2,080 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 10)             │           330 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 52,650 (205.66 KB)
 Trainable params: 52,650 (205.66 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure3.2: configure imageDataGenertor class

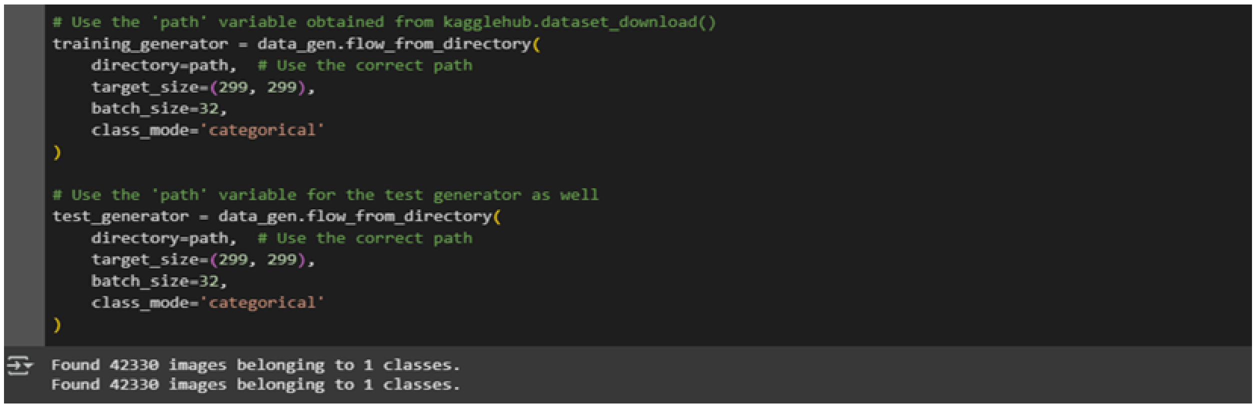
Apply ImageDataGenerator functionality to Train set and Test set


Apply ImageDataGenerator functionality to Train set and Test set

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the ImageDataGenerator
data_gen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,  # Use 20% of the data for validation
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Figure4: Apply ImageDataGenerator functionality to Train set and Test set

This is achieved through the use of Keras's ImageDataGenerator class, which we apply to both the training set and the test (or validation) set.



```
# Use the 'path' variable obtained from kagglehub.dataset_download()
training_generator = data_gen.flow_from_directory(
    directory=path,  # Use the correct path
    target_size=(299, 299),
    batch_size=32,
    class_mode='categorical'
)

# Use the 'path' variable for the test generator as well
test_generator = data_gen.flow_from_directory(
    directory=path,  # Use the correct path
    target_size=(299, 299),
    batch_size=32,
    class_mode='categorical'
)
```

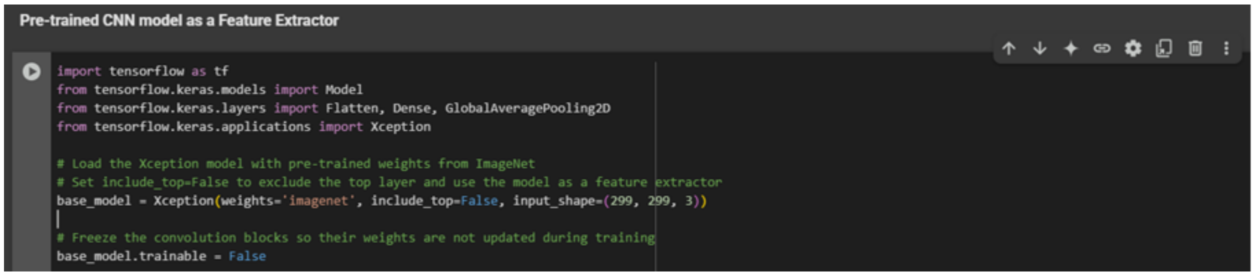
```
Found 42330 images belonging to 1 classes.
Found 42330 images belonging to 1 classes.
```

Figure4.1: Apply ImageDataGenerator functionality to Train set and Test set

- Preprocess images (e.g., normalize pixel values)
- Apply real-time data augmentation (e.g., rotation, shift, zoom)
- Efficiently load images in batches from directories
- Split datasets into training and validation sets

This is especially useful in medical image analysis, where datasets like chest X-rays are often small. The generator increases data variability during training, improving model robustness.

Pre-trained CNN model as a Feature Extractor

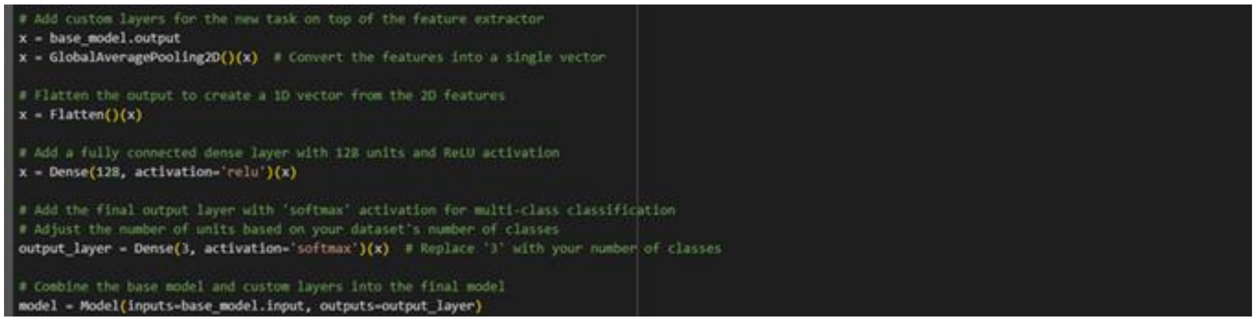


Pre-trained CNN model as a Feature Extractor

```python
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense, GlobalAveragePooling2D
from tensorflow.keras.applications import Xception

# Load the Xception model with pre-trained weights from ImageNet
# Set include_top=False to exclude the top layer and use the model as a feature extractor
base_model = Xception(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

# Freeze the convolution blocks so their weights are not updated during training
base_model.trainable = False
```

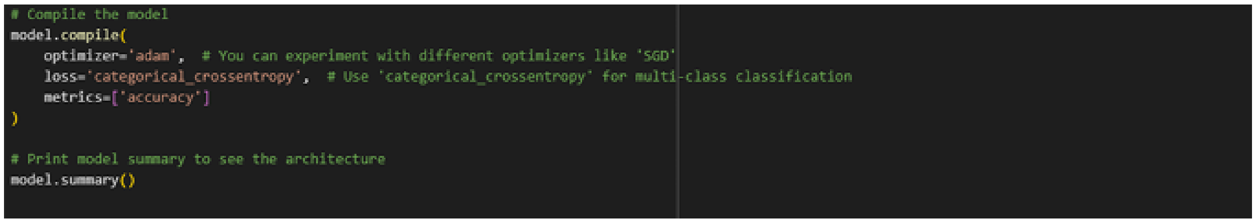
Figure5: Pre-trained CNN model as a Feature Extractor

A pre-trained CNN (e.g., VGG16, ResNet50, or MobileNetV2) has already learned useful feature representations from millions of images. By removing the top (classification) layers, we can use its convolutional base to extract high-level features from our chest X-ray images.



```python
# Add custom layers for the new task on top of the feature extractor
x = base_model.output
x = GlobalAveragePooling2D()(x)  # Convert the features into a single vector

# Flatten the output to create a 1D vector from the 2D features
x = Flatten()(x)

# Add a fully connected dense layer with 128 units and ReLU activation
x = Dense(128, activation='relu')(x)

# Add the final output layer with 'softmax' activation for multi-class classification
# Adjust the number of units based on your dataset's number of classes
output_layer = Dense(3, activation='softmax')(x)  # Replace '3' with your number of classes

# Combine the base model and custom layers into the final model
model = Model(inputs=base_model.input, outputs=output_layer)
```

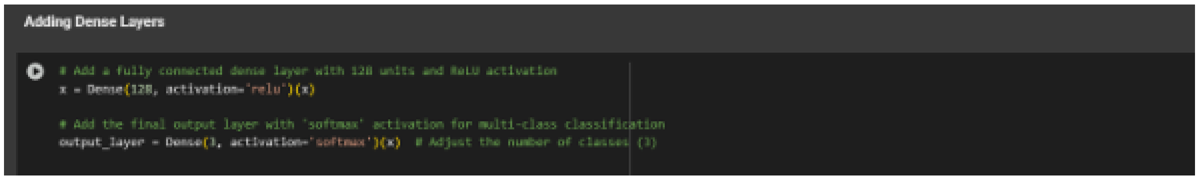
Figure5.1: Pre-trained CNN model as a Feature Extractor



```python
# Compile the model
model.compile(
    optimizer='adam',  # You can experiment with different optimizers like 'SGD'
    loss='categorical_crossentropy',  # Use 'categorical_crossentropy' for multi-class classification
    metrics=['accuracy']
)

# Print model summary to see the architecture
model.summary()
```

Figure5.2: Pre-trained CNN model as a Feature Extractor

Adding Dense Layers



Adding Dense Layers

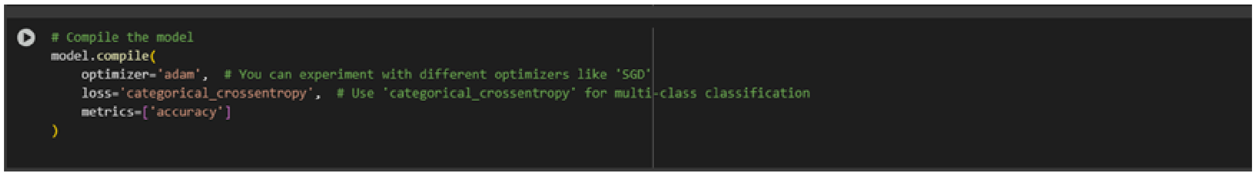
```
# Add a fully connected dense layer with 128 units and ReLU activation
x = Dense(128, activation='relu')(x)

# Add the final output layer with 'softmax' activation for multi-class classification
output_layer = Dense(3, activation='softmax')(x)  # Adjust the number of classes (3)
```

Figure6: Adding Dense Layers

• A Dense layer connects every input node to every output node.

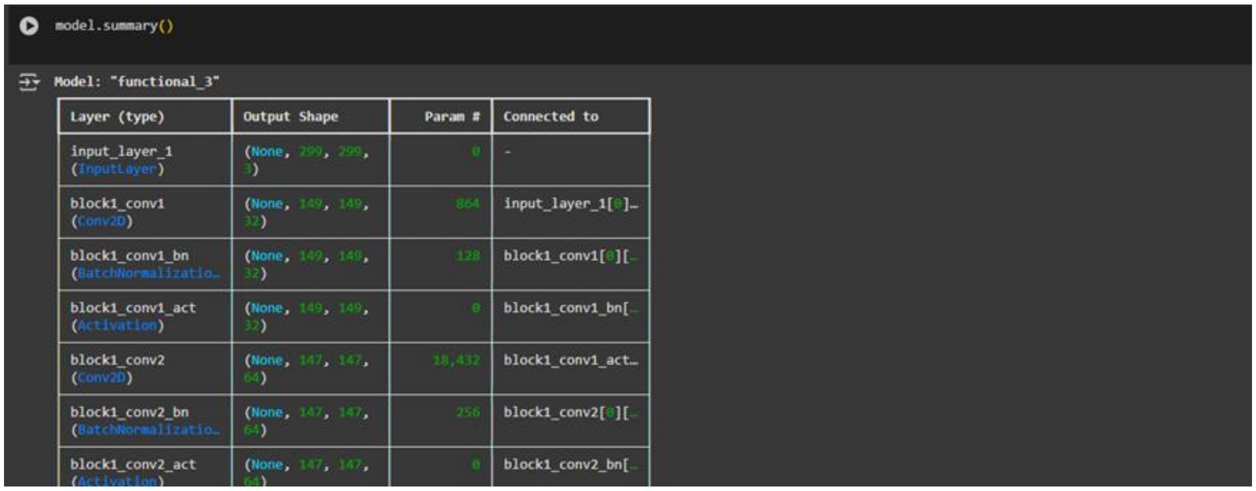
• It helps the model learn complex patterns in the features extracted by the CNN

• We typically add one or more Dense layers, followed by a final output layer for classification.



```
# Compile the model
model.compile(
    optimizer='adam',  # You can experiment with different optimizers like 'SGD'
    loss='categorical_crossentropy',  # Use 'categorical_crossentropy' for multi-class classification
    metrics=['accuracy']
)
```

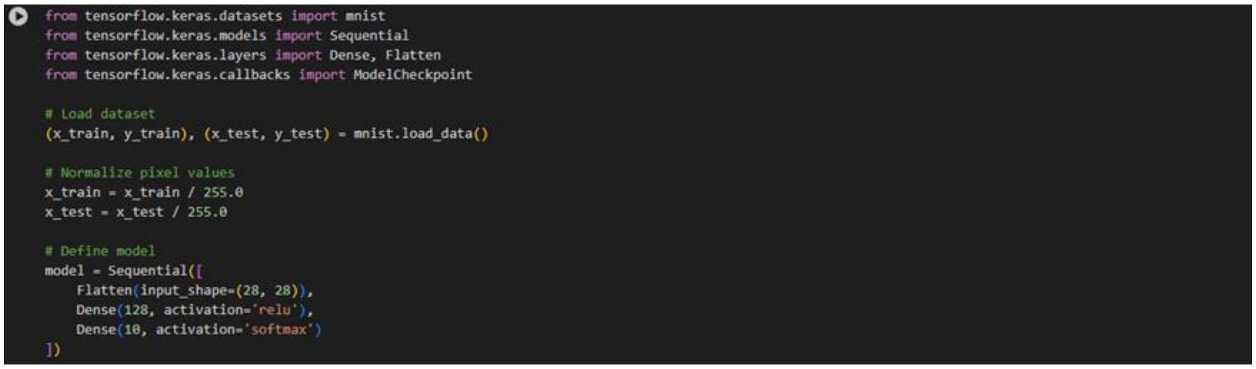
Figure6.1: Adding Dense Layers



```
model.summary()
```

```
Model: "functional_3"
┌──────────────────────┬─────────────────────┬──────────┬──────────────────┐
│ Layer (type)         │ Output Shape        │  Param # │ Connected to     │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ input_layer_1        │ (None, 299, 299,    │        0 │ -                │
│ (InputLayer)         │ 3)                  │          │                  │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ block1_conv1         │ (None, 149, 149,    │      864 │ input_layer_1[0]… │
│ (Conv2D)             │ 32)                 │          │                  │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ block1_conv1_bn      │ (None, 149, 149,    │      128 │ block1_conv1[0][… │
│ (BatchNormalizatio…  │ 32)                 │          │                  │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ block1_conv1_act     │ (None, 149, 149,    │        0 │ block1_conv1_bn[… │
│ (Activation)         │ 32)                 │          │                  │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ block1_conv2         │ (None, 147, 147,    │   18,432 │ block1_conv1_act… │
│ (Conv2D)             │ 64)                 │          │                  │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ block1_conv2_bn      │ (None, 147, 147,    │      256 │ block1_conv2[0][… │
│ (BatchNormalizatio…  │ 64)                 │          │                  │
├──────────────────────┼─────────────────────┼──────────┼──────────────────┤
│ block1_conv2_act     │ (None, 147, 147,    │        0 │ block1_conv2_bn[… │
│ (Activation)         │ 64)                 │          │                  │
```

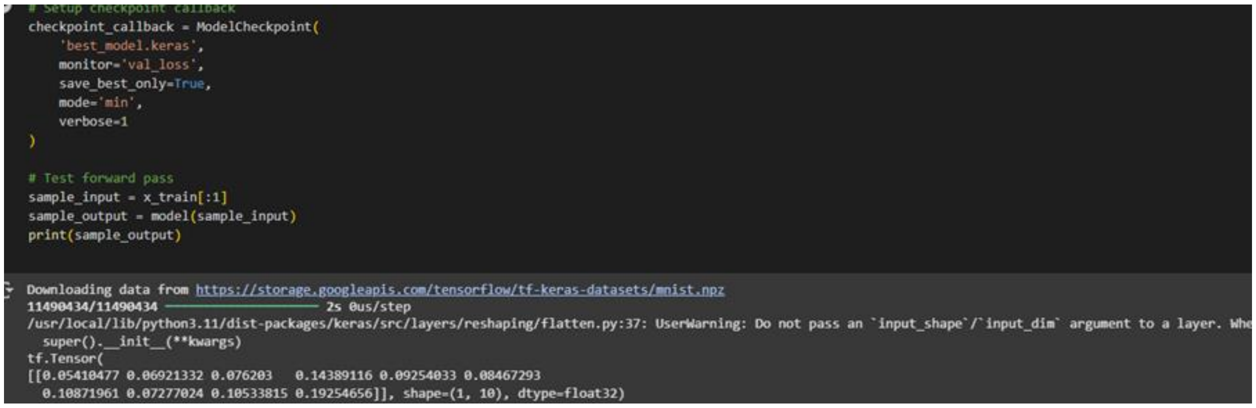
Figure6.2: Adding Dense Layers



```python
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.callbacks import ModelCheckpoint

# Load dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize pixel values
x_train = x_train / 255.0
x_test = x_test / 255.0

# Define model
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Figure6.3: Adding Dense Layers

After using a pre-trained CNN model (like MobileNetV2 or ResNet50) as a feature extractor, the next step in CovidVision is to add Dense layers (also known as fully connected layers) tocomplete the deep learning model.



```python
# Setup checkpoint callback
checkpoint_callback = ModelCheckpoint(
    'best_model.keras',
    monitor='val_loss',
    save_best_only=True,
    mode='min',
    verbose=1
)

# Test forward pass
sample_input = x_train[:1]
sample_output = model(sample_input)
print(sample_output)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
  super().__init__(**kwargs)
tf.Tensor(
[[0.05410477 0.06921332 0.076203   0.14389116 0.09254033 0.08467293
  0.10871961 0.07277024 0.10533815 0.19254656]], shape=(1, 10), dtype=float32)
```

Figure6.4: Adding Dense Layers

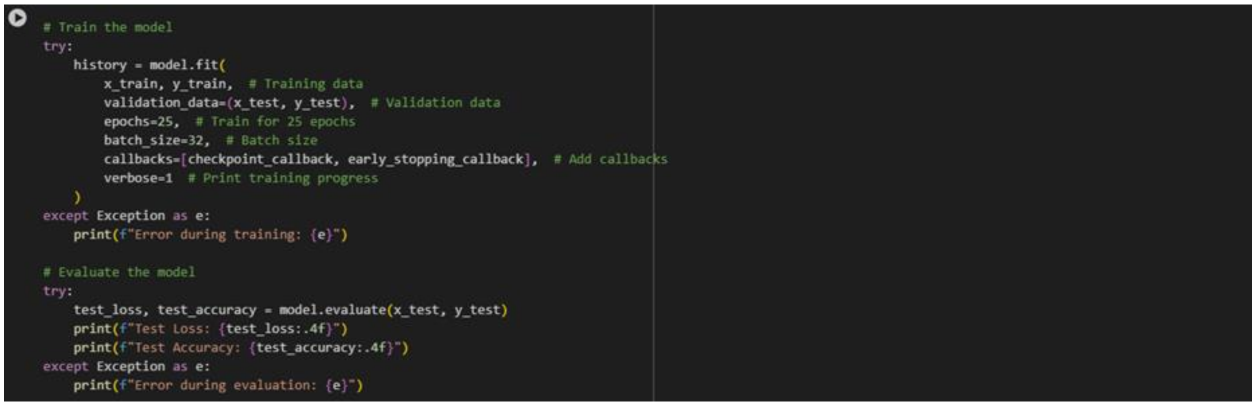
Train the model

Once the CNN feature extractor and Dense layers have been added to the model, the next crucial step in CovidVision is to train the model using the prepared dataset of chest X-ray images. This is the stage where the model learns to detect patterns that distinguish COVID-positive lungs from normal ones.

• Feeding input X-ray images and their labels (COVID/Normal) to the model.

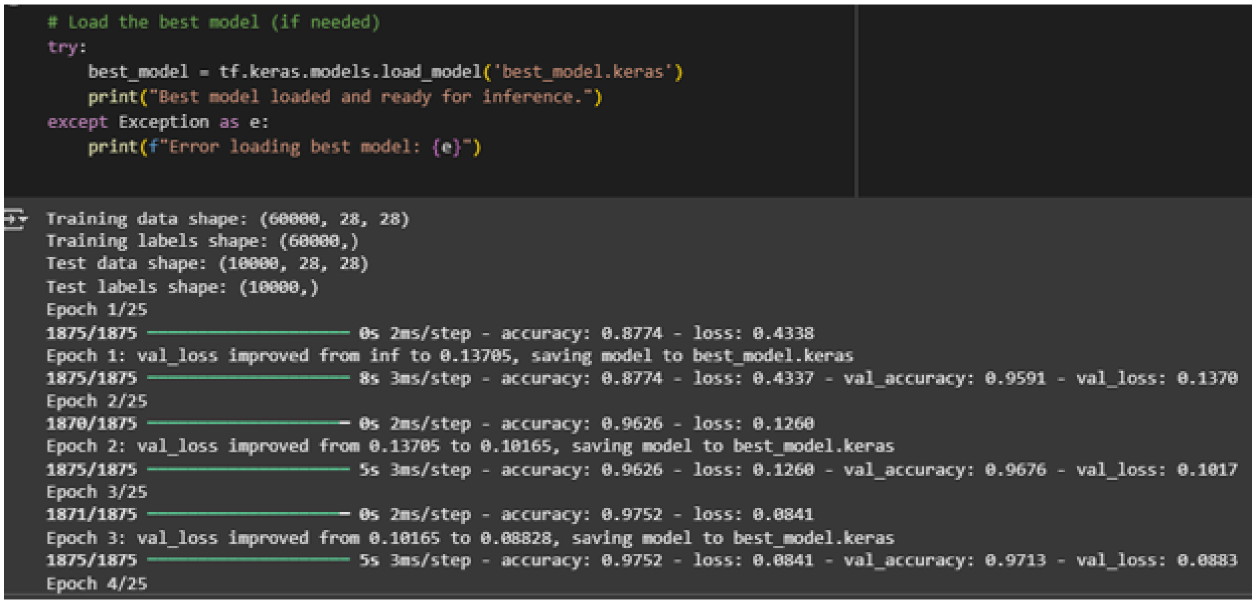
• The model predicts output labels, compares them with actual labels, and calculates loss.

• It uses backpropagation and gradient descent to adjust its internal weights and improve predictions.



```
# Train the model
try:
    history = model.fit(
        x_train, y_train,  # Training data
        validation_data=(x_test, y_test),  # Validation data
        epochs=25,  # Train for 25 epochs
        batch_size=32,  # Batch size
        callbacks=[checkpoint_callback, early_stopping_callback],  # Add callbacks
        verbose=1  # Print training progress
    )
except Exception as e:
    print(f"Error during training: {e}")

# Evaluate the model
try:
    test_loss, test_accuracy = model.evaluate(x_test, y_test)
    print(f"Test Loss: {test_loss:.4f}")
    print(f"Test Accuracy: {test_accuracy:.4f}")
except Exception as e:
    print(f"Error during evaluation: {e}")
```

Figure7: Train the mode



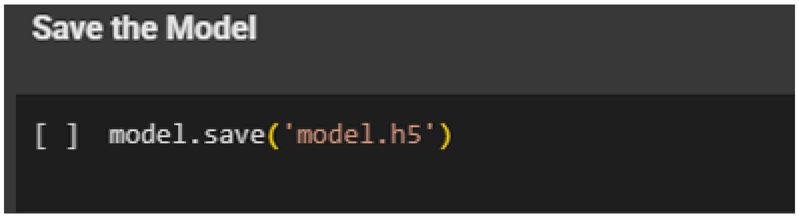
```
# Load the best model (if needed)
try:
    best_model = tf.keras.models.load_model('best_model.keras')
    print("Best model loaded and ready for inference.")
except Exception as e:
    print(f"Error loading best model: {e}")
```

```
Training data shape: (60000, 28, 28)
Training labels shape: (60000,)
Test data shape: (10000, 28, 28)
Test labels shape: (10000,)
Epoch 1/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8774 - loss: 0.4338
Epoch 1: val_loss improved from inf to 0.13705, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.8774 - loss: 0.4337 - val_accuracy: 0.9591 - val_loss: 0.1370
Epoch 2/25
1870/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9626 - loss: 0.1260
Epoch 2: val_loss improved from 0.13705 to 0.10165, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9626 - loss: 0.1260 - val_accuracy: 0.9676 - val_loss: 0.1017
Epoch 3/25
1871/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9752 - loss: 0.0841
Epoch 3: val_loss improved from 0.10165 to 0.08828, saving model to best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9752 - loss: 0.0841 - val_accuracy: 0.9713 - val_loss: 0.0883
Epoch 4/25
```

Figure7.1: Train the mode

Save the model

Now run the code



Save the Model

```
[ ]  model.save('model.h5')
```

Figure8: Save the mode

2.2 HTML CODE

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where they have to upload the image for predictions. The entered image is given to the saved model and prediction is showcased on the UI.

To enhance accessibility and usability, the covid-19 of lung x-ray system includes a simple web interface built using HTML and integrated with a backend (e.g., Flask or FastAPI). This interface allows users to upload images view real-time detection results, and receive visual feedback— all through a browser.

This section has the following tasks

• Building HTML Pages

• Building server side script

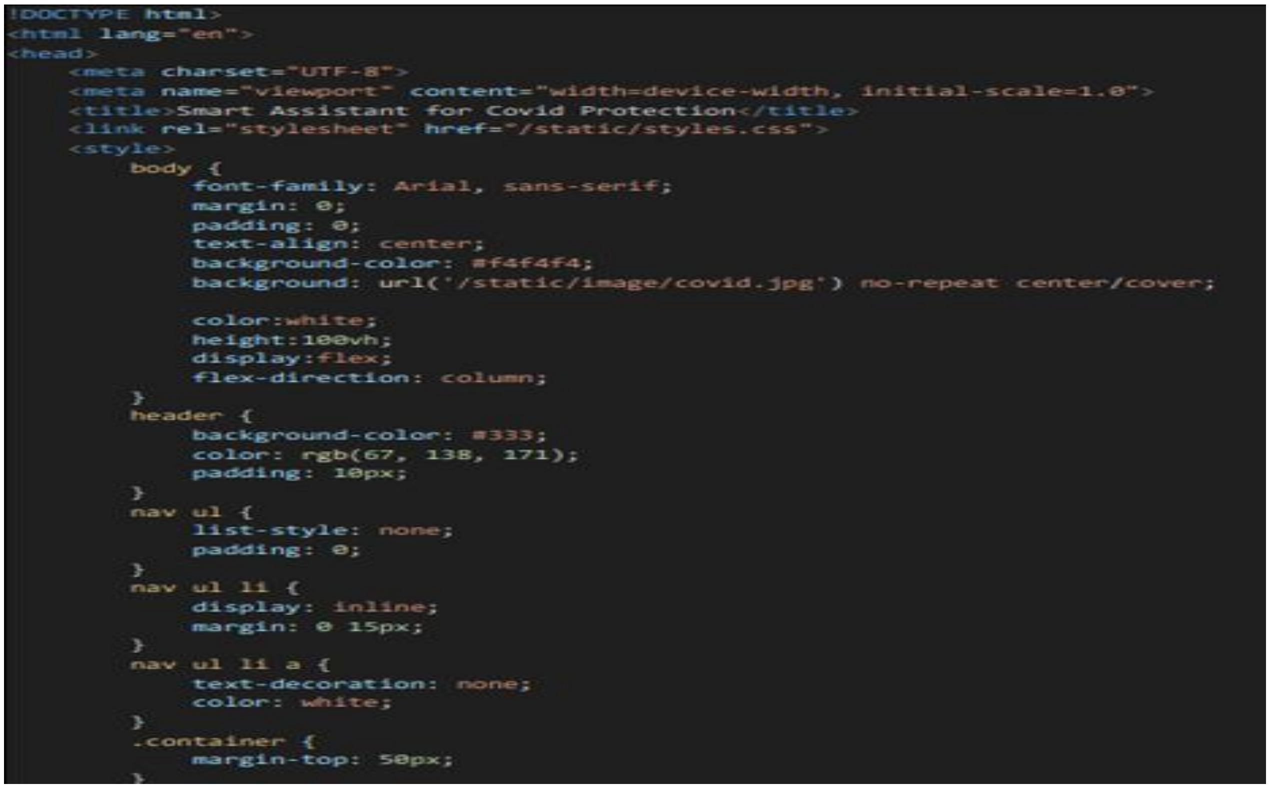
Purpose of HTML Output Pages

• To provide an interactive and user-friendly UI for Covid-19 personnel or operators.

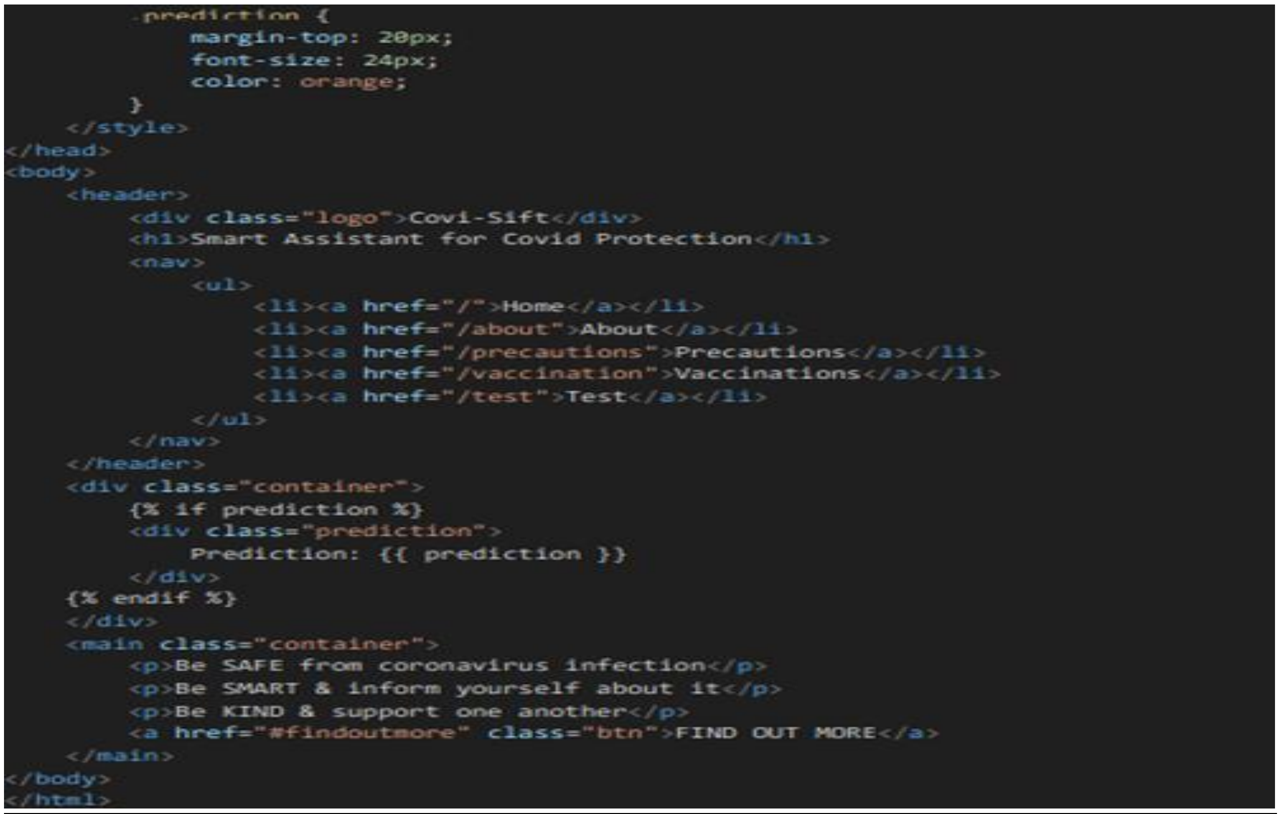
• To display model predictions visually, with bounding boxes and labels.

• To allow image upload input and real-time response.

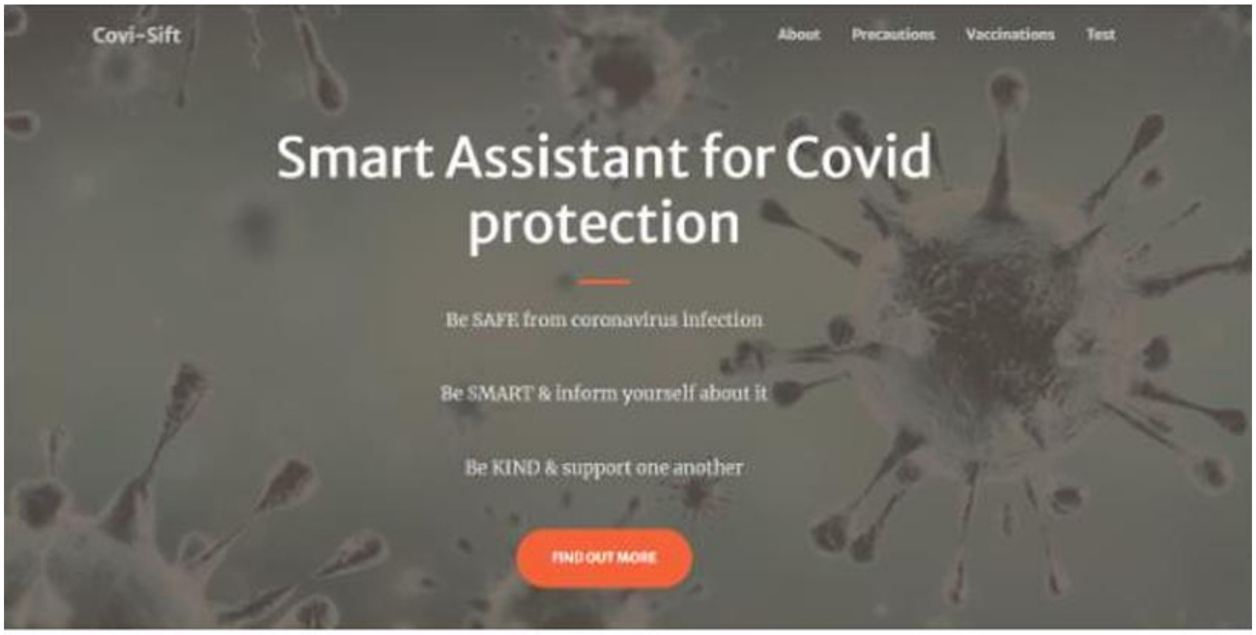
1.home.html



```
!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Smart Assistant for Covid Protection</title>
    <link rel="stylesheet" href="/static/styles.css">
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            text-align: center;
            background-color: #f4f4f4;
            background: url('/static/image/covid.jpg') no-repeat center/cover;

            color:white;
            height:100vh;
            display:flex;
            flex-direction: column;
        }
        header {
            background-color: #333;
            color: rgb(67, 138, 171);
            padding: 10px;
        }
        nav ul {
            list-style: none;
            padding: 0;
        }
        nav ul li {
            display: inline;
            margin: 0 15px;
        }
        nav ul li a {
            text-decoration: none;
            color: white;
        }
        .container {
            margin-top: 50px;
        }
```



```
        .prediction {
            margin-top: 20px;
            font-size: 24px;
            color: orange;
        }
    </style>
</head>
<body>
    <header>
        <div class="logo">Covi-Sift</div>
        <h1>Smart Assistant for Covid Protection</h1>
        <nav>
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/about">About</a></li>
                <li><a href="/precautions">Precautions</a></li>
                <li><a href="/vaccination">Vaccinations</a></li>
                <li><a href="/test">Test</a></li>
            </ul>
        </nav>
    </header>
    <div class="container">
        {% if prediction %}
        <div class="prediction">
            Prediction: {{ prediction }}
        </div>
    {% endif %}
    </div>
    <main class="container">
        <p>Be SAFE from coronavirus infection</p>
        <p>Be SMART & inform yourself about it</p>
        <p>Be KIND & support one another</p>
        <a href="#findoutmore" class="btn">FIND OUT MORE</a>
    </main>
</body>
</html>
```



Smart Assistant for Covid protection

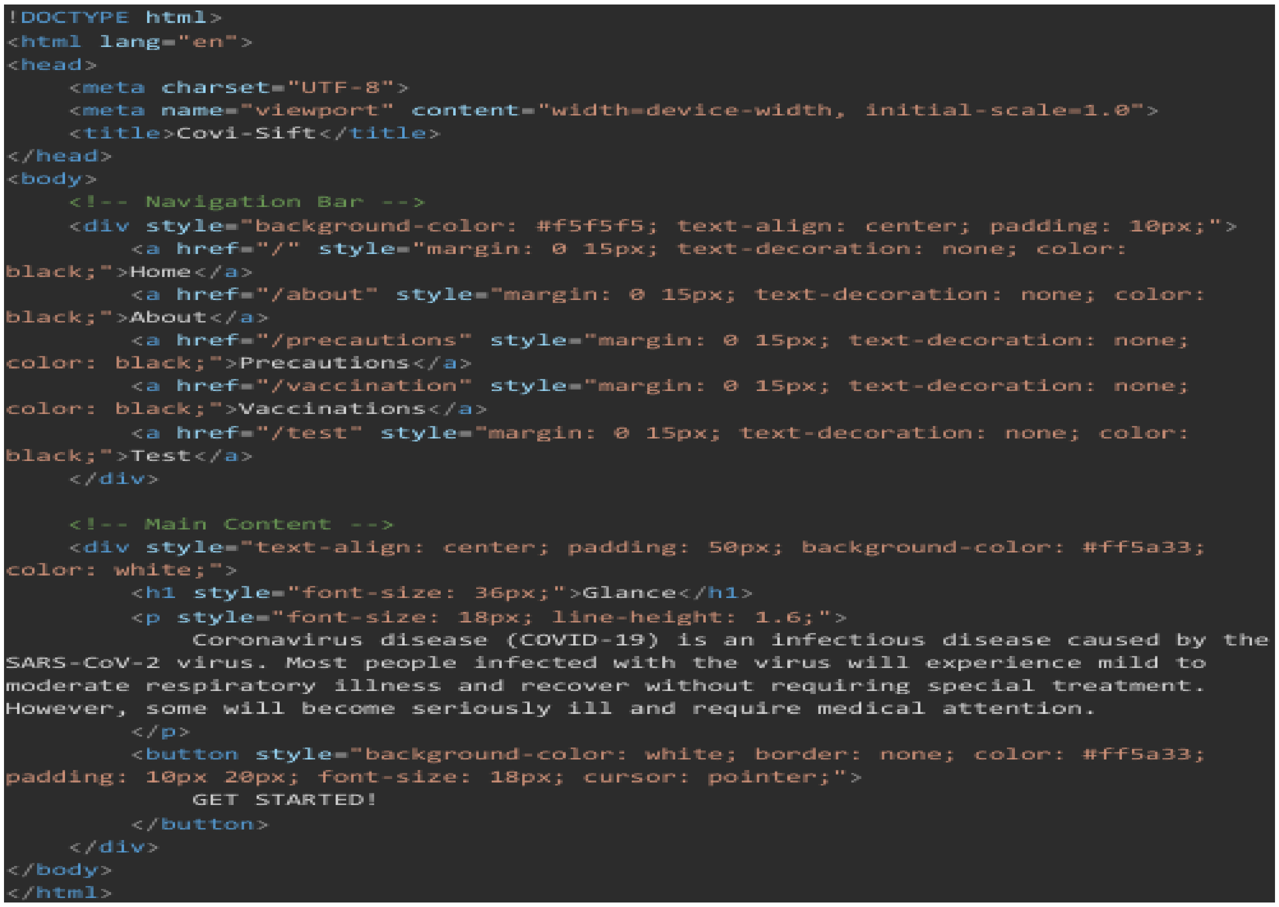
Be SAFE from coronavirus infection

Be SMART & inform yourself about it

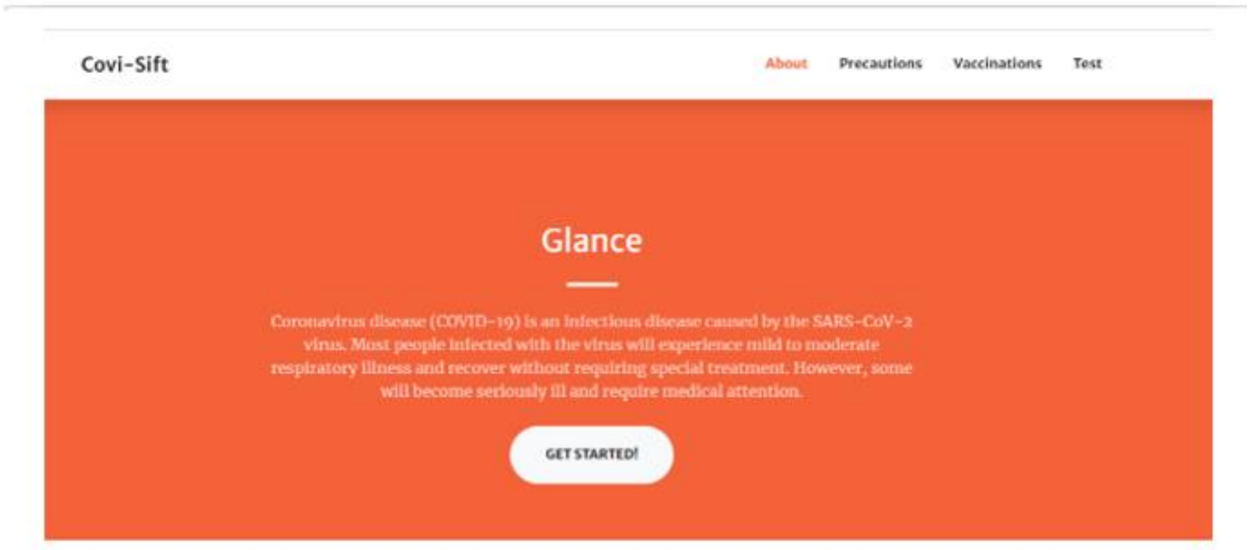
Be KIND & support one another

Figure9: HTML code for Home Page

2.About.html



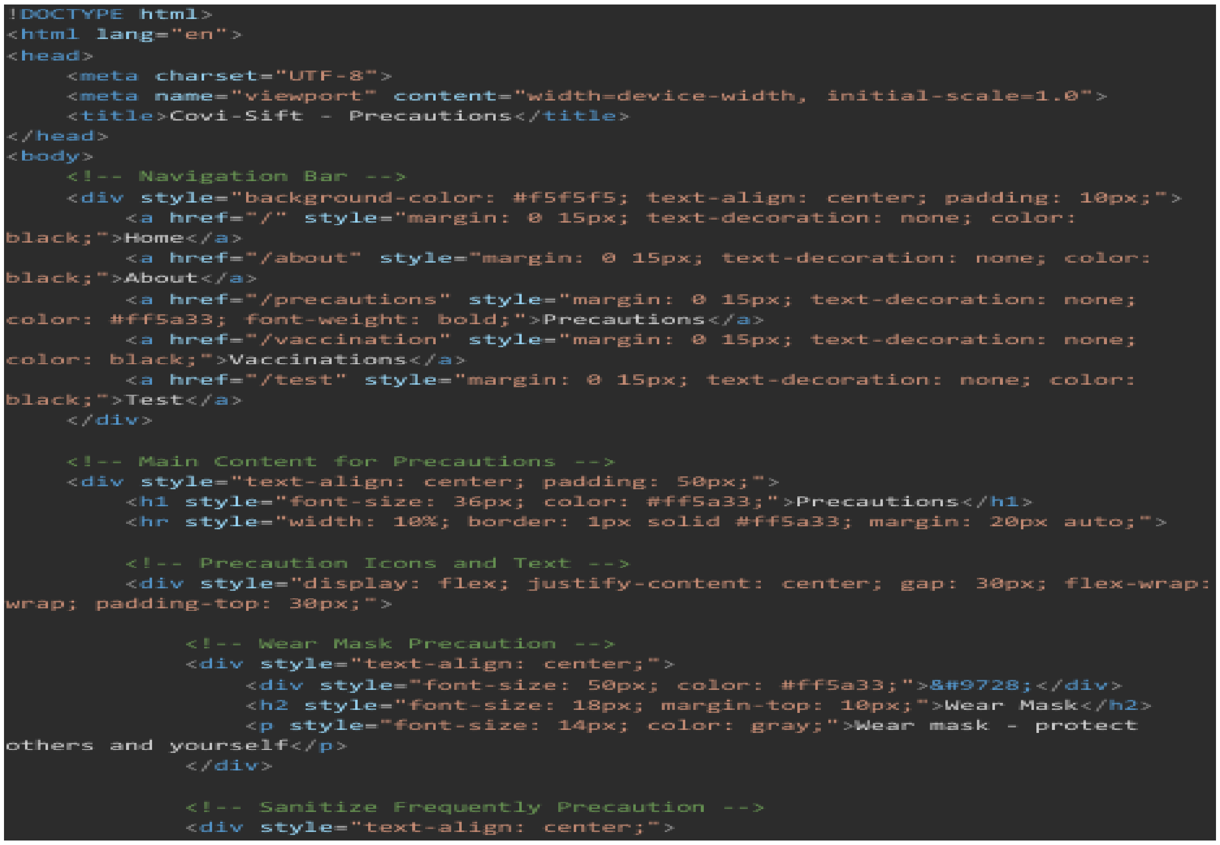
```
!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Covi-Sift</title>
</head>
<body>
    <!-- Navigation Bar -->
    <div style="background-color: #f5f5f5; text-align: center; padding: 10px;">
        <a href="/" style="margin: 0 15px; text-decoration: none; color:
black;">Home</a>
        <a href="/about" style="margin: 0 15px; text-decoration: none; color:
black;">About</a>
        <a href="/precautions" style="margin: 0 15px; text-decoration: none;
color: black;">Precautions</a>
        <a href="/vaccination" style="margin: 0 15px; text-decoration: none;
color: black;">Vaccinations</a>
        <a href="/test" style="margin: 0 15px; text-decoration: none; color:
black;">Test</a>
    </div>

    <!-- Main Content -->
    <div style="text-align: center; padding: 50px; background-color: #ff5a33;
color: white;">
        <h1 style="font-size: 36px;">Glance</h1>
        <p style="font-size: 18px; line-height: 1.6;">
            Coronavirus disease (COVID-19) is an infectious disease caused by the
SARS-CoV-2 virus. Most people infected with the virus will experience mild to
moderate respiratory illness and recover without requiring special treatment.
However, some will become seriously ill and require medical attention.
        </p>
        <button style="background-color: white; border: none; color: #ff5a33;
padding: 10px 20px; font-size: 18px; cursor: pointer;">
            GET STARTED!
        </button>
    </div>
</body>
</html>
```



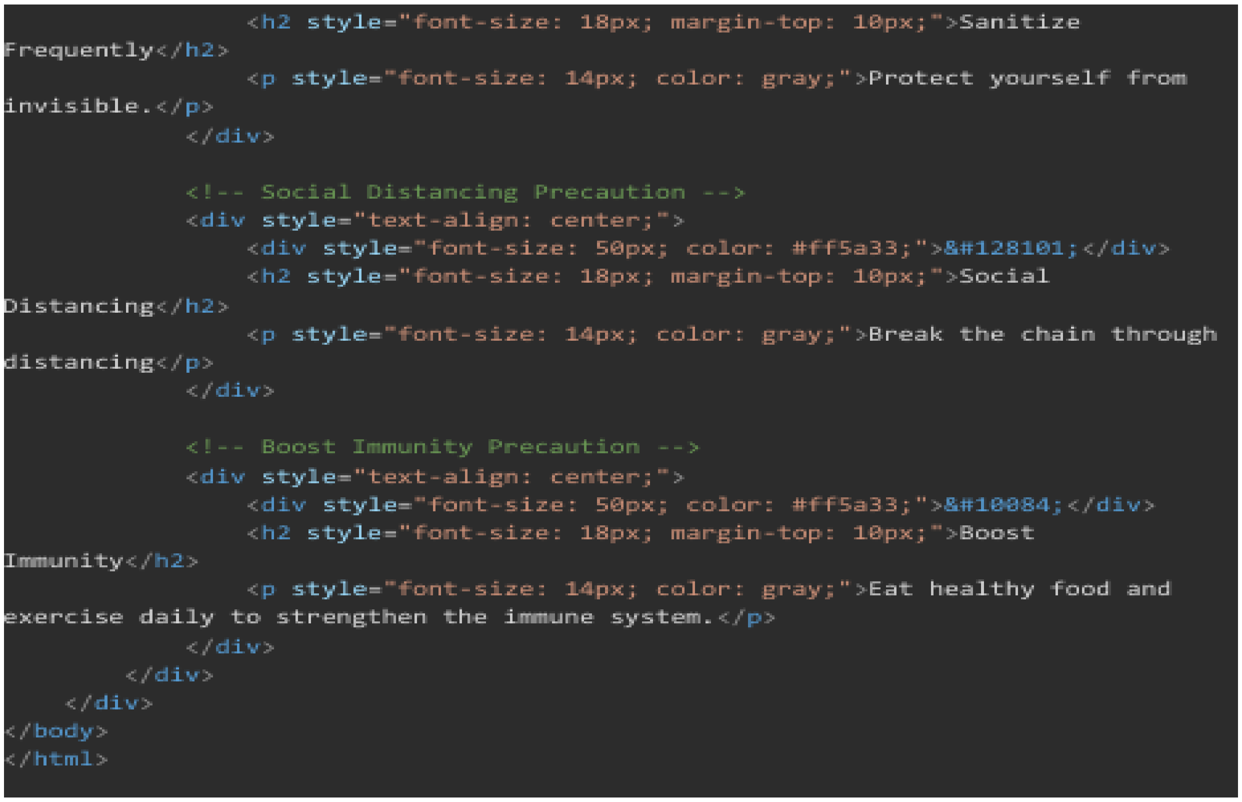
Coronavirus disease (COVID-19) is an infectious disease caused by the SARS-CoV-2 virus. Most people infected with the virus will experience mild to moderate respiratory illness and recover without requiring special treatment. However, some will become seriously ill and require medical attention.

Figure10: HTML code for About page

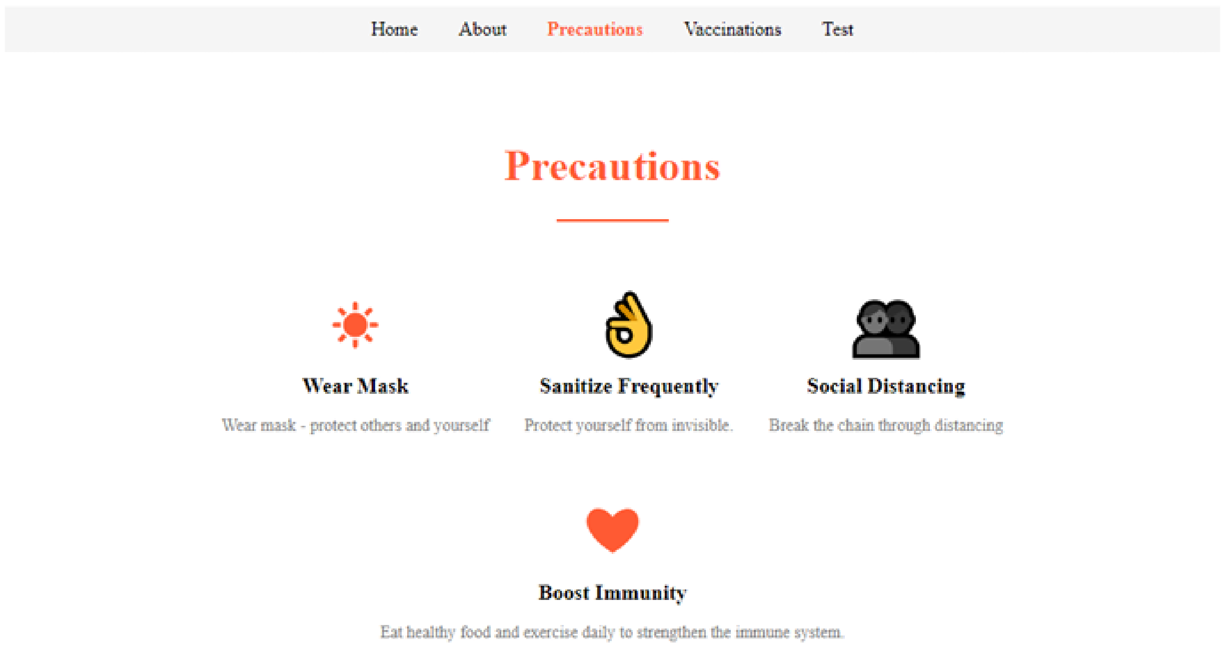
3.Precaution.html



```
!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Covi-Sift - Precautions</title>
</head>
<body>
    <!-- Navigation Bar -->
    <div style="background-color: #f5f5f5; text-align: center; padding: 10px;">
        <a href="/" style="margin: 0 15px; text-decoration: none; color:
black;">Home</a>
        <a href="/about" style="margin: 0 15px; text-decoration: none; color:
black;">About</a>
        <a href="/precautions" style="margin: 0 15px; text-decoration: none;
color: #ff5a33; font-weight: bold;">Precautions</a>
        <a href="/vaccination" style="margin: 0 15px; text-decoration: none;
color: black;">Vaccinations</a>
        <a href="/test" style="margin: 0 15px; text-decoration: none; color:
black;">Test</a>
    </div>

    <!-- Main Content for Precautions -->
    <div style="text-align: center; padding: 50px;">
        <h1 style="font-size: 36px; color: #ff5a33;">Precautions</h1>
        <hr style="width: 10%; border: 1px solid #ff5a33; margin: 20px auto;">

        <!-- Precaution Icons and Text -->
        <div style="display: flex; justify-content: center; gap: 30px; flex-wrap:
wrap; padding-top: 30px;">

            <!-- Wear Mask Precaution -->
            <div style="text-align: center;">
                <div style="font-size: 50px; color: #ff5a33;">&#9728;</div>
                <h2 style="font-size: 18px; margin-top: 10px;">Wear Mask</h2>
                <p style="font-size: 14px; color: gray;">Wear mask - protect
others and yourself</p>
            </div>

            <!-- Sanitize Frequently Precaution -->
            <div style="text-align: center;">
```



```
                <h2 style="font-size: 18px; margin-top: 10px;">Sanitize
Frequently</h2>
                <p style="font-size: 14px; color: gray;">Protect yourself from
invisible.</p>
            </div>

            <!-- Social Distancing Precaution -->
            <div style="text-align: center;">
                <div style="font-size: 50px; color: #ff5a33;">&#128101;</div>
                <h2 style="font-size: 18px; margin-top: 10px;">Social
Distancing</h2>
                <p style="font-size: 14px; color: gray;">Break the chain through
distancing</p>
            </div>

            <!-- Boost Immunity Precaution -->
            <div style="text-align: center;">
                <div style="font-size: 50px; color: #ff5a33;">&#10084;</div>
                <h2 style="font-size: 18px; margin-top: 10px;">Boost
Immunity</h2>
                <p style="font-size: 14px; color: gray;">Eat healthy food and
exercise daily to strengthen the immune system.</p>
            </div>
        </div>
    </div>
</body>
</html>
```



Precautions

Wear Mask

Wear mask - protect others and yourself

Sanitize Frequently

Protect yourself from invisible.

Social Distancing

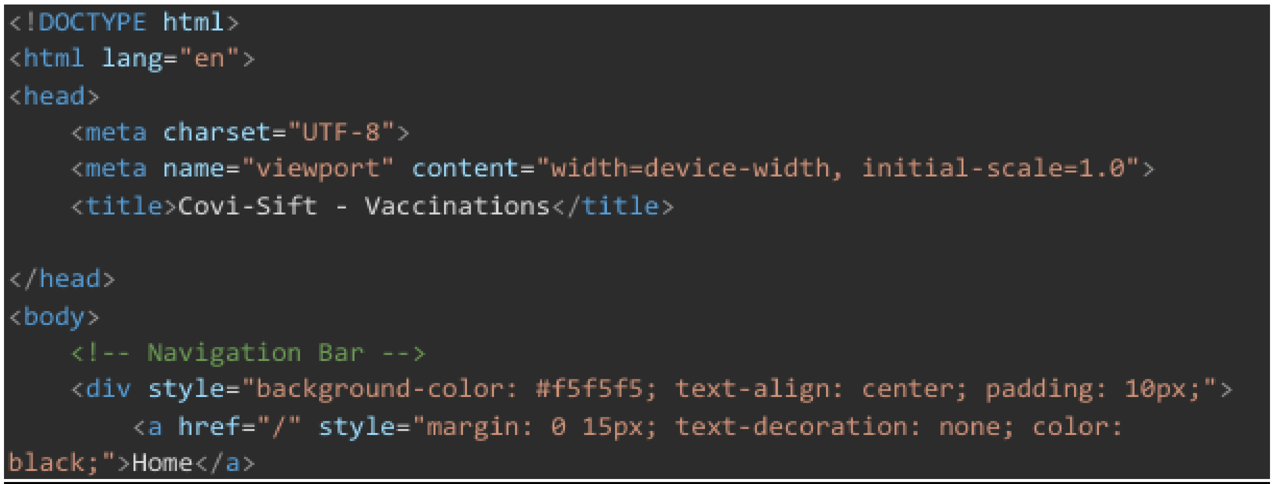
Break the chain through distancing

Boost Immunity

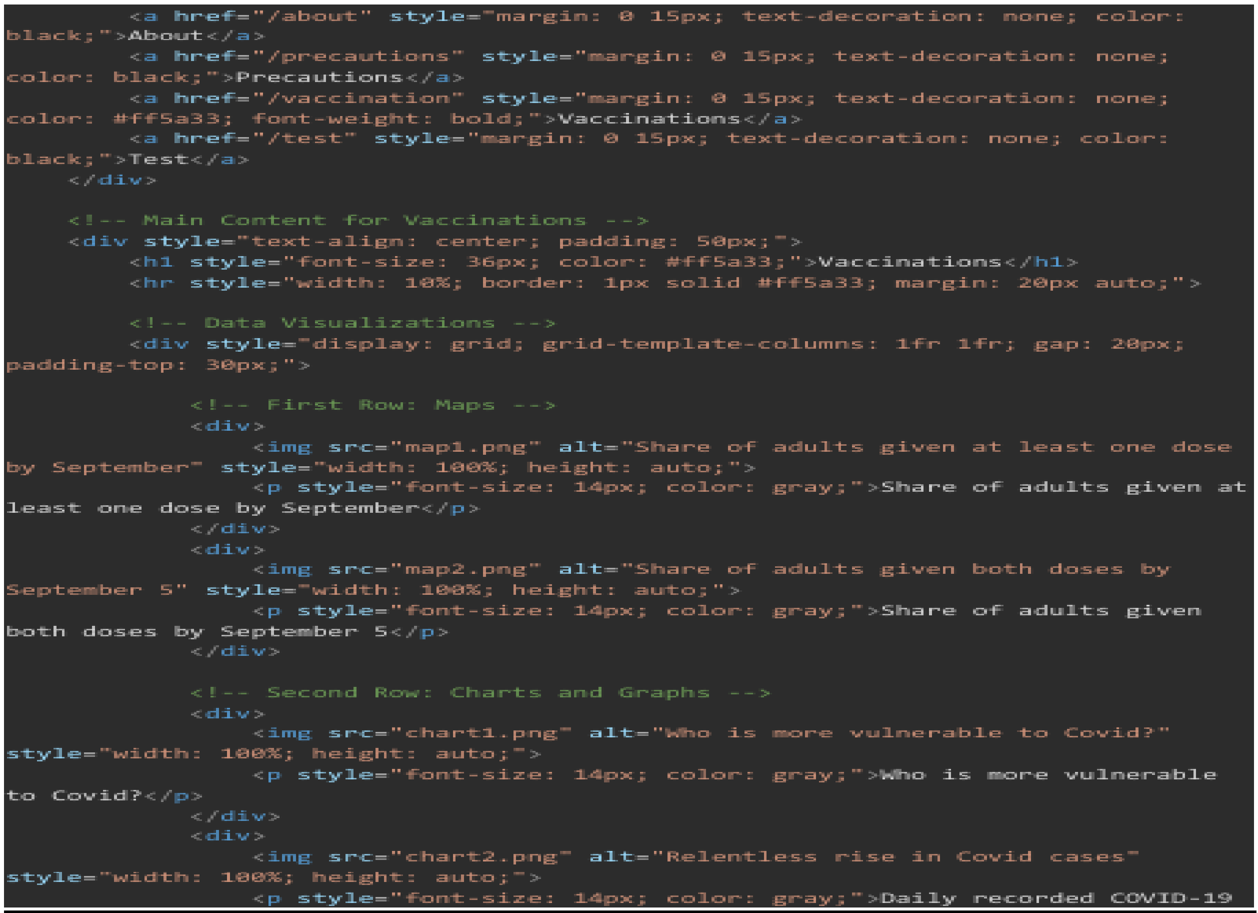
Eat healthy food and exercise daily to strengthen the immune system.

Figure11: HTML code for precaution

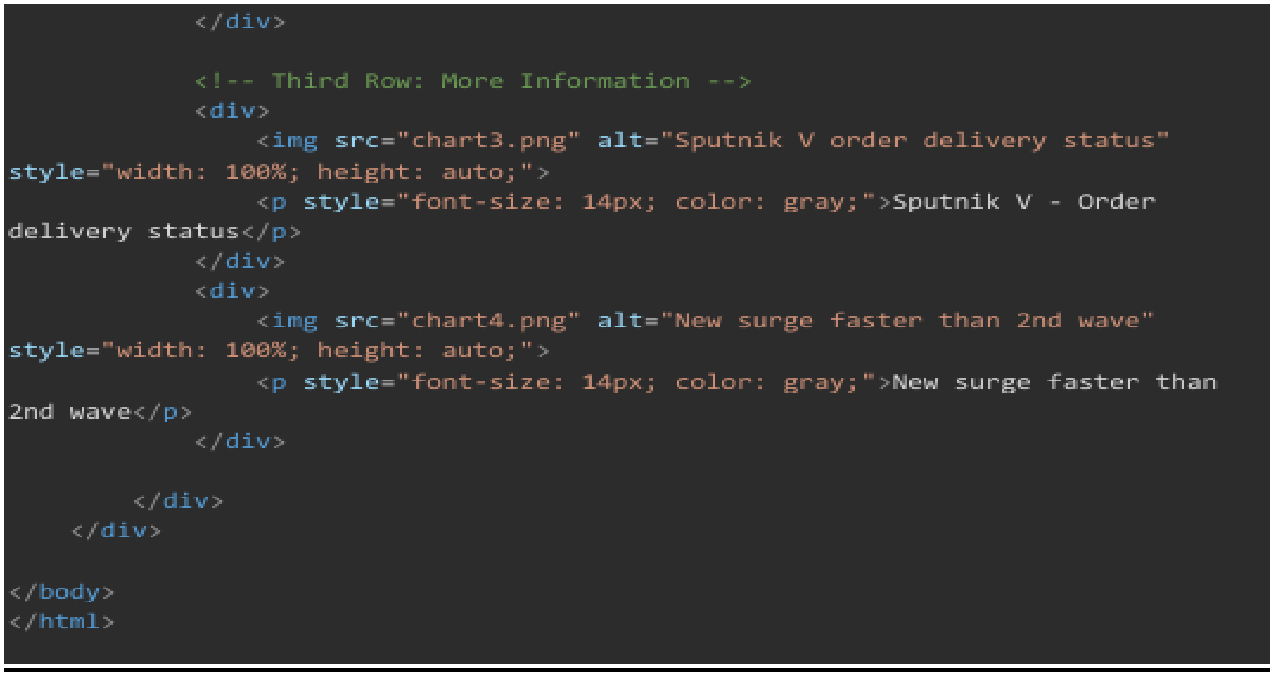
4.Vaccination



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Covi-Sift - Vaccinations</title>

</head>
<body>
    <!-- Navigation Bar -->
    <div style="background-color: #f5f5f5; text-align: center; padding: 10px;">
        <a href="/" style="margin: 0 15px; text-decoration: none; color:
black;">Home</a>
```



```html
        <a href="/about" style="margin: 0 15px; text-decoration: none; color:
black;">About</a>
        <a href="/precautions" style="margin: 0 15px; text-decoration: none;
color: black;">Precautions</a>
        <a href="/vaccination" style="margin: 0 15px; text-decoration: none;
color: #ff5a33; font-weight: bold;">Vaccinations</a>
        <a href="/test" style="margin: 0 15px; text-decoration: none; color:
black;">Test</a>
    </div>

    <!-- Main Content for Vaccinations -->
    <div style="text-align: center; padding: 50px;">
        <h1 style="font-size: 36px; color: #ff5a33;">Vaccinations</h1>
        <hr style="width: 10%; border: 1px solid #ff5a33; margin: 20px auto;">

        <!-- Data Visualizations -->
        <div style="display: grid; grid-template-columns: 1fr 1fr; gap: 20px;
padding-top: 30px;">

            <!-- First Row: Maps -->
            <div>
                <img src="map1.png" alt="Share of adults given at least one dose
by September" style="width: 100%; height: auto;">
                <p style="font-size: 14px; color: gray;">Share of adults given at
least one dose by September</p>
            </div>
            <div>
                <img src="map2.png" alt="Share of adults given both doses by
September 5" style="width: 100%; height: auto;">
                <p style="font-size: 14px; color: gray;">Share of adults given
both doses by September 5</p>
            </div>

            <!-- Second Row: Charts and Graphs -->
            <div>
                <img src="chart1.png" alt="Who is more vulnerable to Covid?"
style="width: 100%; height: auto;">
                <p style="font-size: 14px; color: gray;">Who is more vulnerable
to Covid?</p>
            </div>
            <div>
                <img src="chart2.png" alt="Relentless rise in Covid cases"
style="width: 100%; height: auto;">
                <p style="font-size: 14px; color: gray;">Daily recorded COVID-19
```



```
            </div>

            <!-- Third Row: More Information -->
            <div>
                <img src="chart3.png" alt="Sputnik V order delivery status"
style="width: 100%; height: auto;">
                <p style="font-size: 14px; color: gray;">Sputnik V - Order
delivery status</p>
            </div>
            <div>
                <img src="chart4.png" alt="New surge faster than 2nd wave"
style="width: 100%; height: auto;">
                <p style="font-size: 14px; color: gray;">New surge faster than
2nd wave</p>
            </div>

        </div>
    </div>

</body>
</html>
```

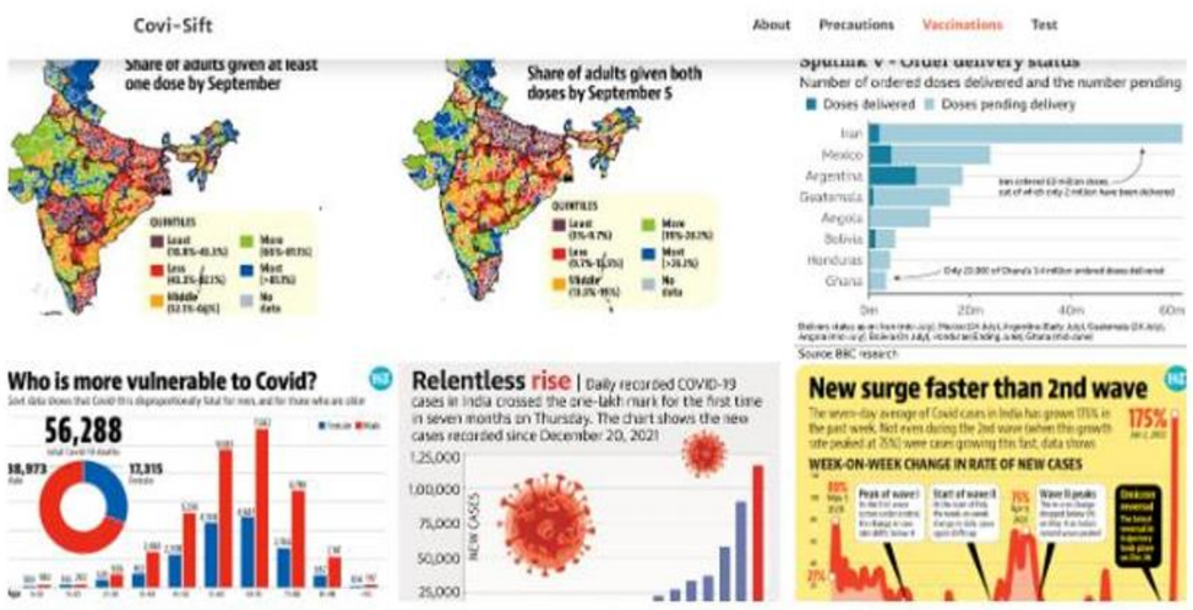
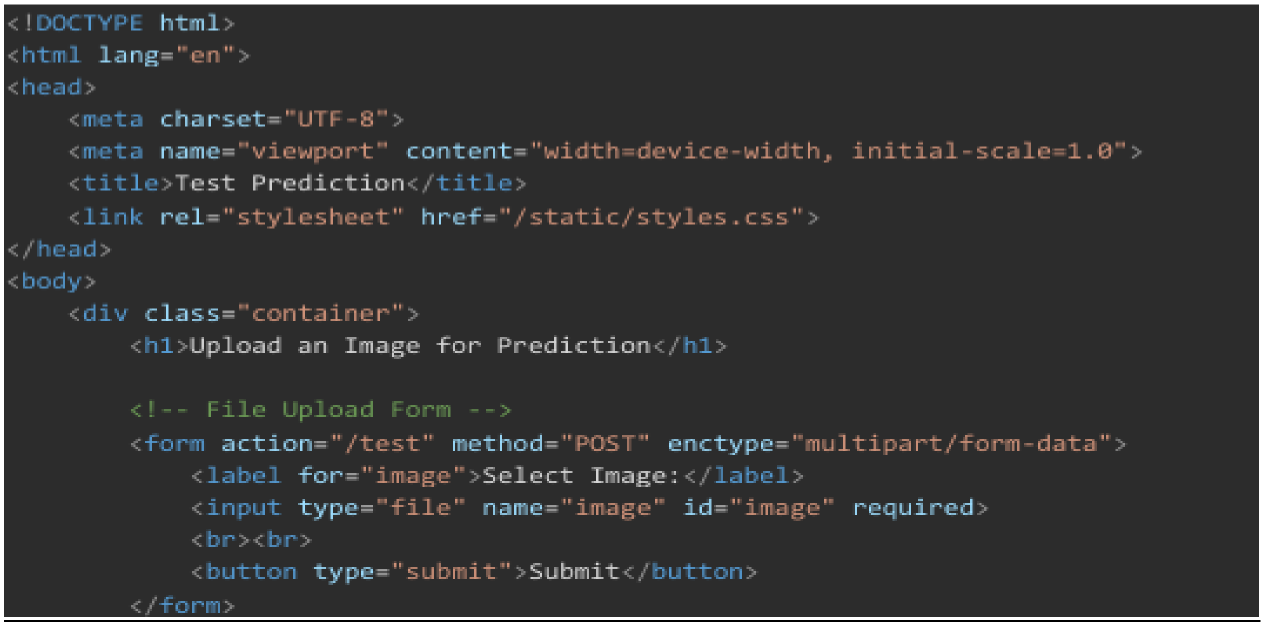
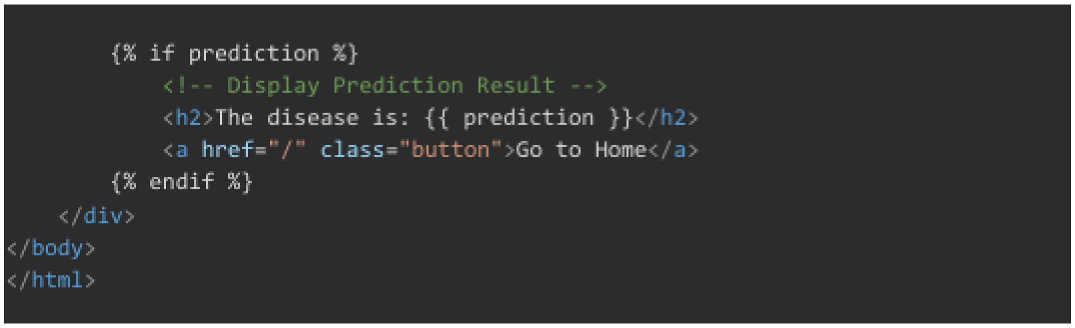

Covi-Sift
About
Precautions
Vaccinations
Test
Share of adults given at least one dose by September
Share of adults given both doses by September 5
Sputnik V - Order delivery status
Number of ordered doses delivered and the number pending
Doses delivered
Doses pending delivery
Who is more vulnerable to Covid?
56,288
Relentless rise
New surge faster than 2nd wave
WEEK-ON-WEEK CHANGE IN RATE OF NEW CASES

Figure12: HTML code for vaccination

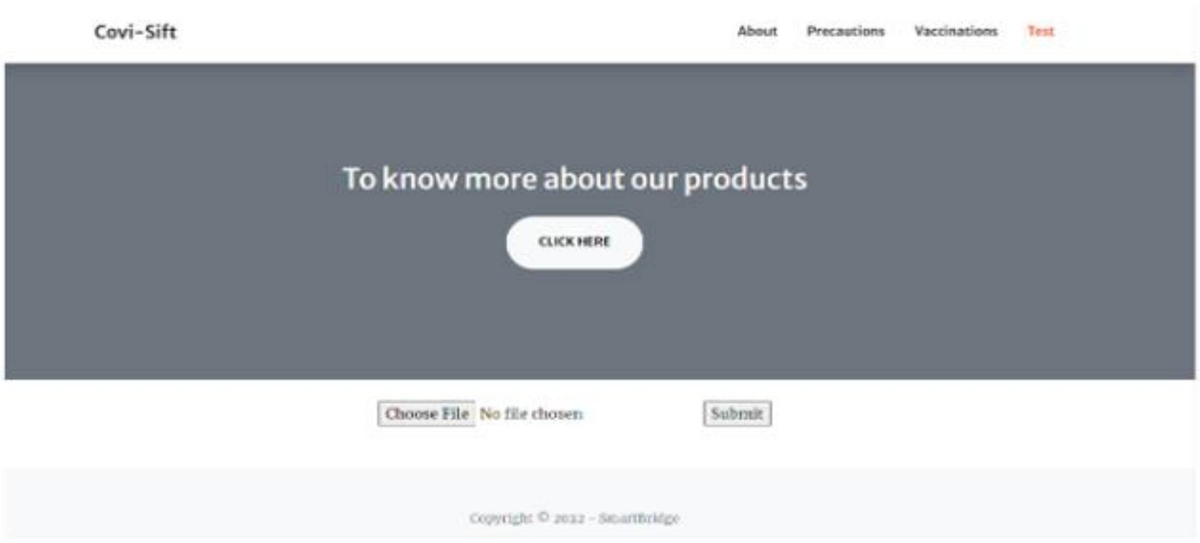
5.Test



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Test Prediction</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <div class="container">
        <h1>Upload an Image for Prediction</h1>

        <!-- File Upload Form -->
        <form action="/test" method="POST" enctype="multipart/form-data">
            <label for="image">Select Image:</label>
            <input type="file" name="image" id="image" required>
            <br><br>
            <button type="submit">Submit</button>
        </form>
```



```
        {% if prediction %}
            <!-- Display Prediction Result -->
            <h2>The disease is: {{ prediction }}</h2>
            <a href="/" class="button">Go to Home</a>
        {% endif %}
    </div>
</body>
</html>
```

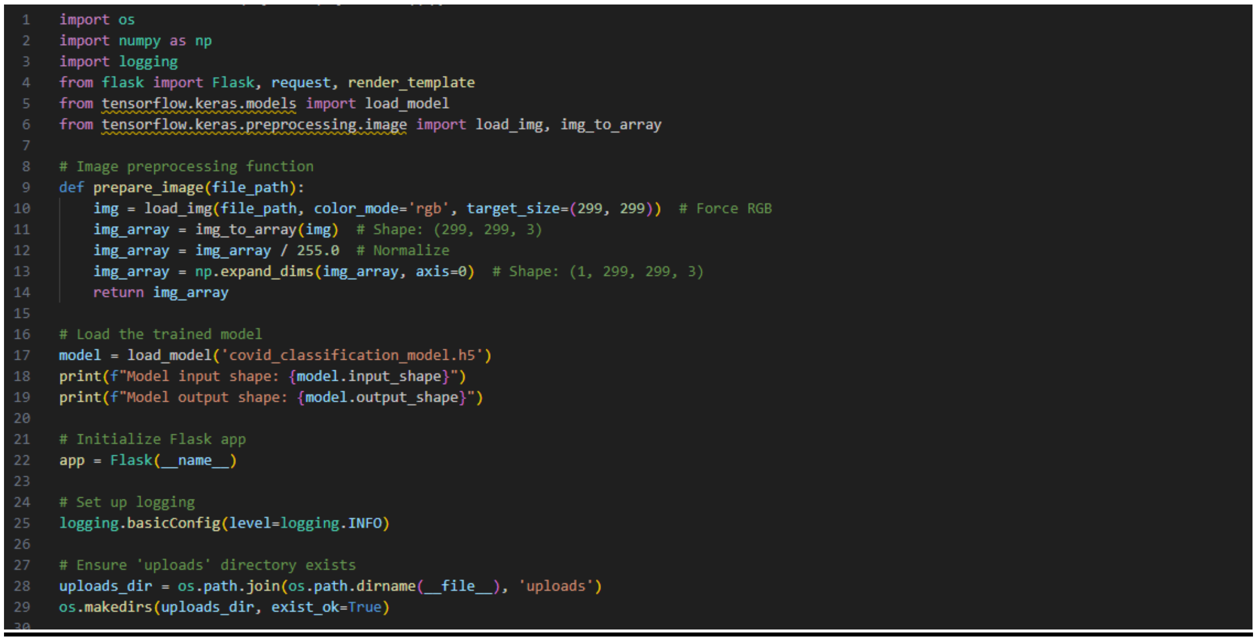


To know more about our products

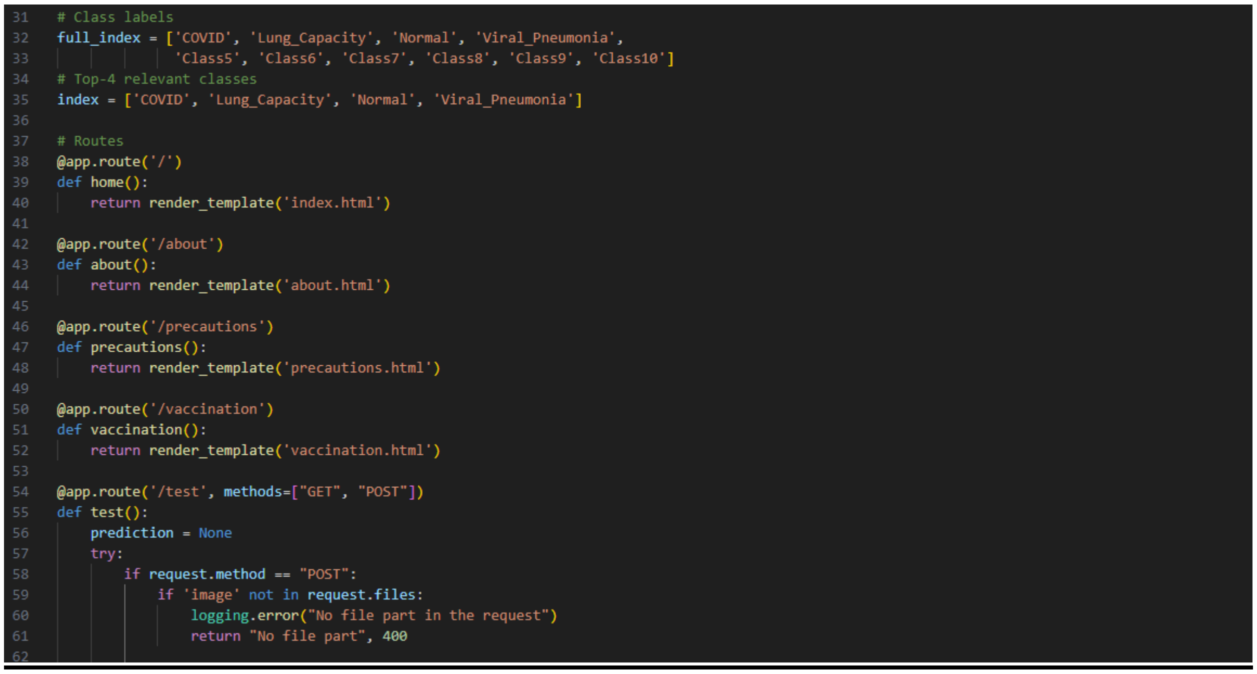
Copyright © 2022 - SmartBridge

Figure13: HTML code for test

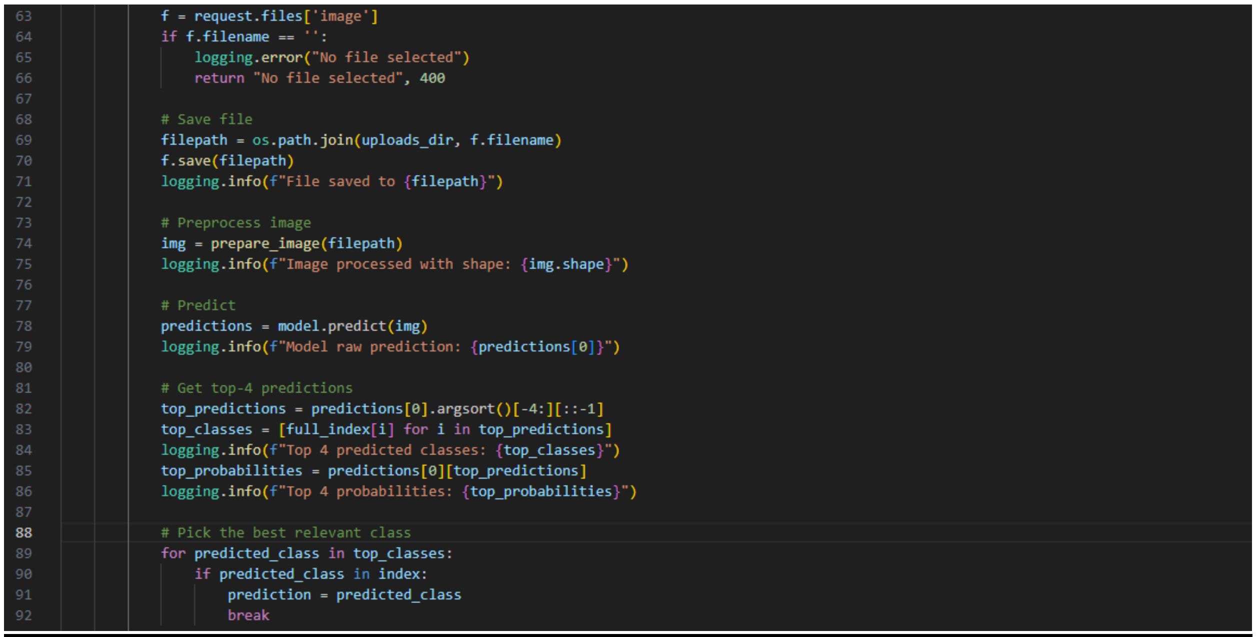
Appy.py



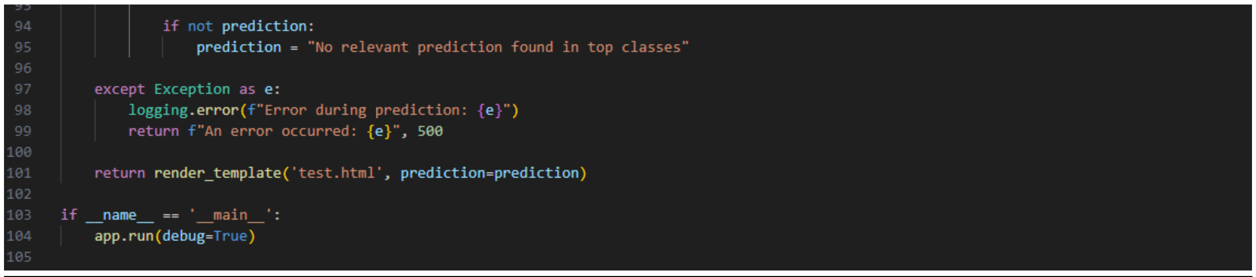
```python
import os
import numpy as np
import logging
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Image preprocessing function
def prepare_image(file_path):
    img = load_img(file_path, color_mode='rgb', target_size=(299, 299))  # Force RGB
    img_array = img_to_array(img)  # Shape: (299, 299, 3)
    img_array = img_array / 255.0  # Normalize
    img_array = np.expand_dims(img_array, axis=0)  # Shape: (1, 299, 299, 3)
    return img_array

# Load the trained model
model = load_model('covid_classification_model.h5')
print(f"Model input shape: {model.input_shape}")
print(f"Model output shape: {model.output_shape}")

# Initialize Flask app
app = Flask(__name__)

# Set up logging
logging.basicConfig(level=logging.INFO)

# Ensure 'uploads' directory exists
uploads_dir = os.path.join(os.path.dirname(__file__), 'uploads')
os.makedirs(uploads_dir, exist_ok=True)
```



```python
# Class labels
full_index = ['COVID', 'Lung_Capacity', 'Normal', 'Viral_Pneumonia',
              'Class5', 'Class6', 'Class7', 'Class8', 'Class9', 'Class10']
# Top-4 relevant classes
index = ['COVID', 'Lung_Capacity', 'Normal', 'Viral_Pneumonia']

# Routes
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/precautions')
def precautions():
    return render_template('precautions.html')

@app.route('/vaccination')
def vaccination():
    return render_template('vaccination.html')

@app.route('/test', methods=["GET", "POST"])
def test():
    prediction = None
    try:
        if request.method == "POST":
            if 'image' not in request.files:
                logging.error("No file part in the request")
                return "No file part", 400

```



```python
            f = request.files['image']
            if f.filename == '':
                logging.error("No file selected")
                return "No file selected", 400

            # Save file
            filepath = os.path.join(uploads_dir, f.filename)
            f.save(filepath)
            logging.info(f"File saved to {filepath}")

            # Preprocess image
            img = prepare_image(filepath)
            logging.info(f"Image processed with shape: {img.shape}")

            # Predict
            predictions = model.predict(img)
            logging.info(f"Model raw prediction: {predictions[0]}")

            # Get top-4 predictions
            top_predictions = predictions[0].argsort()[-4:][::-1]
            top_classes = [full_index[i] for i in top_predictions]
            logging.info(f"Top 4 predicted classes: {top_classes}")
            top_probabilities = predictions[0][top_predictions]
            logging.info(f"Top 4 probabilities: {top_probabilities}")

            # Pick the best relevant class
            for predicted_class in top_classes:
                if predicted_class in index:
                    prediction = predicted_class
                    break
```



```python
            if not prediction:
                prediction = "No relevant prediction found in top classes"

    except Exception as e:
        logging.error(f"Error during prediction: {e}")
        return f"An error occurred: {e}", 500

    return render_template('test.html', prediction=prediction)

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

Figure14: Python code for appy.py

Here we are routing our app to predict function. This function retrieves all the values from the HTML page using Post request. That is stored in variable image and then converted into an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the result.html page earlier.

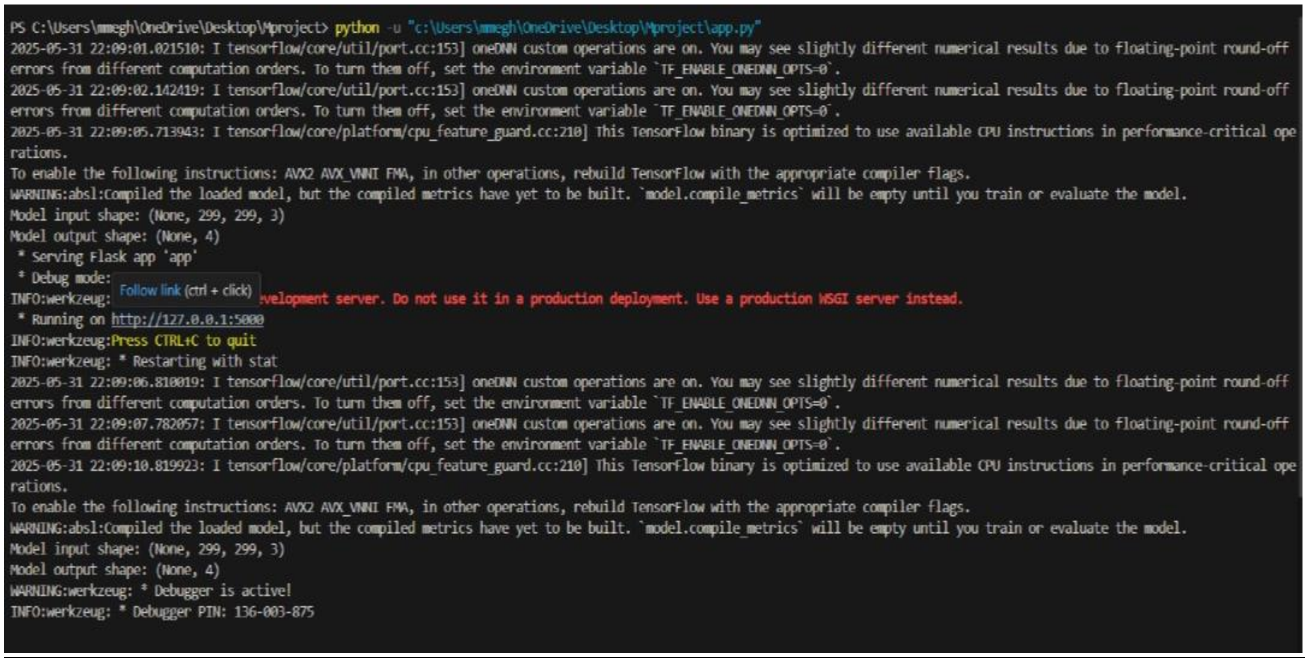
Getting local host in the terminal while running app.py:

To provide a user-friendly interface for detecting covid vision, the COVID-19 lung x-ray includes a web-based or GUI application—typically built using frameworks like Flask, Streamlit, or FastAPI. The script app.py serves as the main entry point for running this application.

Purpose of app.py

The app.py file is responsible for:

- Loading the trained CNN model.
- classify the image as COVID-positive, COVID-negative
- Running real-time inference.
- Displaying detection results

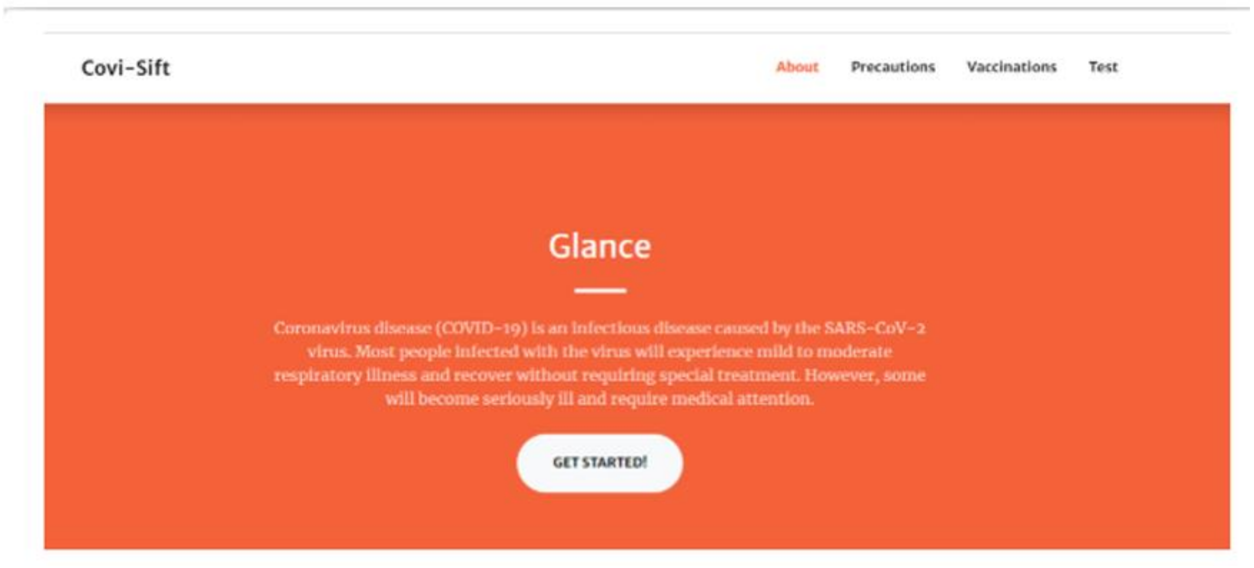


```
PS C:\Users\mmegh\OneDrive\Desktop\Mproject> python -u "c:\Users\mmegh\OneDrive\Desktop\Mproject\app.py"
2025-05-31 22:09:01.021510: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off
errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-05-31 22:09:02.142419: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off
errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-05-31 22:09:05.713943: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical ope
rations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Model input shape: (None, 299, 299, 3)
Model output shape: (None, 4)
 * Serving Flask app 'app'
 * Debug mode: Follow link (ctrl + click)
INFO:werkzeug: evelopment server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
2025-05-31 22:09:06.810019: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off
errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-05-31 22:09:07.782057: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off
errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-05-31 22:09:10.819923: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical ope
rations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Model input shape: (None, 299, 299, 3)
Model output shape: (None, 4)
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 136-003-875
```

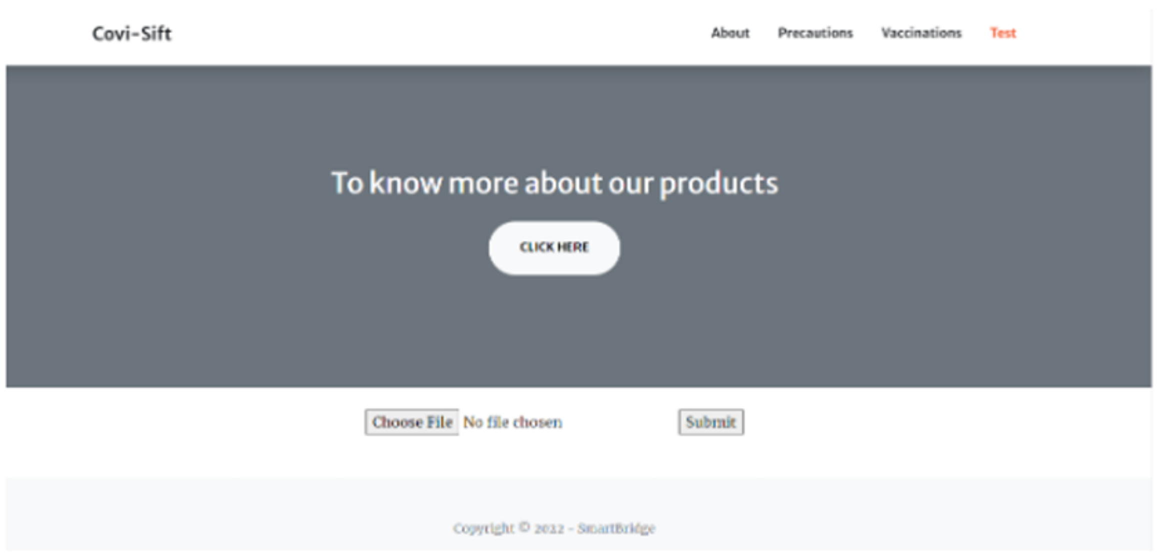
Figure15: Getting localhost on terminal

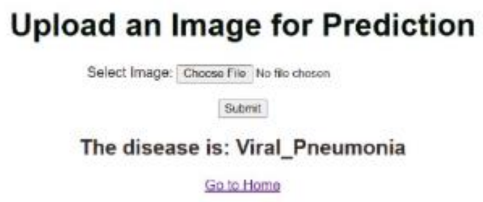
3.RESULT

The COVIDVISION system successfully demonstrated the effectiveness of deep learning for detecting COVID-19 infections from chest X-ray images. The project utilized a Convolutional Neural Network (CNN) architecture deployed and trained using IBM Cloud tools.



Coronavirus disease (COVID-19) is an infectious disease caused by the SARS-CoV-2 virus. Most people infected with the virus will experience mild to moderate respiratory illness and recover without requiring special treatment. However, some will become seriously ill and require medical attention.



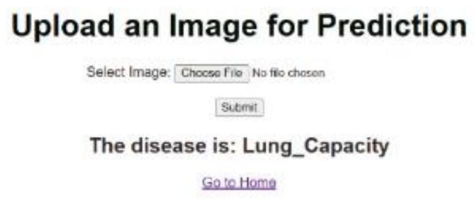

Upload an Image for Prediction

Select Image: Choose File No file chosen

Submit

The disease is: Viral_Pneumonia

Go to Home

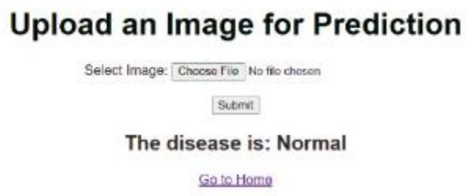

Upload an Image for Prediction

Select Image: Choose File No file chosen

Submit

The disease is: Lung_Capacity

Go to Home


Upload an Image for Prediction

Select Image: Choose File No file chosen

Submit

The disease is: Normal

Go to Home

Figure16: Final output images

4.APPLICATION

1. Deep Learning in Medical Imaging

- Convolutional Neural Networks (CNNs): Widely used for analyzing X-ray and CT scans, particularly for tasks like pneumonia and lung disease detection.

2. COVID-19 Detection from X-Rays

- Datasets: Public datasets (e.g., COVIDx, ChestX-ray8) have been pivotal, though imbalances in data (fewer COVID-19 cases) remain a challenge.

3. Cloud Computing in Healthcare

- Applications: Cloud platforms enable remote diagnostics, large-scale data processing, and integration with telemedicine.

4. Real-World Applications

- Telemedicine and Scalability: Cloud-based systems ensure accessibility in remote and underserved areas.

5.ADVANTAGES

- CovidVision can process and analyze chest X-rays within seconds, allowing for rapid detection of COVID-19.
- This is especially useful in emergency situations or for large-scale screenings.
- Uses deep learning models trained on large datasets, providing high precision and recall.
- Minimizes human error in diagnosis and provides consistent results.
- The application is hosted on IBM Cloud, allowing it to handle thousands of users and image analyses simultaneously.
- Easy to scale up during pandemic spikes or public health crises.
- Medical professionals can access the app from anywhere through a web-based interface, enabling remote diagnosis and telemedicine support.
- Reduces dependency on expensive or slow testing methods like RT-PCR by offering a low-cost screening alternative using existing X-ray machines.

6.DISADVANTAGES

- Chest X-rays may not show clear signs of COVID-19 in early or asymptomatic cases, leading to false negatives.
- Cannot replace RT-PCR as the gold standard for diagnosis—best used as a supplementary too
- Model accuracy depends heavily on the quality and diversity of training data.
- Biases in datasets (e.g., age, region, equipment type) may lead to inconsistent performance across populations.
- Deep learning models often function as “black boxes”, making it difficult for doctors to understand why the model made a specific prediction.
- Lack of transparency can hinder clinical trust.
- Deployment in clinical settings requires regulatory approval, which can be a long and expensive process.
- Handling sensitive medical data raises ethical and legal concerns, especially across borders

7. CONCLUSION

In this Conclusion, I have explored the innovative approach of using deep learning techniques for the detection of COVID-19 from lung X-rays, utilizing IBM Cloud’s powerful computational capabilities. The integration of deep learning models with X-ray imaging offers a promising avenue for rapid and accurate diagnosis, particularly in resource-limited settings where access to PCR testing may be constrained. By leveraging IBM Cloud’s infrastructure, we were able to process and analyze vast datasets of lung X-ray images, training a model capable of distinguishing between COVID-19, pneumonia, and healthy lungs with high precision.

While the results demonstrate significant potential for improving the efficiency and scalability of COVID-19 detection, further research is necessary to refine the model's accuracy and ensure its robustness across diverse patient demographics and imaging conditions. The potential to scale this solution globally is immense, and as technology advances, such deep learning models could serve as a pivotal tool in the ongoing fight against pandemics.

In conclusion, **CovidVision** represents a significant step forward in integrating AI with healthcare solutions, enabling quicker, cost-effective, and potentially life-saving diagnoses. Future developments will focus on expanding the dataset, refining model performance, and integrating this solution into clinical workflows, ultimately contributing to global health resilience.

8. FUTURE SCOPE

Implementing explainable AI techniques will allow radiologists and medical staff to better understand the model's decision-making, increasing trust and adoption in clinical environments. Deployment of the model on mobile devices or edge devices (like Raspberry Pi or NVIDIA Jetson) can make diagnostics accessible in remote or under-resourced areas without requiring constant internet or cloud access. Utilizing IBM Cloud's advanced services such as Watson Machine Learning and AutoAI can help in auto-tuning models, managing large-scale deployments, and supporting millions of X-ray evaluations per day. Beyond detection, COVIDVISION can evolve to include patient history analysis and predictive modeling to forecast disease progression or re-infection risks. For real-world use, future work must focus on clinical validation and obtaining approvals from health authorities like the FDA or WHO, ensuring the tool is ready for large-scale deployment in healthcare systems.

9.BIBLIOGRAPHY

[1] "COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest Radiography Images" by Linda Wang et al. (Published in IEEE Transactions on Medical Imaging, 2020).

[2] "COVID-19 Detection from X-Ray Images Using Deep Learning and Convolutional Neural Networks" by Ozturk et al. (Published in arXiv, 2020).

[3] "COVID-19 detection in chest X-ray images using deep learning" by Muhammad Usama et al. (Published in Cognitive Systems Research, 2021).

[4] "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.

[5] "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron.

10.HELP FILE

PROJECT EXECUTION:

STEP-1: Go to start, search and launch ROBOFLOW.

STEP-2: After launching of ROBOFLOW, launch GOOGLE COLAB.

STEP-3: Open “best.py” code

STEP-4: Import all the packages and check whether error present in the code are not.

STEP-5: Create PYTHON CODE folder on DESKTOP.

STEP-6: Create the home.html file to display the home page.

STEP-7: Launch the VSCODE.

STEP-8: After launching Spyder, give the path of best.py which is created in your laptop and run the program.

STEP-9: After running the app.py, then the URL is created “localhost:5000”.

STEP-10: Copy the URL and paste it in the Web Browser.

STEP-11: Then the home page of the project will be displayed.

STEP-12: In the opened home page when click on predict button it will show upload an image of Lung X-ray.

STEP-13: After uploading image of Lung x-ray as COVID-positive, COVID-negative