PNEUMONIA DETECTION THROUGH X-RAY

PYTHON PROJECT

Increment 2

TEAM 6

B2N

MEMBERS:

KOLLURI, NIKHITHA(12)

INAKOLLU, SRI NAGA BHUVANESHWARI(9)

DUKKIPATI, SRI SAI NITHIN CHOWDARY(4)

INTRODUCTION

It is very important for all the countries to perform some tests to know whether their people are suffering from any pandemics. Now-a-days pandemic crisis is progressing rapidly. Unfortunately, the capacity of the testing centres is very low in most of the countries. Usually, pneumonia is caused by viruses, bacteria and sometimes by microorganisms. We can confirm the diagnosis by taking the chest x-ray or blood cells, so here we are using chest x-rays. It is hard for every person to get the appointment of the doctor due to the evolution of the new virus Covid-19. Through this application, we want to give some mental peace for those who are suspecting whether they suffer from pneumonia or not, and through this, we can also have a chance to predict whether a person is affected by corona or not (as corona mainly affects respiratory system). All this provides us a motivation to choose this dataset and build an application which detects whether a person is suffering from pneumonia or not, as early diagnosis would be an important factor to be treated successfully.

In this project we will be predicting whether there exists pneumonia in the given image. We do this by taking the x-rays of some healthy persons and compare them with the x-rays of the people who are suffering from pneumonia. Then we train the model and will predict whether the person is suffering from pneumonia or not. In the present world, the doctors are very busy treating people with covid-19, through this application we provide computer-aided diagnosis system which would reduce the burden of clinicians. We can also have a chance to say whether a person is suffering from COVID or not because COVID virus mainly affects the respiratory system. We have opted for CNN motivated deep learning algorithms, since they have been used as the standard and best choice to classify the medical images.

Objectives:

- To train a model in a way, such that it predicts whether the X-ray uploaded has Pneumonia or not.
- To build an angular website that takes in the image as input and communicates with model which is already built and displays the prediction.
- To deploy the model on some cloud platform, so that the website can call the model API.

Features:

- A model which is trained to predict the detection of Pneumonia from the given Chest X-ray.
- A website that takes X-rays and displays the predicted results.

GIT Links:

- **Model : https://github.com/ntihindukkipati/CS5590_Python_DL.git**
- **Angular Site: https://github.com/Nikhitha8563/Python_Pro_UI**
- **Heroku: https://calm-peak-20297.herokuapp.com/handle_form**

Technologies Used:

- Python
- Angular
- Node
- Express

Python Concepts Used:

- From Keras
    - Convolutional Neural Networks
    - Dense
    - MaxPooling
    - Early Stopping
    - Image Generator

- Flask
    - Requests
    - RenderTemplates
    - Request

- OpenCV2
    - ImRead
    - ImShow
    - ImResize

Remaining parts of the projects:

- Establishing a connection between web application and API.
- Decreasing the slug size for deployment in Heroku.

Here we are showing the previous version(Version 1), the mistakes we have done and the present version(Version 2) where we have corrected the mistakes and compiled a new model.

Version - 1

DATASET:

We have used Kaggle's data set for Image classification using chest x-ray

STEP 1: (Reading the data set and pushing into List)

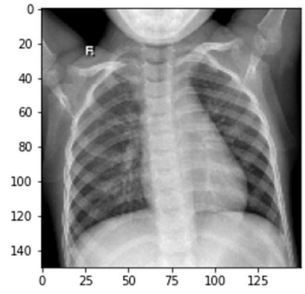
We are building our model in google COLAB so for this we have uploaded the image data set in to google COLAB.

1. we have used glob in-order to retrieve the image data from google drive. (in order to get all images of same pattern)

TRAINING DATA

```
[ ] from PIL import Image
    import glob
    c=0
    train_images=[]
    train_labels=[]
    for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/train/NORMAL/*.jpeg'): #assuming gif
        #img_normal = load_img(filename)
        #print(img_normal)
        img_normal = cv2.imread(filename,0)
        #print(im.shape)
        #print(type(im.shape))
        output = cv2.resize(img_normal, (150,150))
        train_images.append([output])
        train_labels.append(0)
        c=c+1
        print(c)
```

```
[ ] #help(cv2.imread)
    plt.imshow(output,cmap="gray")
    plt.show()
    c=0
    for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/train/PNEUMONIA/*.jpeg'): #assuming gif
        #img_normal = load_img(filename)
        img_normal = cv2.imread(filename,0)
        #print(im.shape)
        #print(type(im.shape))
        output = cv2.resize(img_normal, (150,150))
        train_images.append([output])
        train_labels.append(1)
        c=c+1
    print(c)
```



3875

TEST DATA

```
[ ] c=0
    test_images=[]
    test_labels=[]
    for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/test/NORMAL/*.jpeg'): #assuming gif
        #img_normal = load_img(filename)
        #print(img_normal)
        img_normal = cv2.imread(filename,0)
        #print(im.shape)
        #print(type(im.shape))
        output = cv2.resize(img_normal, (150,150))
        test_images.append([output])
        test_labels.append(0)
        c=c+1
        print(c)
```

```
[ ] c=0
    for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/test/PNEUMONIA/*.jpeg'): #assuming gif
        #img_normal = load_img(filename)
        img_normal = cv2.imread(filename,0)
        #print(im.shape)
        #print(type(im.shape))
        output = cv2.resize(img_normal, (150,150))
        test_images.append([output])
        test_labels.append(1)
        c=c+1
    print(c)
```

```
390
```

NOTE

We have used cv2.imread to read the image data and we have also converted into 1 channel in-order to reduce complexity. We have resized the image to 150,150,1 shape.

STEP 2:

Converted the list into numpy to convert the matrix form of 1 image to 1 dimension array.

```
[ ]  len(test_images)
     test_data=[]
     for x in test_images:
       test_data.append(x)
     test_data=np.array(test_data).reshape(-1,150,150,1)
     test_data.shape
     dimData = np.prod(train_data.shape[1:])
     print(dimData)
     train_data = train_data.reshape(train_data.shape[0],dimData)
     train_data.shape
     test_data = test_data.reshape(test_data.shape[0],dimData)
     test_data.shape
```

```
22500
(624, 22500)
```

STEP 3:

Scaled the training data and test data by dividing each pixel value by 255.0 in order to reduce the complexity and to get better accuracy.

```
[ ]  train_data = train_data.astype('float')
     test_data = test_data.astype('float')
     train_data /=255.0
     test_data /=255.0
```

```
[ ]  train_labels_one_hot = to_categorical(train_labels)
     test_labels_one_hot = to_categorical(test_labels)
     train_data.shape
```

```
(5216, 22500)
```

STEP 5:

Builded the model

MODEL

```
[ ] model = Sequential()
    model.add(Dense(512, activation='relu', input_shape=(dimData,)))
    model.add(Dense(512, activation='relu'))
    model.add(Dense(512, activation='relu'))
    model.add(Dense(512, activation='tanh'))
    model.add(Dense(512, activation='sigmoid'))
    model.add(Dense(2, activation='softmax'))
    model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
    model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=100, verbose=1,
                      validation_data=(test_data, test_labels_one_hot))
```

```
[ ] 5216/5216 [==============================] - 8s 1ms/step - loss: 0.1229 - acc: 0.9523 - val_loss: 2.1797 - val_acc: 0.7644
    Epoch 96/100
    5216/5216 [==============================] - 8s 1ms/step - loss: 0.1177 - acc: 0.9546 - val_loss: 1.5186 - val_acc: 0.7869
    Epoch 97/100
    5216/5216 [==============================] - 8s 1ms/step - loss: 0.1241 - acc: 0.9484 - val_loss: 2.1035 - val_acc: 0.7179
    Epoch 98/100
    5216/5216 [==============================] - 8s 1ms/step - loss: 0.1063 - acc: 0.9551 - val_loss: 2.3352 - val_acc: 0.7324
    Epoch 99/100
    5216/5216 [==============================] - 8s 1ms/step - loss: 0.1155 - acc: 0.9525 - val_loss: 2.2383 - val_acc: 0.7276
    Epoch 100/100
    5216/5216 [==============================] - 8s 1ms/step - loss: 0.1250 - acc: 0.9484 - val_loss: 2.1548 - val_acc: 0.6971
    <keras.callbacks.History at 0x7f22b9055ac8>
```

NOTE:

We have got an accuracy of 94% and validation of 69%. The loss is too high in the both ends.

STEP 6:

We have saved the model using pickle library.

```
[ ] import pickle
    with open("/content/drive/My Drive/Colab Notebooks/chest_xray/First_Model.pk1",'wb') as file:
          pickle.dump(model,file)
```

STEP 7:

We have used flask frame work in-order to create a API.

```
@app.route('/handle_form', methods=['POST'])
def handle_form():
    print("Posted file: {}".format(request.files['file']))
    file = request.files['file']
    #file.shape
    print(file)
    file.save(secure_filename("save.jpeg"))
    #im = cv2.imread(file)
    im=cv2.imread("save.jpeg",0)
    #print(im.shape)
    #print(im)
    result=process_img(im)
    #print(type(im))
    #plt.imshow(file)
    #plt.show()
    #print(file.shape)
    #file.save()
    #print(file)
    #cv2.imwrite(filename='saved_img.jpg')
    return result

@app.route("/")
def index():
   return "working"

if __name__ == "__main__":
    app.run()
```

In-order to predict we need to convert the image format to feed into our model. So, for that we have used the same pre-processing technique in-order to convert received image.

```
from werkzeug.utils import secure_filename
import joblib
from flask import Flask, request,jsonify, render_template
import cv2
#from PIL import Image
#import tensorflow as tf
#import keras
import numpy as np

im=[]
app = Flask(__name__)

def process_img(imk):
    output1 = cv2.resize(imk, (150, 150))
    output1 = output1.astype('float')
    output1 /= 255.0
    print(type(output1))
    output1 = np.array(output1).reshape(-1, 150, 150, 1)
    #output1.shape
    dimData = np.prod(output1.shape[1:])
    output1 = output1.reshape(output1.shape[0], dimData)
    print(output1)
    classifer = joblib.load("First_Model.pk1")
    x=classifer.predict_classes(output1[[0], :])
    #return "hello"
    if x[0]==1:
        result="Patient have Pnemonia"
    else:
        result="patient doesn't have pnemonia"
    return jsonify({"data" : result })
```

STEP 8:

We have deployed this into HEROKU CLOUD in-order to make it a rest service so that it can be useful for us to use in our angular application.

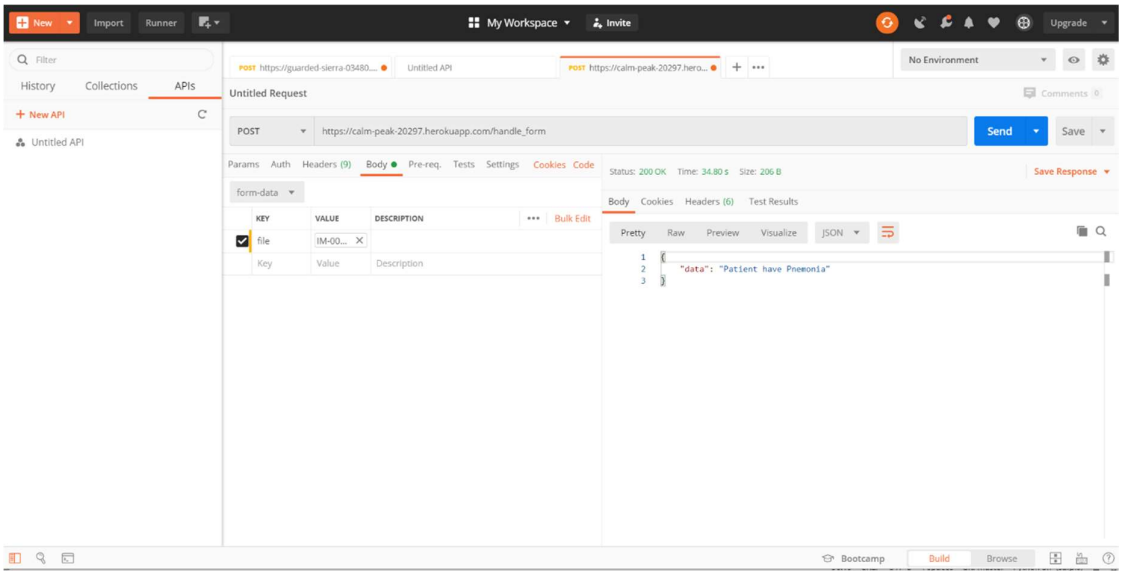
LINK:

GET REQUEST:

https://calm-peak-20297.herokuapp.com/

POST REQUEST:

https://calm-peak-20297.herokuapp.com/handle_form



CHALLENGES FACED:

1. While deploying into Heroku we have slug size issue because of TensorFlow library. We have only free space of 500MB and only TensorFlow was taking 540MB and my model size was around 90MB. In order to solve this we have used .slugignore file which will ignore the un wanted libraries and stop installing them but in my case it didn't work because we the TensorFlow Complier size was itself 540MB. Finally we have tried with TensorFlow version 1.13 version which has less compiler size and it did solve our problem.

MISTAKES THAT WE HAVE LEARNED:

1. Shuffle the data before feeding into model. In the first version we haven't shuffle the data which has made our model biased.

2. We were not fully aware of the convolutional neural network.

VERSION 2:

STEP 1:

Importing the libraries and importing the images from the google drive

```
[ ] %tensorflow_version 1.15
    import tensorflow as tf
    from keras import Sequential
    import numpy as np
    from keras.layers import Dense
    from keras.utils import to_categorical
    from keras.preprocessing.image import ImageDataGenerator, load_img
    import os
    import matplotlib.pyplot as plt
    from keras.layers import Dense, Flatten, Dropout, Input, BatchNormalization
    from keras.constraints import maxnorm
    from keras.models import Model
    from keras.optimizers import SGD, Adam
    from keras.layers.convolutional import Conv2D, MaxPooling2D
    from keras.utils import np_utils
    from keras import backend as K
    K.common.image_dim_ordering()
    from tensorflow.keras.callbacks import EarlyStopping
```

```
`%tensorflow_version` only switches the major version: 1.x or 2.x.
You set: `1.15`. This will be interpreted as: `1.x`.
```

READING THE TEST DATA TRAIN DATA INTO LIST

```
[ ] from PIL import Image
    import glob
    import cv2
    train_images=[]
    for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/train/NORMAL/*.jpeg'):
        img_normal = cv2.imread(filename)
        output = cv2.resize(img_normal, (50,50))
        train_images.append([output,0])
```

```
[ ] for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/train/PNEUMONIA/*.jpeg'):
        img_normal = cv2.imread(filename)
        output = cv2.resize(img_normal,(50,50))
        train_images.append([output,1])
```

```
[ ]  test_images=[]
     for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/test/NORMAL/*.jpeg'):
         img_normal = cv2.imread(filename)
         output = cv2.resize(img_normal, (50,50))
         test_images.append([output,0])
```

```
[ ]  for filename in glob.glob('/content/drive/My Drive/Colab Notebooks/chest_xray/test/PNEUMONIA/*.jpeg'):
         img_normal = cv2.imread(filename)
         #print(im.shape)
         #print(type(im.shape))
         output = cv2.resize(img_normal, (50,50))
         test_images.append([output,1])
```

STEP 2:

Shuffling the data in order to conflict bias in our model. And we have pushed the list data into nd.array to feed into our model

```
[ ]  import random
     random.shuffle(train_images)
```

```
[ ]  x_train=[]
     y_train=[]
     for im,label in train_images:
       x_train.append(im)
       y_train.append(label)
```

```
]  random.shuffle(test_images)
```

```
]  x_test=[]
   y_test=[]
   for im,label in test_images:
     x_test.append(im)
     y_test.append(label)
```

```
]  x_test=np.array(x_test).reshape(-1,50,50,3)
```

We have reshaped the image size to 50,50,3. We reduced the size in order to avoid crash while running epochs.

STEP 3:

Covert the labels to categorical values and scale the images by dividing each pixel by 255.

```
[ ]  x_train = x_train.astype('float32')
     x_test = x_test.astype('float32')
     x_train = x_train / 255.0
     x_test = x_test / 255.0
```

01

[1,0] [0,1] 51231,50,50,3

pnemonia

```
[ ]  y_train = np_utils.to_categorical(y_train)
     y_test = np_utils.to_categorical(y_test)
     num_classes = y_test.shape[1]
```

STEP 4:

BUILD THE MODEL

```
[ ]  model = Sequential()
     model.add(Conv2D(256, (3, 3), input_shape=(x_train.shape[1:]), padding='same', activation='relu'))
     model.add(Dropout(0.5))
     model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     model.add(Dropout(0.5))
     model.add(Flatten())
     model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
     model.add(Dropout(0.3))
     model.add(Dense(num_classes, activation='softmax'))
```

STEP 5:

Used the call back function to reduce overfitting in the model. Here we are observing the for 5 epochs whether my validation loss in decreasing (1e-1=0.001).

```
epochs = 20
lrate = 0.001
decay = lrate/epochs
sgd = Adam(lr=lrate)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

```
monitor=EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto',restore_best_weights=True)
```

```
model.fit(x_train, y_train, validation_data=(x_test, y_test), callbacks=[monitor],epochs=150, batch_size=512)
```

STEP 6:

We have got an accuracy of 74% and val_acc of 62% and loss is to high which is leading in wrong prediction.

```
Epoch 26/150
5216/5216 [==============================] - 486s 93ms/step - loss: 0.6071 - accuracy: 0.7429 - val_loss: 0.6617 - val_accuracy: 0.6250
Epoch 27/150
5216/5216 [==============================] - 486s 93ms/step - loss: 0.6052 - accuracy: 0.7429 - val_loss: 0.6616 - val_accuracy: 0.6250
Epoch 28/150
5216/5216 [==============================] - 485s 93ms/step - loss: 0.6033 - accuracy: 0.7429 - val_loss: 0.6616 - val_accuracy: 0.6250
Epoch 29/150
5216/5216 [==============================] - 484s 93ms/step - loss: 0.6014 - accuracy: 0.7429 - val_loss: 0.6616 - val_accuracy: 0.6250
Restoring model weights from the end of the best epoch.
Epoch 00029: early stopping
<keras.callbacks.callbacks.History at 0x7f4bb4148668>
```

```
import pickle
with open("/content/drive/My Drive/Colab Notebooks/chest_xray/MODEL_V2_PN.pk2",'wb') as file:
      pickle.dump(model,file)
```

```
x=model.predict_classes(x_train[[50],:])
print(x[0])
```

We have saved our model using pickle library.

STEP 7:

We have used flask frame in order to create web UI

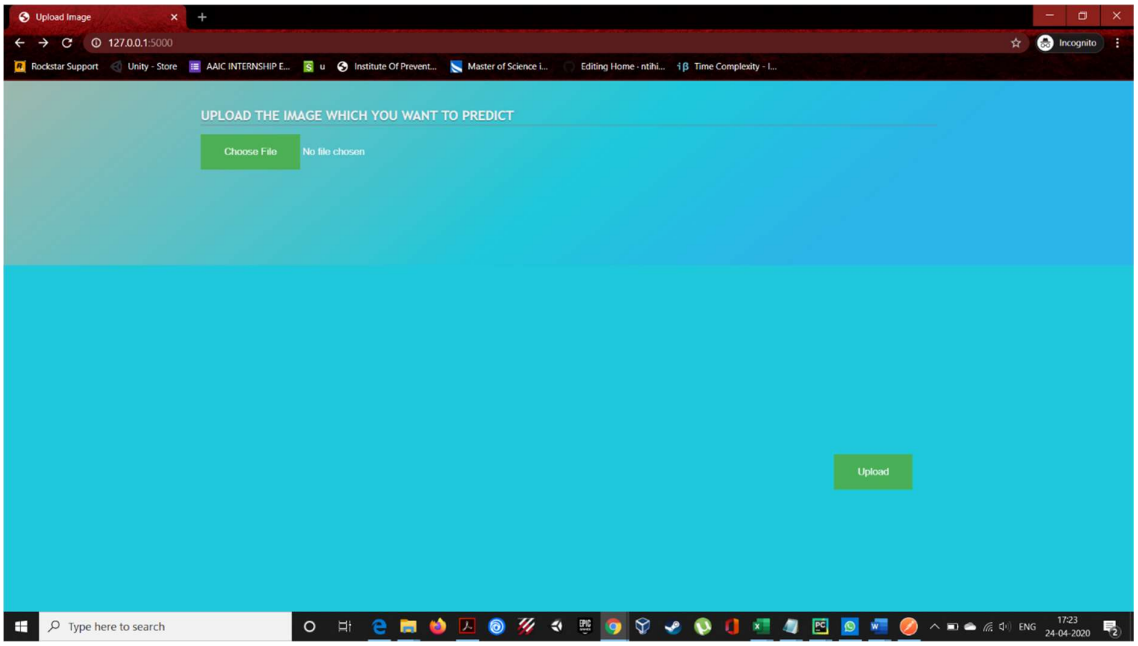
```
@app.route('/', methods=['GET'])
def index():
    return render_template('fileUpload.html')

@app.route('/', methods=['GET', 'POST'])
def handle_form():
    if request.method == 'POST':
        file = request.files['file']
        file.save(secure_filename("save.jpeg"))
        im=cv2.imread("save.jpeg")
        result=process_eval(im)
        return render_template('fileUpload.html',result=result)

if __name__ == "__main__":
    app.run()
```

For feeding received image we need to convert that image into feedable format.

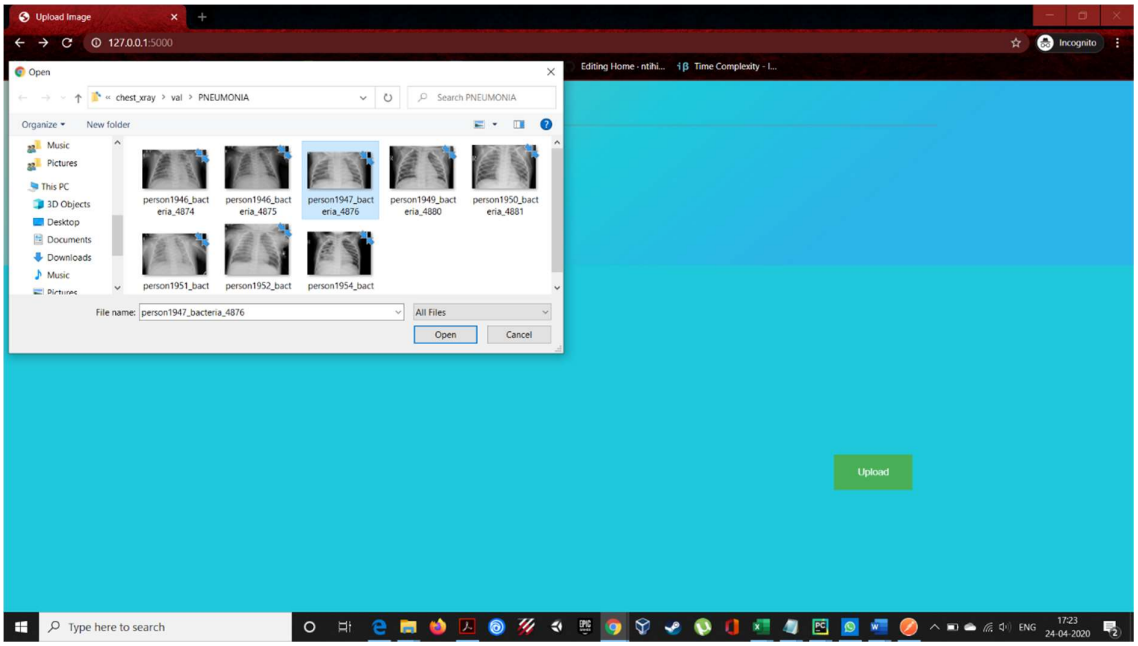
```
from werkzeug.utils import secure_filename
import joblib
from flask import Flask, request, render_template
import cv2
import numpy as np

# Define a flask app
app = Flask(__name__)

def process_eval(imk):
    output1 = cv2.resize(imk, (50,50))
    output1 = output1.astype('float')
    output1 /= 255.0
    output1 = np.array(output1).reshape(-1, 50, 50, 3)
    classifer = joblib.load("MODEL_V2_PN.pk2")
    x = classifer.predict_classes(output1[[0], :])
    if x[0] == 1:
        result = "PATIENT IS HAVING PNEMONIA"
    else:
        result = "PATIENT IS NORMAL"
    return result

@app.route('/', methods=['GET'])
def index():
    return render_template('fileUpload.html')
```

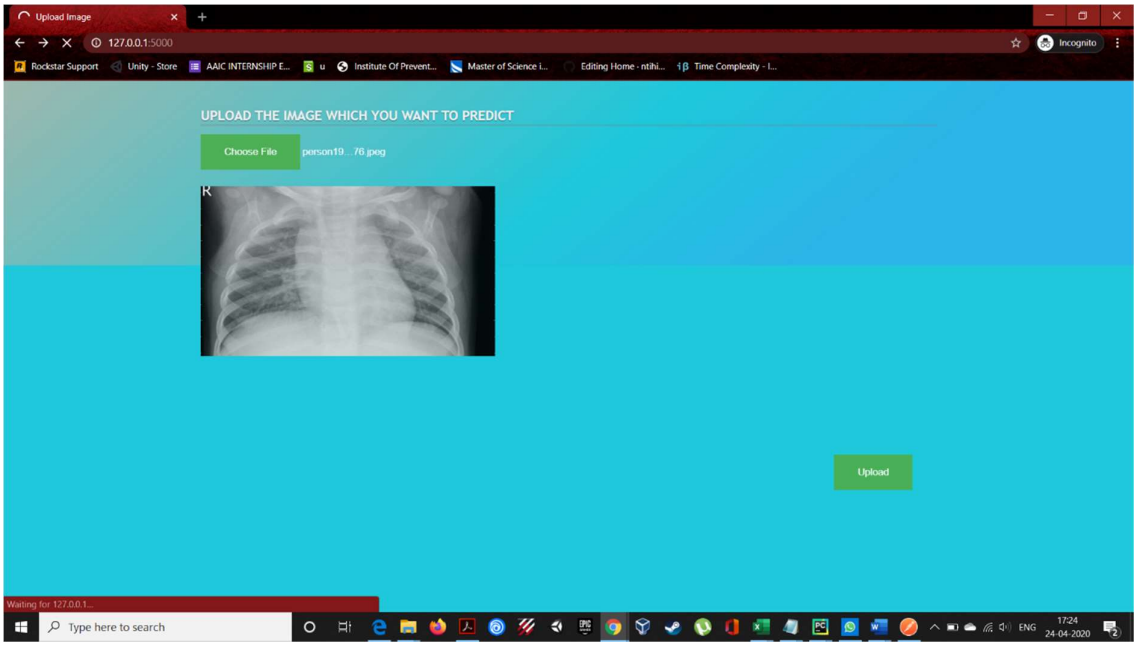
Converted the image into size of 50,50,3 and passed through our model for prediction.

1. local host 127.0.0.1.5000/

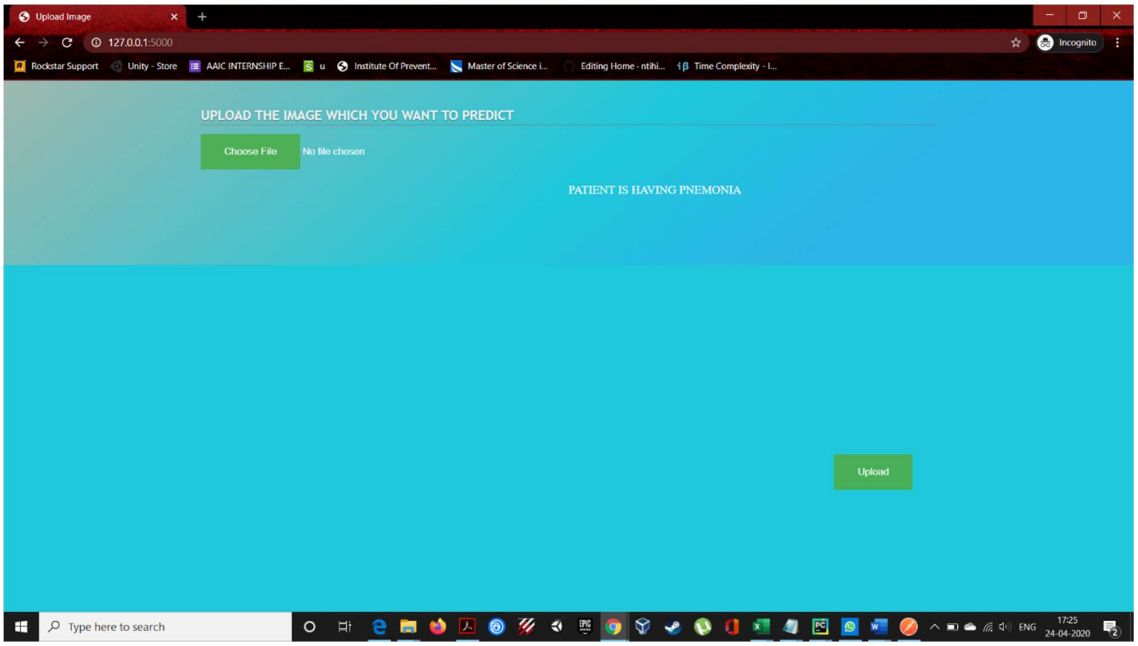


2. Uploading the image





3. FINAL RESULT



WHAT WE WANT TO CHANGE:

1. We will use hyperparameter to get best possible functions and values to use.

2. Enlarge our image size so that I could help in better prediction.

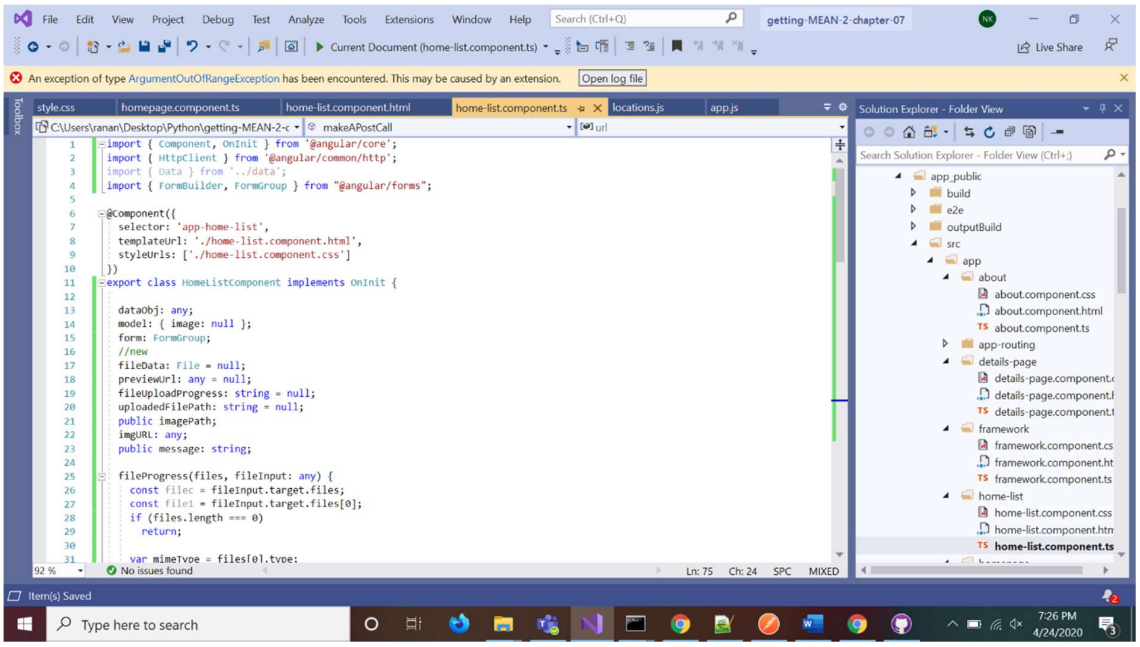
3. Changing the last layer to sigmoid instead of SoftMax.

***NEED HELP IN:

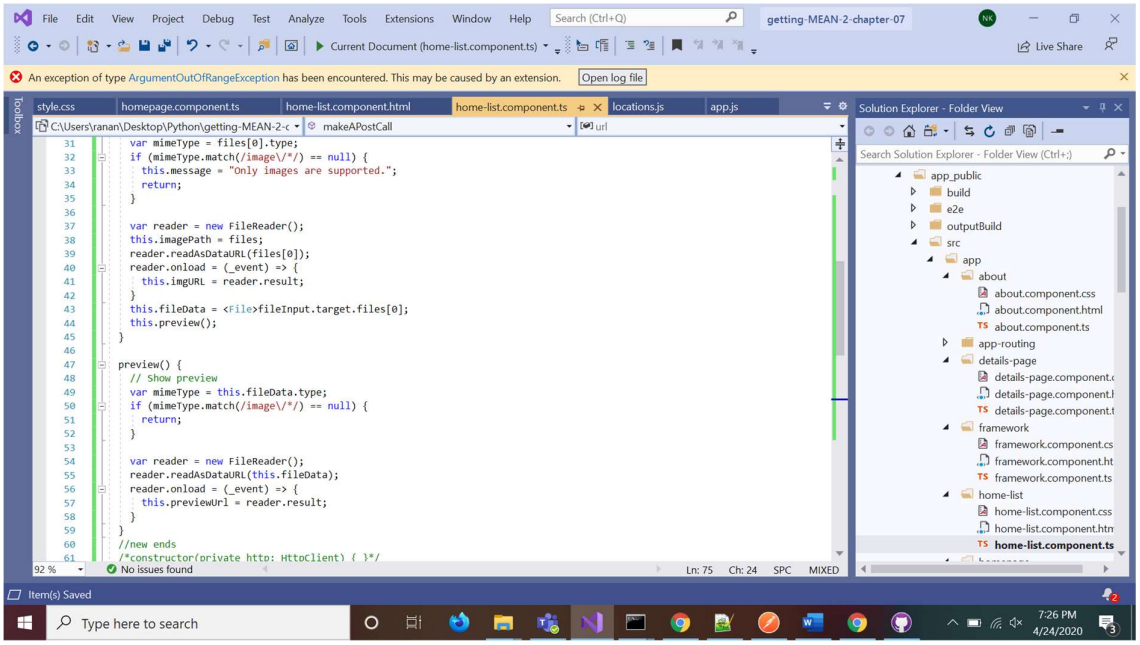
1. In our first version of our model we have tried to solve the slug issue by down grading the version of TensorFlow but in the first case my model was only 90MB but now my model size is around 1GB which makes me impossible it to deploy in HEROKU. Is there any way we could shrink the size of model.

2. We are facing some post request issue in our angular application. Please can you let us know how we should take image in rest API side so that it could be easy for us to make a call from angular application.

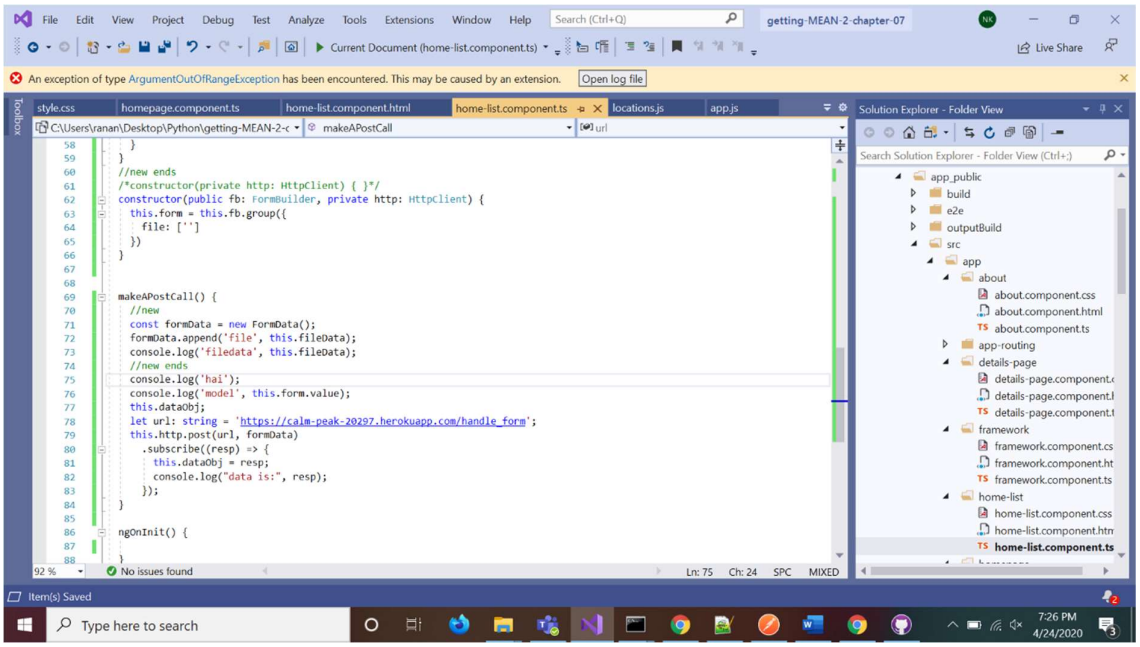
- Please find the code snippets for the API call from angular.



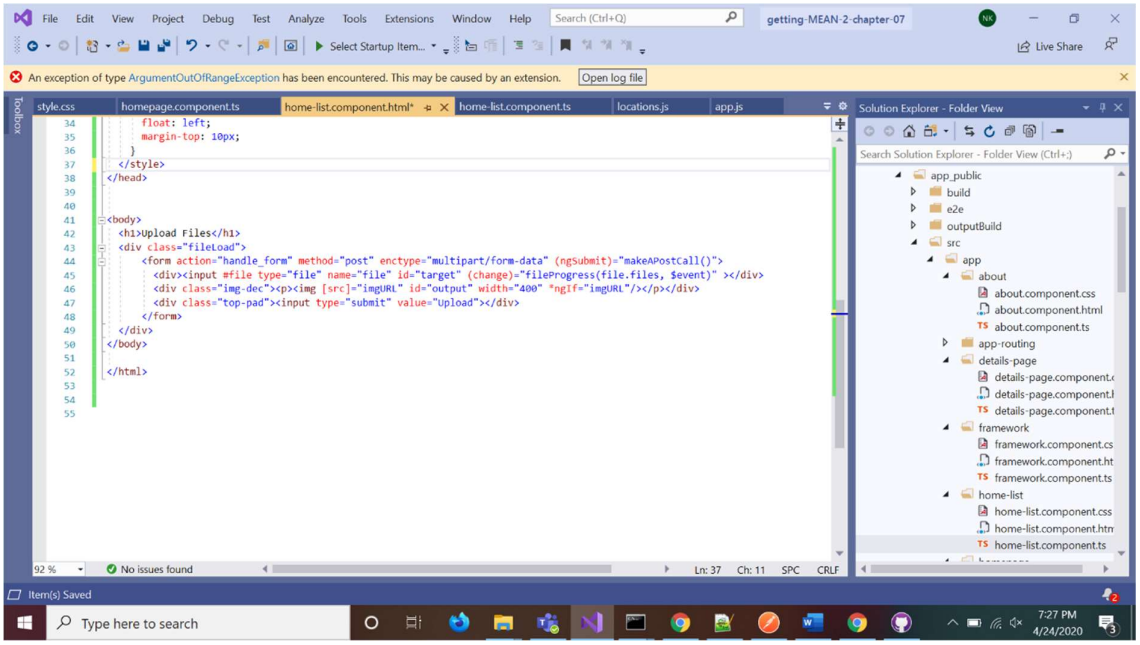
- Reading the file which is taken from the UI



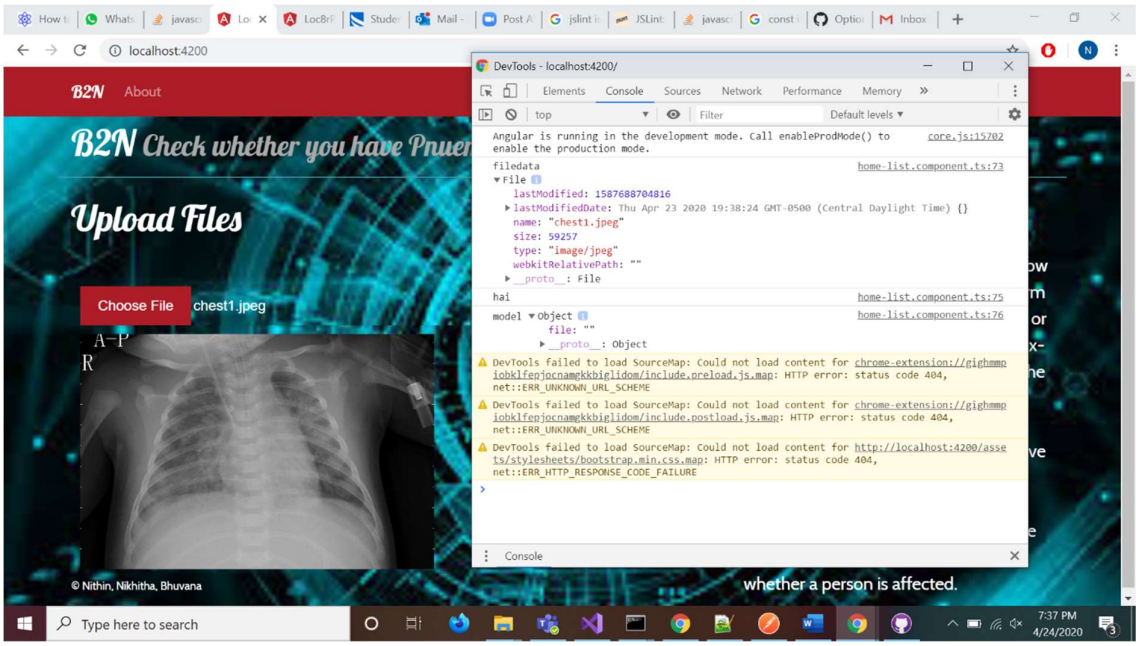
- After reading the file, we are sending the that to the API call.



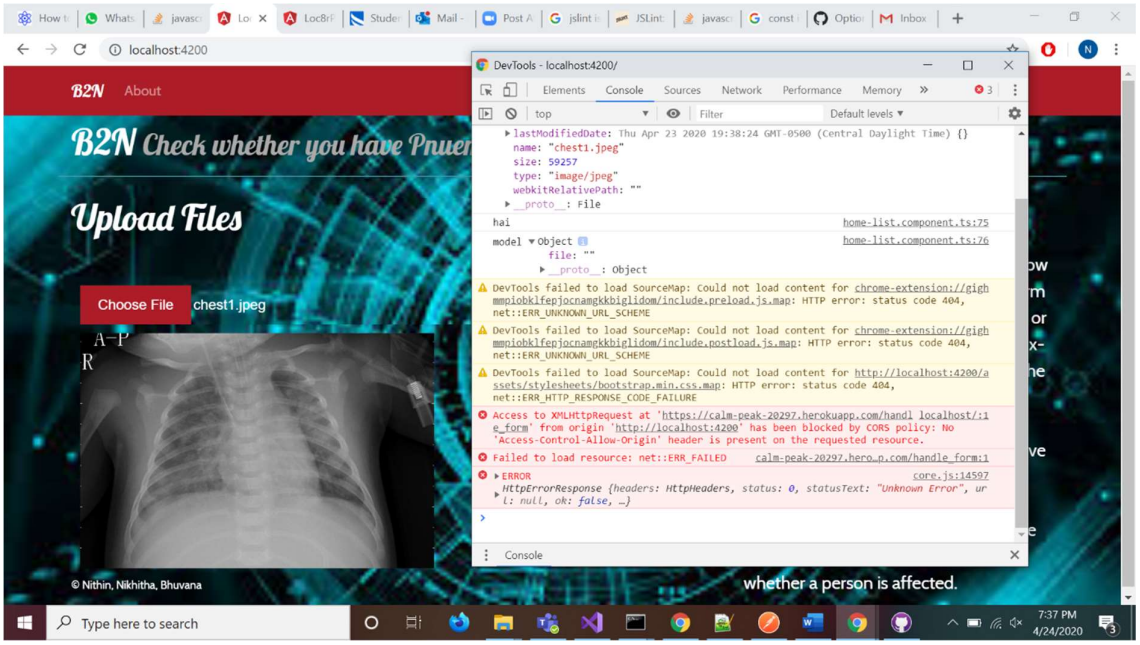
- This is the HTML file.



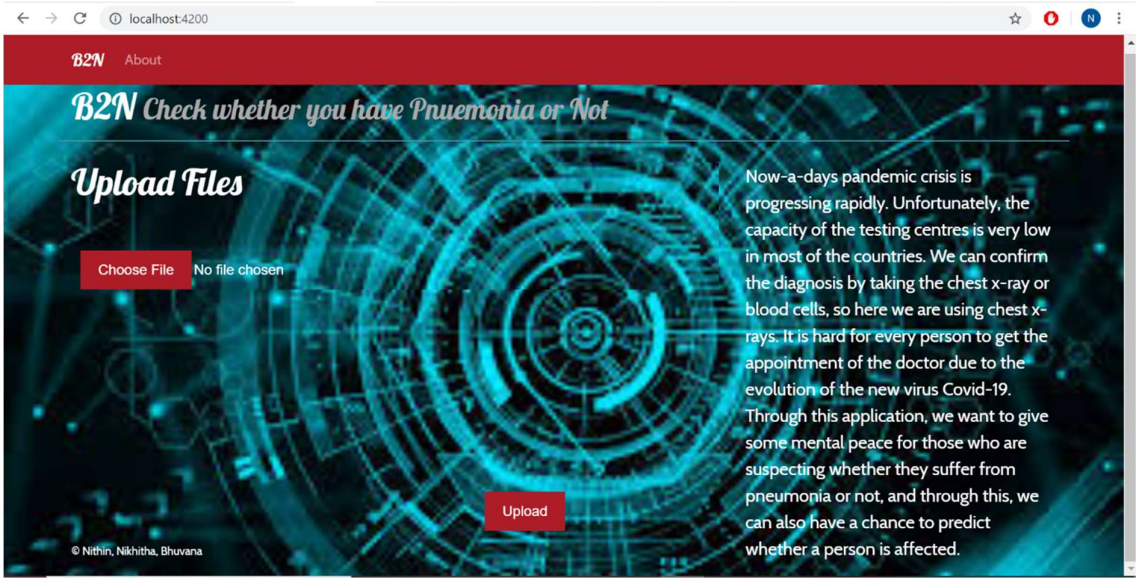
- We are taking the input file in this format to send it to the API call.



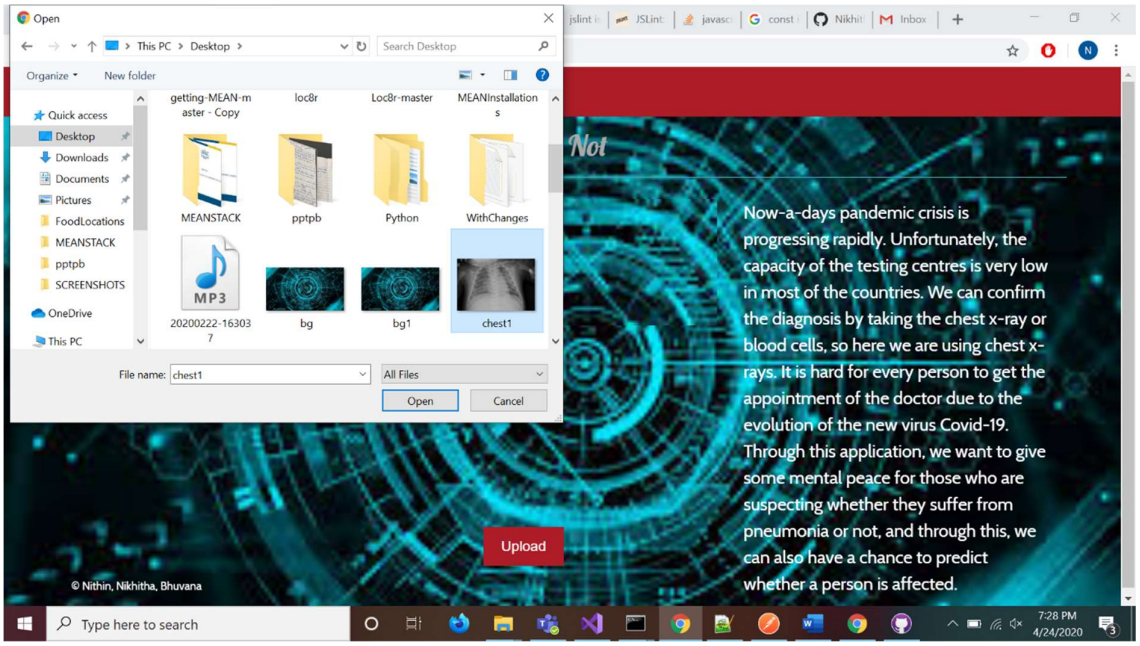
- We are facing the following issue after calling our API.

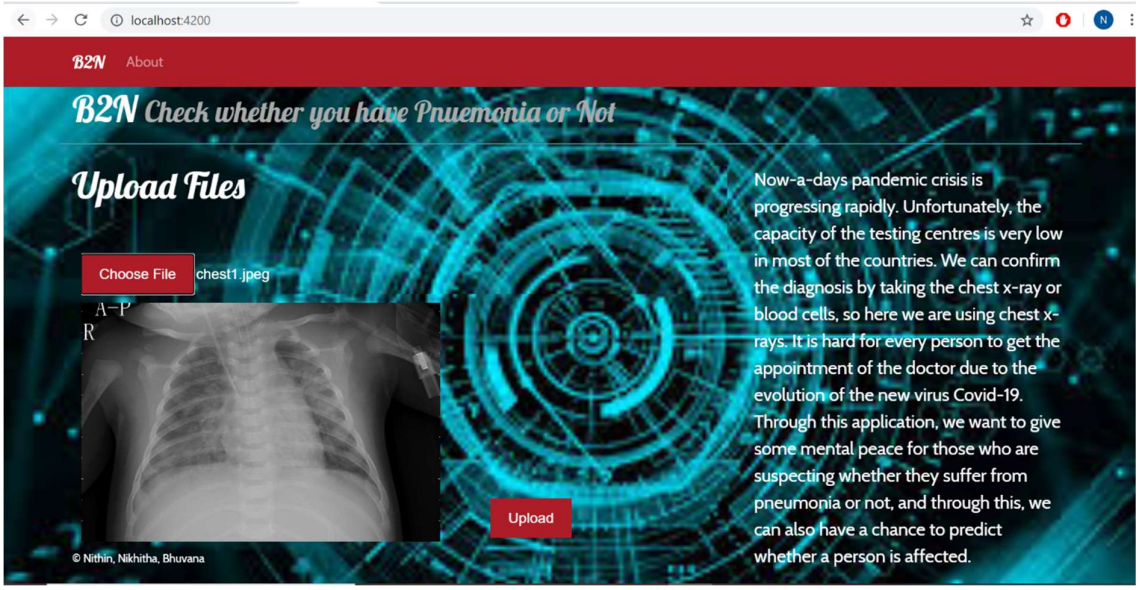


- We have build a new website using angular to upload the Chest X-ray and get the prediction.

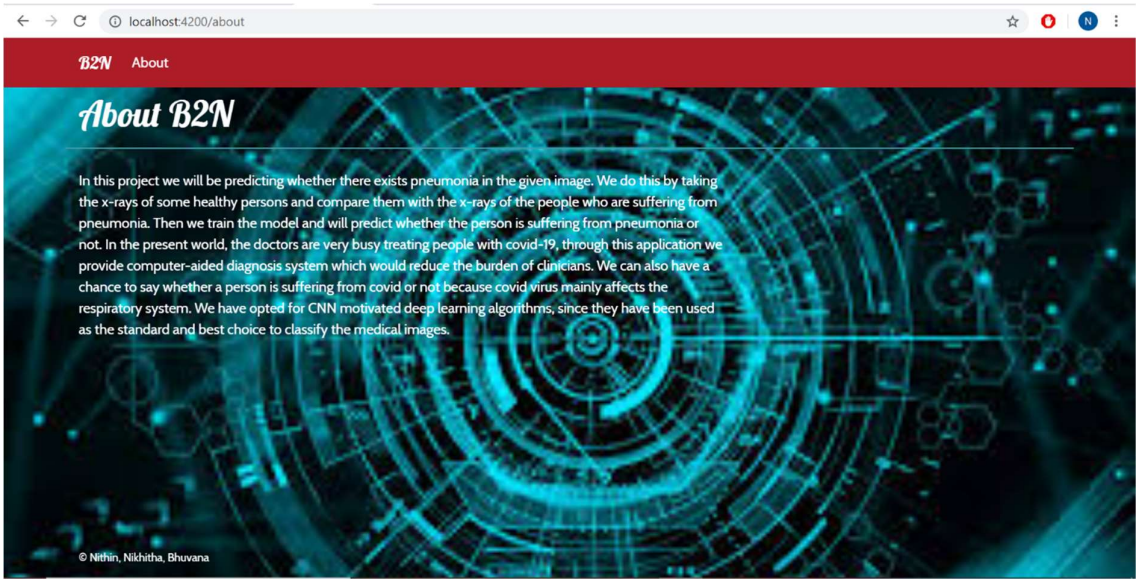


- Once we open the site, there will be a button named Choose File. On clicking it we can select the image from our desktop. The file which we have selected will be displayed below it and the name of the file will get displayed beside the Choose File button.





- We also have an about page, which just gives a brief description of our site.


localhost:4200/about
B2N
About
About B2N
In this project we will be predicting whether there exists pneumonia in the given image. We do this by taking the x-rays of some healthy persons and compare them with the x-rays of the people who are suffering from pneumonia. Then we train the model and will predict whether the person is suffering from pneumonia or not. In the present world, the doctors are very busy treating people with covid-19, through this application we provide computer-aided diagnosis system which would reduce the burden of clinicians. We can also have a chance to say whether a person is suffering from covid or not because covid virus mainly affects the respiratory system. We have opted for CNN motivated deep learning algorithms, since they have been used as the standard and best choice to classify the medical images.
© Nithin, Nikhitha, Bhuvana