### Smart city Waste Management System with connected Trash Cans

### A PROJECT REPORT

***Submitted by***

**SOBI AMIRTHA N 811220106039**

**SUNIL KUMAR B 811220106041**

**SUSILA N 811220106042**

***in partial fulfillment for the award of the degree***

***of***

## BACHELOR OF ENGINEERING

### IN

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**INDRA GANESAN COLLEGE OF ENGINEERING, TRICHY**

## ANNA UNIVERSITY: CHENNAI-600025

**MAY 2023**

## TABLE OF CONTENTS

|  |  |
| --- | --- |
| **CHAPTER NO** | **TITLE** |
| **1** | **INTRODCUTION** |
|  | 1.1. Project Overview |
|  | 1.2. Purpose |
| **2** | **IDEATION & PROPOSED SOLUTION**  2.1.Problem statement Definition 2.2.Empathy Map Canvas 2.3.Ideation & Brainstorming  2.4.Proposed Solution |
| **3** | **REQUIREMENT ANALYSIS** |
|  | 3.1.Functional Requirement |
|  | 3.2.Non Functional Requirements |
| **4** | **PROJECT DESIGN** |
|  | 4.1. Data Flow Diagrams |
|  | 4.2.Solution & Technical Architecture |
|  | 4.3.User Stories |
| 5 | **CODING & SOLUTIONING**   * 1. Feature 1   2. Feature 2   3. Database Schema |

|  |  |
| --- | --- |
| **6** | **RESULTS** |
|  | 6.1. Performance Metrics |
| 7 **ADVANTAGES & DISADVANTAGES** | |
| 8 **CONCLUSION** | |
| 9 **FUTURE SCOPE** | |
| 10 **APPENDIX**  Source Code  GitHub & Project Video Demo Link | |

**INTRODUCTION**

### PROJECT OVERVIEW:

Smartcity waste management system The main goal of this project was to create upload to clean the cityimage itself, which allows user to login and upload. Moreover,the

project is also designed in such a way it lets detecting the report information. user can upload here , and they will be get the report result.In today’s busy world, people don’t

have time for their personal needs. And the technology fast that anyone can do by

sitting in a room. If someone buy a new thing, he can know the information of x-ray

report online with the help of Internet. Waste management is an techniques like Inception V3,Resnet50,Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

### SCOPE:

The main scope of this system is to identify the type of covide. This system

could efficiently classify many number so proof .

This system is useful for predicting covid in efficient manner. It can be

achieved by comparing the images with the trained dataset. It helps to predict type of

covid with an improved accuracy.

It is a user-friendly interface to interact with the system. Once algorithm is

proposed error free classification is done. This system requires less manpower and

small amount of time for the covid identification.

This system helps to predict the coviid by preprocessing the images even it is.

Covid is identified and notified to the user

# IDEATION & PROPOSED SOLUTION

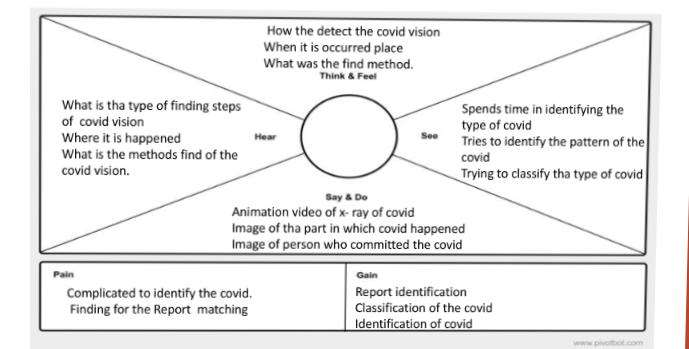
### PROBLEM STATEMENT DEFINITION :

User from the department of covid vision.The imaging tool, such as objective is to

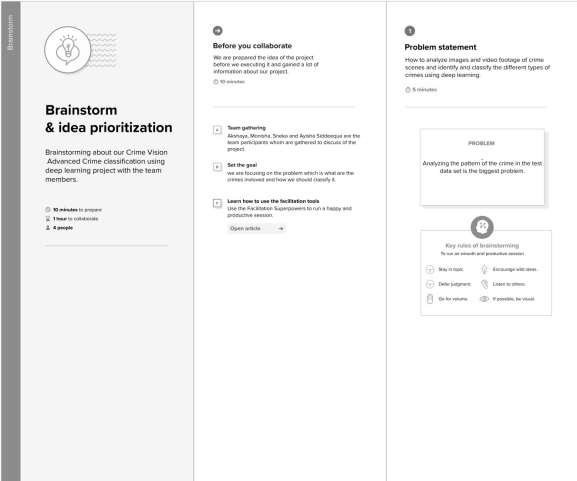
develop an automated CAD system for the detection of waste management samples from

healthy and pneumonia cases using CXR images.

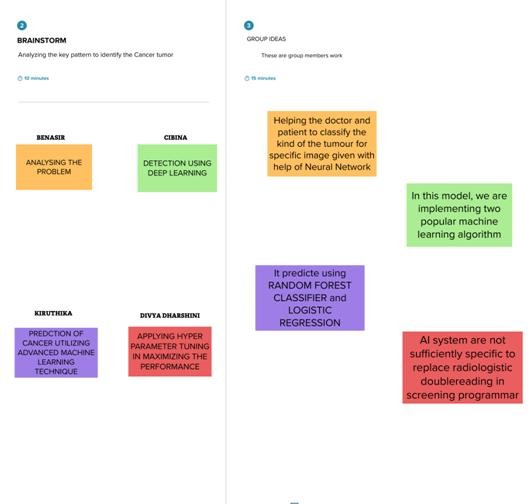
**EMPATHY MAP CONVAS :**

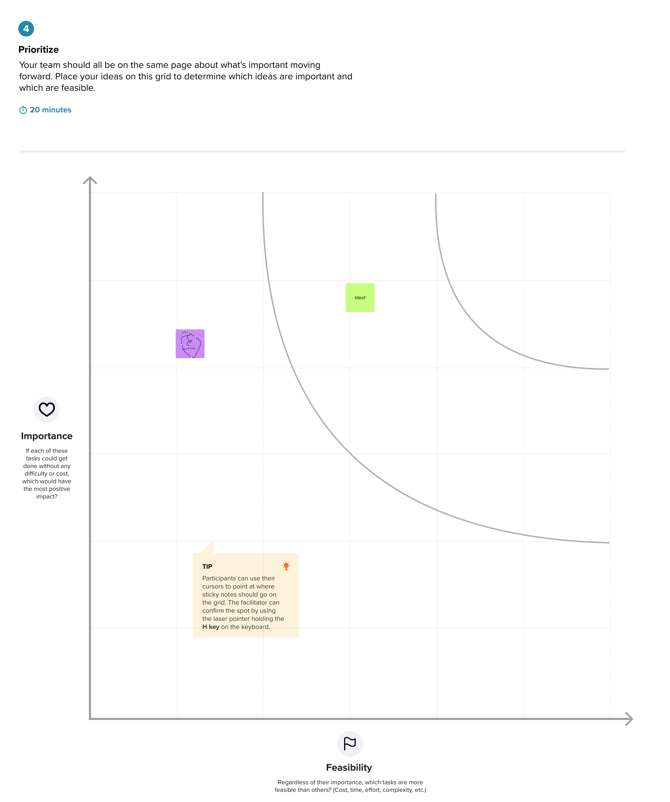
**IDEATION & BRAINSTORMING:**

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



**Step-2: Brainstorm, Idea Listing and Grouping**



**Step-3:Idea Prioritization**

ANALYSIS THE REPORT

CAD SYSTEM REPORT

USING TOOLS

# PROPOSED SOLUTION:

The proposed system is working on various algorithms and based on that it proposes that which algorithm is best to predict the breast cancer. The proposed system is

This paper presented the techniques and methods that can be used to predict

. The proposed system uses deep learning technique for identifying the type

of . This method helps in solving classification and forecasting problems.

They have also been shown to be a useful tool for working with big data oriented

environments such as law enforcement. The convolutional neural network

algorithm is used to predict the type of covid. Using convolutional neural network

algorithm in deep learning technique can substantially impact the overall

functionality. These systems predict covid with improved accuracy.

It required less manpower for predicting the type of cov

id. It automatically

predicts and classifies the covid to the datasets respectively. It takes less time for

predicting the crime. It is an efficient system for covid prediction.It can provide

results much faster than PCR tests, which can take days to process.

The proposed model yielded 96.01% sensitivity, 96.20% precision, and 98.00%

accuracy for the Radiography Dataset while achieving 97.84%

accuracy, 96.76% sensitivity and 96.80% precision, for the Covid-ChestX-ray-15k

dataset.The proposed deep learning CAD system can reliably differentiate COVID-

19 from other respiratory diseases The proposed CAD system was trained with an

annotated training set of 50,490 chest X-ray images.respiratory diseases:

atelectasis, infiltration, pneumothorax, masses, effusion, pneumonia, cardiomegaly,

and nodules.10

**Information**:

The deep learning paradigm has been applied vastly to investigate

radiographic images such as Chest X-Rays (CXR) and CT scan images. These

radiographic images are rich in information such as patterns and clusters like

structures, which are evident in conformance and detection of COVID-19 like

pandemics

**Time saving:**

This advantage of covid 19 detection is very obvious. The time necessary to

visit an online and upload here, looking or particular result is an average of within

the second. considering the upload and checking the sometime results. The time to

visit a conventional detection covid x-ray depends on many factors, but it will

suffice to say that it is not in the comparable margins

# REQUIREMENT ANALYSIS

**FUNCTIONAL REQUIREMENTS:**

### SOFTWARE REQUIREMENTS

 Operating system : Windows 10 OS

 Front End : Keras,Numpy,Tensorflow,Pandas

 Back End : Python 3.7 version, Anaconda software,

Pycharm,Deep Learning-Framework

 NumPy

 Matplotlib

 Keras

 Tensorflow

**4.2. HARDWARE REQUIREMENTS**

 Processor : AMD PRO A4-4350B

 Hard disk : 500 GB

 Keyboard : Standard keyboard

 Monitor : 15-inch color monitor

# NON\_FUNCTIONAL REQUIREMENTS:

### SECURITY:

Deep learning allows you the user to quickly add common security mechanisms to your Flask application.

### SCALABLE:

Flask is highly scalable as it can process a high number of requests each day.

This micro-framework modularizes the entire code and let developers work on independent chunks and use them as the code base grows

### PERFORMANCE:

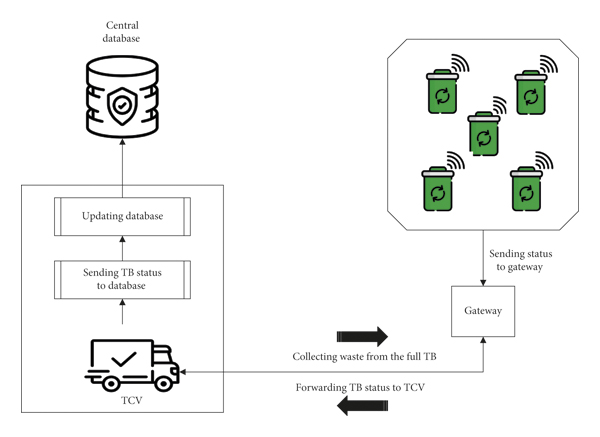
Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc.

### AVAILABILITY

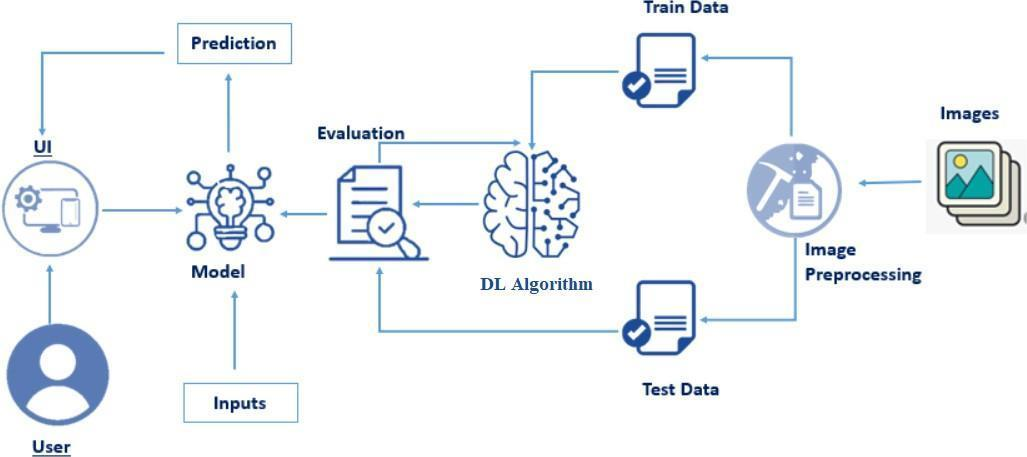
Justify the availability of application (e.g. use of load balancers, distributed servers etc.)

# PROJECT DESIGN

**DATA FLOW DIAGRAM:**



# SOLUTION & TECHNICAL ARCHITECTURE:



# CODING & SOLUTIONING

**FEATURE 1:**

Classification results for feature set #9.

The ROC curves of feature subsets #4, #5 and #9 for best performing classifier (ANN) are given in Figure 4(C), (D), (E).Figure 4(F), displays the ROC for feature set #5 when subjected to quadratic SVM for RH validation protocol.

As it is observed from different tables (Table 5 to Table 7), the ANN classifier with 10-fold cross validation protocol possesses the highest values of accuracy and AUC (Both > 0.95).

Therefore, it may be affirmed that ANN gives the best classifier model for WBCD database when top 4 and top 5 features based on their dominance rank and otherwise all 9 features are considered. Also, the average value of accuracies of all feature sets (feature set #1 to #9) followed incremental order (95.9%, 96.8% and 97.7%) for ANN when the validation protocol shifted from RH to 5- fold to 10-fold whereas for all other classifiers it was almost constant (around 95.5%). Accuracies of ANN classifier for three data division protocols (RH, 5-fold, 10-fold) is depicted .

# FEATURE 2:

Receiver operating characteristics curves.

* + 1. ANN and (B) Naïve Bayes classifiers with RH protocol for feature set #1. (C), (D) and (E) ANN classifier for feature sets #4, #5 and #9 with 10-fold

cross validation protocol.

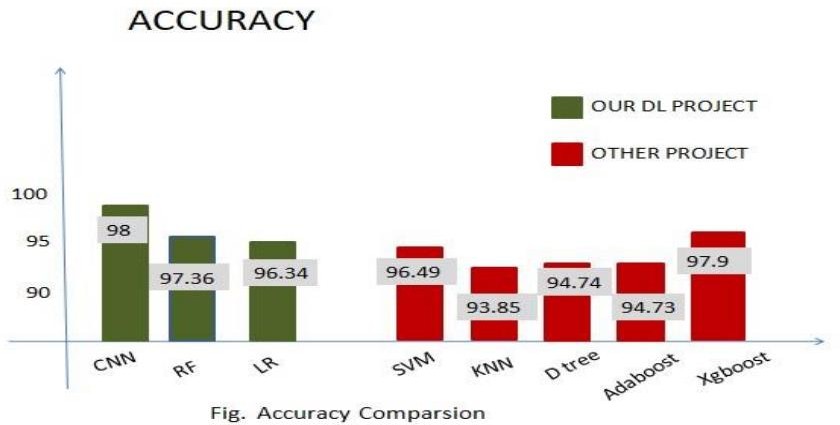
* + 1. (F) Quadratic SVM classifier with RH protocol for feature set #5.

Evaluating feature sets #4 and #5 (top 4 & 5 dominant feature sets), the ANN clearly outperformed all other classifiers with highest classification accuracies 98.9% and 99.6% respectively for 10-fold cross validation protocol.The AUC values for the above cases are 0.986 and 0.995 respectively. However, for feature set #5 the quadratic SVM classifier obtained the highest classification accuracy and AUC namely 97% and 0.967 if subjected to RH validation protocol. Tables 5 and Table 6 display the results in detail along with the values of sensitivities and specificities.

# DATABASE SCHEMA:

Dataset helps you to organize unstructured data collected from multiple sources to get the target outcome. Initial data that you give to an algorithm for learning is usually called a training dataset. Training data is a foundation for further development that determines how effective and useful your Machine Learning system will be.

# RESULT PERFORMANCE METRICS:



We used Google Colab and Microsoft Visual Code as a Coding platform and get a prediction output from the Flask in Local Server. Our Methods

Includes Supervised Learning Algorithms and Classification Techniques like Support Vector Classifier (SVM), Random Forest, Naïve Bayes, Decision Tree, KNN, Adaboost and XGboost. Dataset contains features which highly vary in units and magnitudes. So, it is required to bring all features to the same level of magnitudes. We did that by using Standard Scaling in SKLearn.

Model selection is the most important step in Deep Learning. Deep Learning algorithms can be classified as: supervised learning and unsupervised learning. For Our project, we only need supervised learning. We used all Methodologies to predict the result and noted their Accuracy.47 With other research paper we can see that among Naïve Bayes, Support

Vector Machine, Adaboost, KNN, Decision Tree, XGboost etc. We concluded that CNN, Random Forest, Logistic Regression is the most accurate algorithm for best accurate result for prediction of breast cancer with the efficiency of 98%.

# ADVANTAGES & DISADVANTAGES

**ADVANTAGES:**

The usages of CNNs are motivated by the fact that they can capture / are able to learn relevant features from an image /video at different levels similar

to a human brain.This is feature learning.

In terms of performance, CNNs outperform NNs on conventional image recognition tasks and many other tasks.

For a completely new task / problem CNNs are very good feature extractors. This means that we can extract useful attributes from an already trained CNN with its trained weights by feeding your data on each level and tune the CNN a bit for the specific task.

E.g.: Add a classifier after the last layer with labels specific to the task. This is also called pertaining and CNNs are very efficient in such tasks compared to NNs.

# DISADVANTAGES:

### HARDWARE DEPENDENCE:

Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent.

### UN EXPLAINED BEHAVIOUR OF NETWORK:

This is the most important problem of ANN. When ANN produces a

probing solution, it does not give a clue as to why and how. This reduces trust in the network.

### DETERMINATION OF PROPER NETWORK STRUCTURE:

There is no specific rule for determining the structure of artificial neural networks.

Appropriate network structure is achieved through experience and trial and error.

### DIFFICULTY OF SHOWING PROBLEM TO NETWORK:

ANNs can work with numerical information. Problems have to be

translated into numerical values before being introduced to ANN. The display mechanism to be determined here will directly influence the performance of the network. This depends on the user's ability.

### THE DURATION OF NETWORK IS UNKNOWN:

The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum Results.

# CONCLUSION:

Advanced COVID-19 detection from lungs X-ray with deep learning or

machine learning has the potential to revolutionize the way we diagnose and

treat COVID-19.

It provides a faster and more efficient way of detecting COVID-19,

especially in areas with limited access to testing facilities.

With continued research and development, advanced COVID-19 detection

from lungs X-ray could becomean indispensable tool in the fight against

COVID-19.

**9.2. FUTURE ENHANCEMENT**

The scope of the project includes that what all future enhancements can

be done in this system to make it more feasible to us: -

• Databases for different report result range and storage can be provided.

• Multilingual support can be provided so that it can be

understandable by the person of any language.

More graphics can be added to make it more user-friendly and understandable.

• Manage & backup versions of documents online

# APPENDIX

**SOURCE CODE:**

**TIRE22.PY:**

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications.vgg16 import preprocess\_input, decode\_predictions

import numpy as np

import matplotlib.pyplot as plt

import time

import cv2

import os

os.environ["KMP\_DUPLICATE\_LIB\_OK"]="TRUE"

#load saved model

model = load\_model('best\_resnet1.h5')

#image pass to test -----------

img\_path = '111.jpeg'

img = image.load\_img(img\_path, target\_size=(224, 224))

img1=cv2.imread(img\_path)

img22=cv2.resize(img1,(100,100))

gray\_image = cv2.cvtColor(img22, cv2.COLOR\_BGR2GRAY)

canny\_edge= cv2.Canny(gray\_image,200,200)

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

x = preprocess\_input(x)

preds=model.predict(x)# create a list containing the class labels

class\_labels = ['covid','normal ']

# find the index of the class with maximum score

pred = np.argmax(preds, axis=-1)

print("resnet prediction",preds)

pred = np.argmax(preds, axis=-1)

print(preds)

predict\_value=preds\*100

print("value percent =",predict\_value)

# print(class\_labels[pred[0]])

# print(pred)

cv2.imshow("canny",canny\_edge)

cv2.imshow("img",img22)

cv2.waitKey(1)

if preds<0.05:

print("output result = covid")

**#** plt.title('covid')

# plt.imshow(img)

# # plt.show()

# gray\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# cv2.imshow("gray", gray\_image)

# x = ["prediction"]

# y = [float(preds)]

# plt.xlabel('Prediction')

# plt.ylabel('Values')

# plt.title('covid')

# plt.bar(x, y)

# plt.show()

elif preds>0.05:

print("output result = normal")

plt.title('normal')

plt.imshow(img)

plt.show()

# # cv2.imshow("gray", gray\_image)

# x = ["prediction"]

# y = [float(preds)]

# plt.xlabel('Prediction')

# plt.ylabel('Values')

# plt.title('diseased leaf')

# plt.bar(x, y)

# plt.show()

# print the label of the class with maximum score

# print(class\_labels[pred[0]])

#

#

**RESNET\_TRAIN11.PY:**

import cv2

import gradio as gr

import matplotlib.pyplot as plt

import numpy as np

import random

import pandas as pd

import tensorflow as tf

from skimage import transform

from sklearn.metrics import classification\_report, confusion\_matrix, ConfusionMatrixDisplay

from tensorflow.keras.applications.resnet import ResNet50, preprocess\_input

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from tensorflow.keras.layers import Dense, Dropout, Flatten

from tensorflow.keras.models import Model

np.random.seed(42)

tf.random.set\_seed(42)

random.seed(42)

image\_size = (224, 224)

input\_shape = image\_size + (3,)

num\_classes = 2

batch\_size = 32

data\_dir = "dataset/"

train\_ds = tf.keras.utils.image\_dataset\_from\_directory(

data\_dir,

shuffle=True,

validation\_split=0.3,

subset="training",

label\_mode="binary",

seed=123,

image\_size=image\_size,

batch\_size=batch\_size)

val\_ds = tf.keras.utils.image\_dataset\_from\_directory(

data\_dir,

shuffle=True,

validation\_split=0.3,

subset="validation",

label\_mode="binary",

seed=123,

image\_size=image\_size,

batch\_size=batch\_size)

val\_batches = tf.data.experimental.cardinality(val\_ds)

test\_ds = val\_ds.take((2\*val\_batches) // 3)

val\_ds = val\_ds.skip((2\*val\_batches) // 3)

print(f"Using : {len(np.concatenate([i for x, i in test\_ds], axis=0))} files for testing")

data\_temp = tf.keras.utils.image\_dataset\_from\_directory(

data\_dir,

batch\_size=5,

shuffle=False,

seed=123,

)

class\_names = data\_temp.class\_names

fig, axes = plt.subplots(5, 5, figsize=(10, 10))

axes = axes.flatten()

for i, cls in enumerate(class\_names):

j = 0

for img, label in data\_temp:

if label[0] == i:

ax = axes[i \* 5 + j]

ax.imshow(img[0].numpy().astype("uint8"))

ax.set\_title(cls)

ax.axis('off')

j += 1

if j == 5:

break

plt.tight\_layout()

plt.show()

inputs = tf.keras.Input(shape=input\_shape)

x = tf.keras.layers.Lambda(preprocess\_input)(inputs)

resnet\_model = ResNet50(weights='imagenet',

include\_top=False,

input\_shape=input\_shape,

input\_tensor=x)

for layer in resnet\_model.layers:

layer.trainable = False

x = tf.keras.layers.GlobalAveragePooling2D()(resnet\_model.output)

x = tf.keras.layers.Dense(256, activation='relu')(x)

x = tf.keras.layers.Dropout(0.5)(x)

predictions = tf.keras.layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.models.Model(inputs=inputs, outputs=predictions)

optimizer = tf.keras.optimizers.Adam(learning\_rate=0.001)

model.compile(loss = 'binary\_crossentropy',

optimizer = optimizer,

metrics = ['accuracy'])

checkpoint\_path = "best\_resnet1.h5"

model\_checkpoint = ModelCheckpoint(

checkpoint\_path,

save\_best\_only=True,

monitor='val\_accuracy',

mode='max',

verbose=1

)

early\_stopping = EarlyStopping(monitor='val\_accuracy', patience=5)

history = model.fit(train\_ds, validation\_data=val\_ds, epochs=20, callbacks=[model\_checkpoint, early\_stopping])

history\_df = pd.DataFrame(history.history)

# history\_df.to\_csv("history/history\_resnet.csv", index=False)

best\_model = tf.keras.models.load\_model("best\_resnet1.h5")

print("training completed")

**INCEP11.PY:**

import os

import cv2

import glob

import pandas as pd

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from keras.layers import Dense

from keras.models import Sequential

from keras.preprocessing import image

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import Model

from keras.callbacks import ReduceLROnPlateau

from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.inception\_v3 import InceptionV3

from tensorflow.keras.applications.inception\_v3 import preprocess\_input

from keras.layers import Convolution2D,Dense,MaxPool2D,Activation,Dropout,Flatten

from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D

def get\_files(directory):

if not os.path.exists(directory):

return 0

count = 0

# crawls inside folders

for current\_path, dirs, files in os.walk(directory):

for dr in dirs:

count += len(glob.glob(os.path.join(current\_path, dr + "/\*")))

return count

train\_dir = "covid\_lung\_resnet/train"

test\_dir = "covid\_lung\_resnet/val"

train\_samples =get\_files(train\_dir)

#to get tags

num\_classes=len(glob.glob(train\_dir+"/\*"))

#test file image count

test\_samples=get\_files(test\_dir)

print(num\_classes,"Classes")

print(train\_samples,"Train images")

print(test\_samples,"Test images")

train\_datagen=ImageDataGenerator(

rescale=1./255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True

)

test\_datagen=ImageDataGenerator(rescale=1./255)

input\_shape=(224,224,3)

train\_generator =train\_datagen.flow\_from\_directory(train\_dir,target\_size=(224,224),batch\_size=32)

test\_generator=test\_datagen.flow\_from\_directory(test\_dir,shuffle=True,target\_size=(224,224),batch\_size=32)

IMAGE\_SIZE = [224, 224]

inception = InceptionV3(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

# We don't need to train existing weights

for layer in inception.layers:

layer.trainable = False

# Model layers -> can add more if required

x = Flatten()(inception.output)

prediction = Dense(num\_classes, activation='softmax')(x)

# Create a model object

model = Model(inputs=inception.input, outputs=prediction)

# View the structure of the model

model.summary()

validation\_generator = train\_datagen.flow\_from\_directory(

test\_dir,

target\_size=(224, 224),

batch\_size=32)

model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

history4 = model.fit(

train\_generator,#egitim verileri

steps\_per\_epoch=None,

epochs=10,

validation\_data=validation\_generator,

validation\_steps=None,

verbose=1,

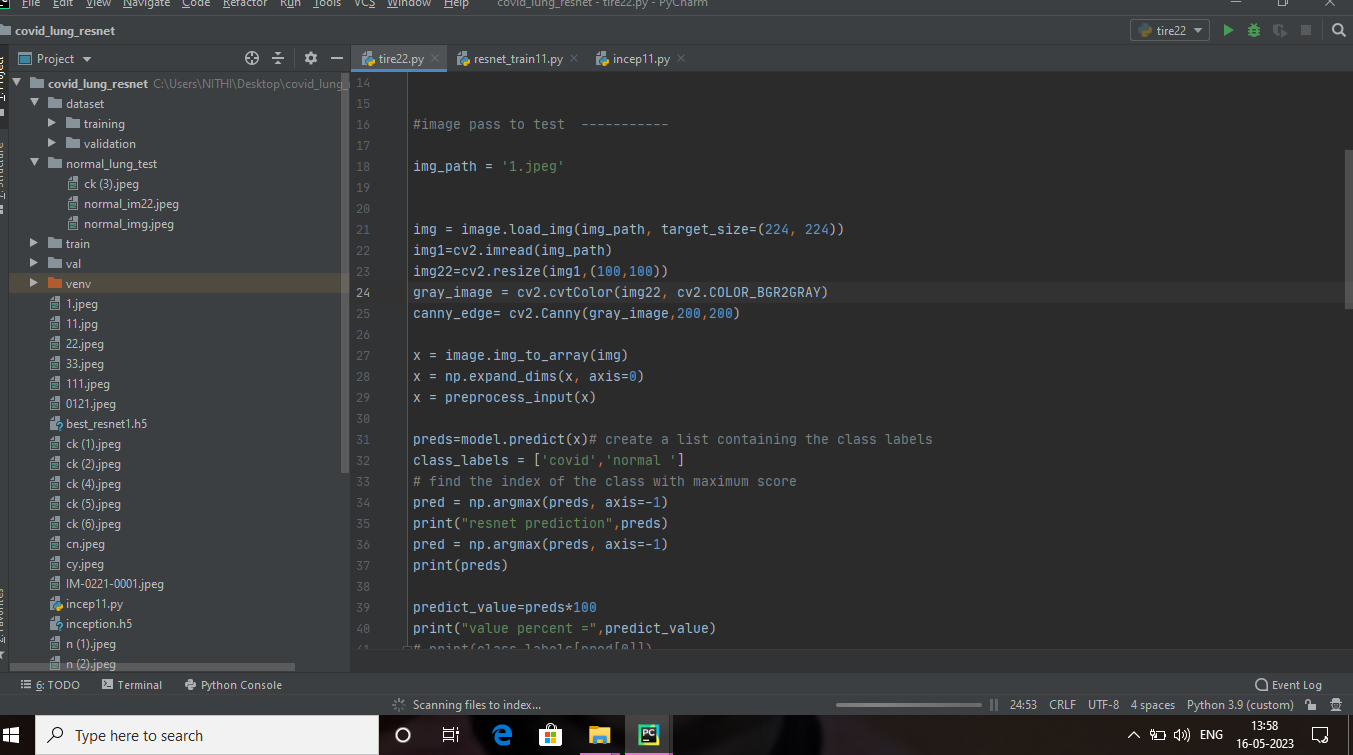
callbacks=[ReduceLROnPlateau(monitor='val\_loss', factor=0.3,patience=3, min\_lr=0.000001)],

shuffle=True)

model.save('inception.h5')

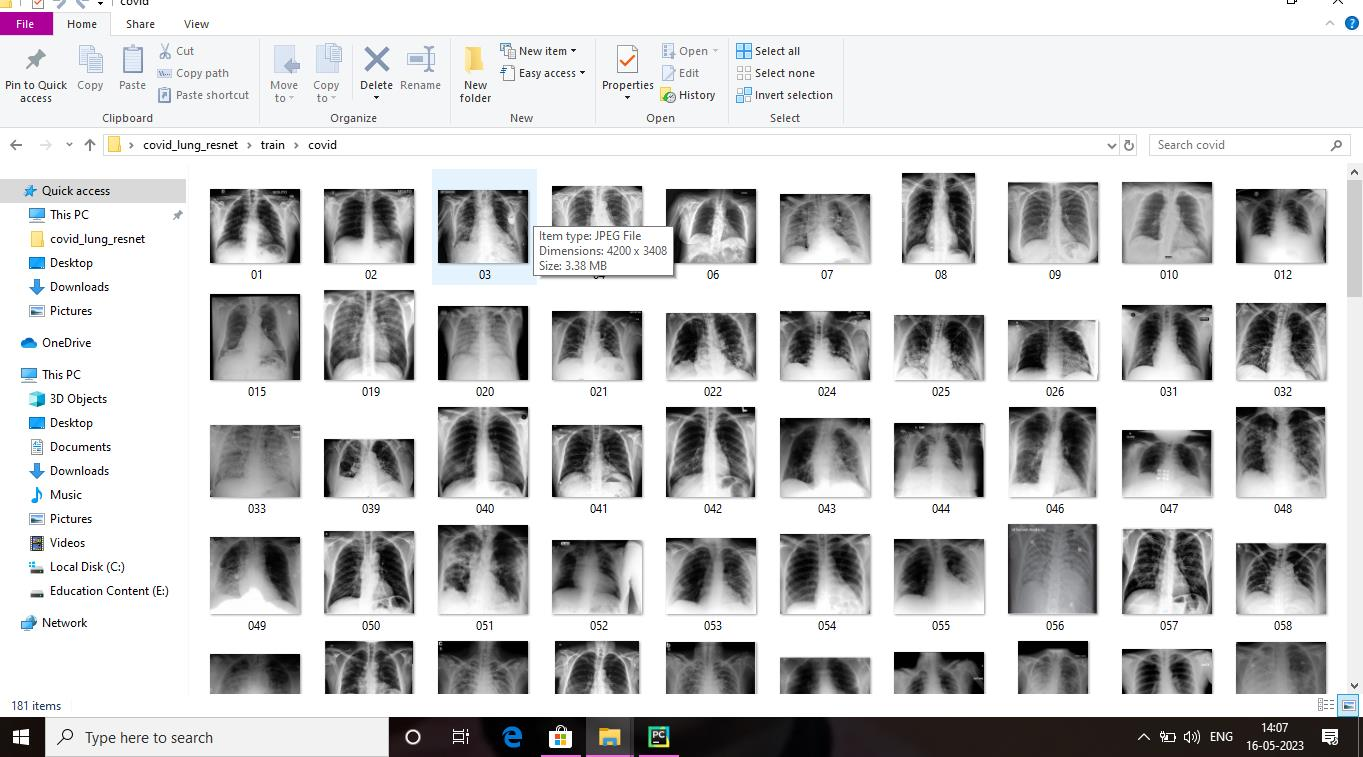
# SCREENSHOTS:

**IMAGE UPLOAD:**

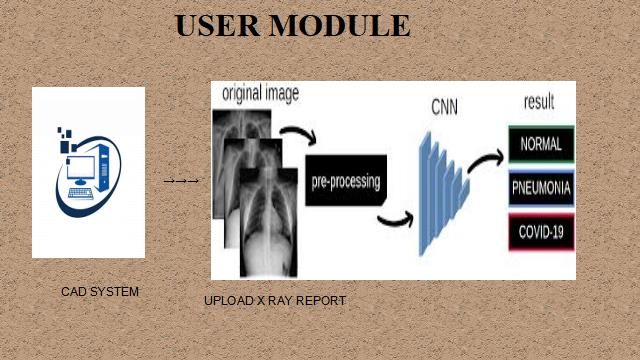


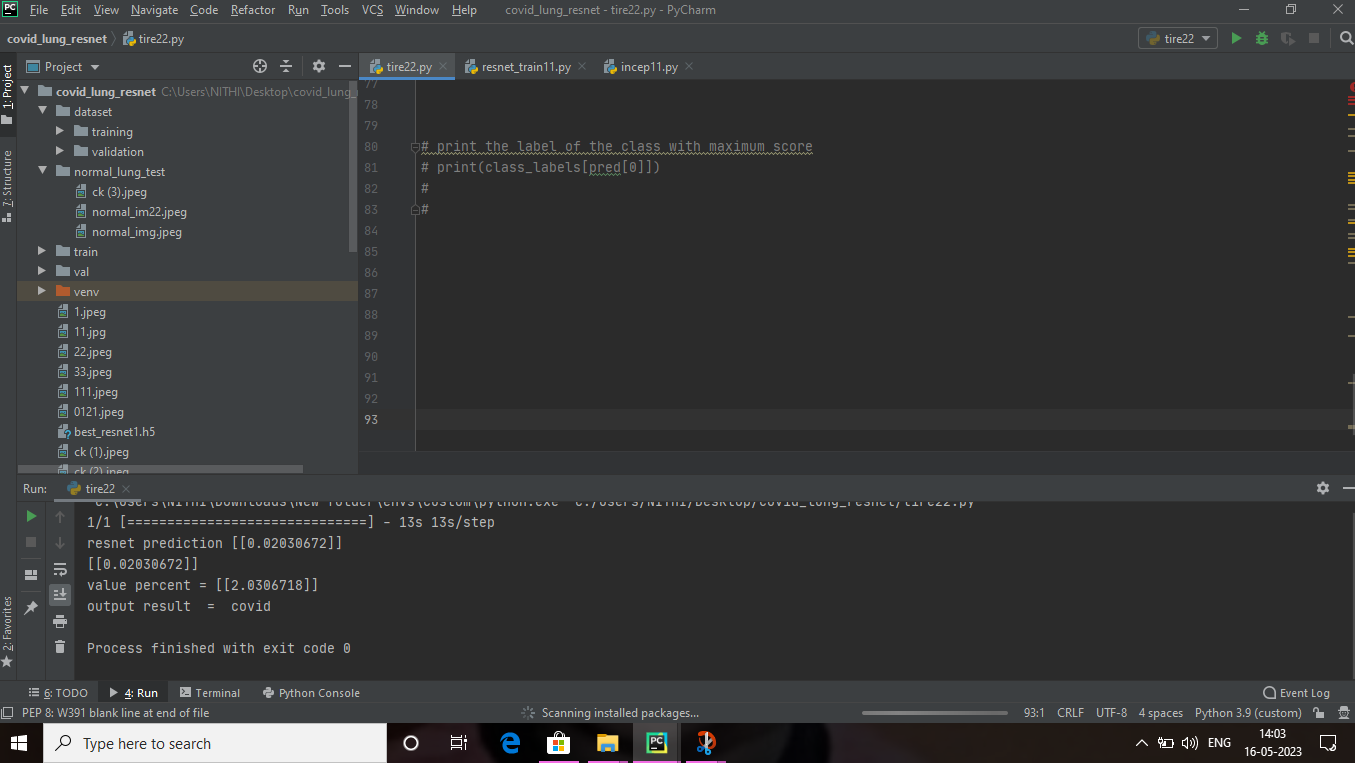
The home page looks like this. When you click on the button “Drop the scan”, you’ll be redirected to the predict section.

**X-RAY IMAGES:**



**PROCESS:**





**RESULT**