AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

# PROJECT TEAM : M.MANIKANDNA(TL).

# V.RAJA.

# T.SARAN KUMAR.

# M.VISAGAN.

**PROJECT LINK :** [IBM-EPBL](https://github.com/IBM-EPBL)/  
**IBM-Project-6177-1658824241**

INTRODUCTION

## OVERVIEW :

* Erythema, abnormal redness of the skin.
* Erythema is caused by dilation and irritation of the superficial capillaries; the augmented flow of blood through them imparts a reddish hue to the skin. Skin redness can have cause that aren't due to underlying disease. Examples include sunburn,friction,poorly fitting clothes, messages, too much pressure on the area, blushing or exercise.
* Erythema may arise from a great variety of cause and disease conditions. Blushing is a transient from of erythema.
* Erythema multiforme (EM) is a cutaneous and mucosal hypersensitivity reaction with characteristic lesions in target triggered by certain antigenic stimuli.
* It represents an acute condition, sometimes recurrent, of the skin and mucosal membranes manifested by papular, bullous, and necrotic lesions.

LITERATURE SURVEY :

# EXISTING PROBLEM:

Rednes of the skin, caused by hyperemia of the capillaries in the lower layers of the

skin.

Computer-aided diagnosis (CAD) is a computer-based sytem that is used in the medical imaging field to aid healthcare workers in their diagnoses. CAD has become a mainstream tool in several medical field such as a memmography and colonography.

However, in dermatology, although skin disease is a common disease, one in which early detection and classification is crucial for the successful treatment and recovery of patients, dermatologists perform most nonivasive screening tests only with the naked eye.

We have shown that even without a large dataset and high-quality images,with higher quality and larger quantity of data, it will be viable to use state-of-the-art models to enable the use of CAD in the field of dermatology.

## PROPOSED SOLUTION :

For all forms of erythema multiforme (EM), the most important treatment is usually symptomatic, including oral antihistamines, analgesics, local skin care, and soothing mouthwashes (eg, oral rinsing with warm saline or a solution of diphenhydramine, xylocaine, and kaopectate).

Topical steroids may be considered. For more severe cases, meticulous wound care and use of Burrow or Domeboro solution dressings may be necessary.

The cause of the erythema multiforme should be identified, if possible. If a drug is suspected, it must be withdrawn as soon as possible. This includes all medications begun during the preceding 2 months. Discontinue all unnecessary medications.

## KEY WORDS :

Capillary malformation-arteriovenous malformation; EPHB4 mutation; Erythema; Port-wine stain.

## TREATMENT :

### SELF-TREATMENT

Avoiding harsh or perfumed soaps, detergents and lotions, as well as any known allergy triggers may help to reduce redness. Using an antihistamine or steroid cream may also help.

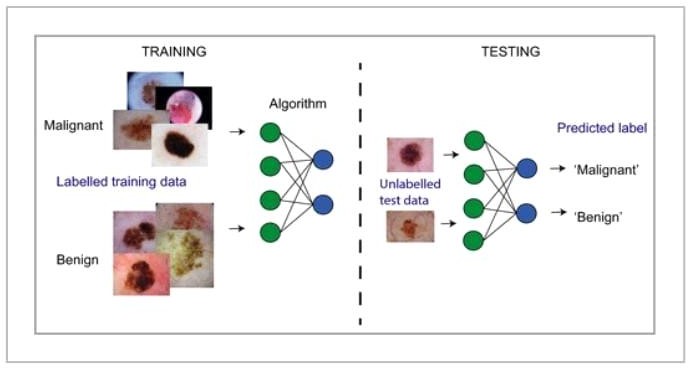
### SEEKING MEDICAL CARE:

* See a doctor immediately if the redness.
* Appears suddenly or for no apparent reason.
* Spreads quickly or covers most of your body.
* Is accompanied by fever.

Make an appointment to see a doctor if the redness:

* Is persistent.
* Feels painful.
* Oozes yellow or green fluid.
* Occurs along with blisters.

## BLOCK DIAGRAM :



FLOW CHART :

* Create IBM services.
* Creating skills for erythema.
* Collecting the database for Erythema .
* Creating & Installing for Python IDE.
* Creating Microsoft's Visual Object Tagging Tool (VoTT).
* Installing Python Packages.
* Creating IBM cloud, Register & Login for IBM cloud.
* Creating a Service credentials.
* Creating a Dataset.
* Creating a HTML web page.
* Creating a PYTHON code.
* Integrate the Skin disease for Erythema with web page.

## SIGNS AND SYMPTOMS :

Erythema multiforme:

* Fatigue, fever, and itching (before lesions appear).
* Sudden outbreak of spots, bumps, and lesions (usually on knees, elbows, palms, hands, and feet).
* Target lesions (spots surrounded by rings of normal and red skin, looking like a target).
* Erythema infectiosum (caused by a virus and known as fifth disease), rash on face and arms lasting about 2 weeks.

## Erythema nodosum:

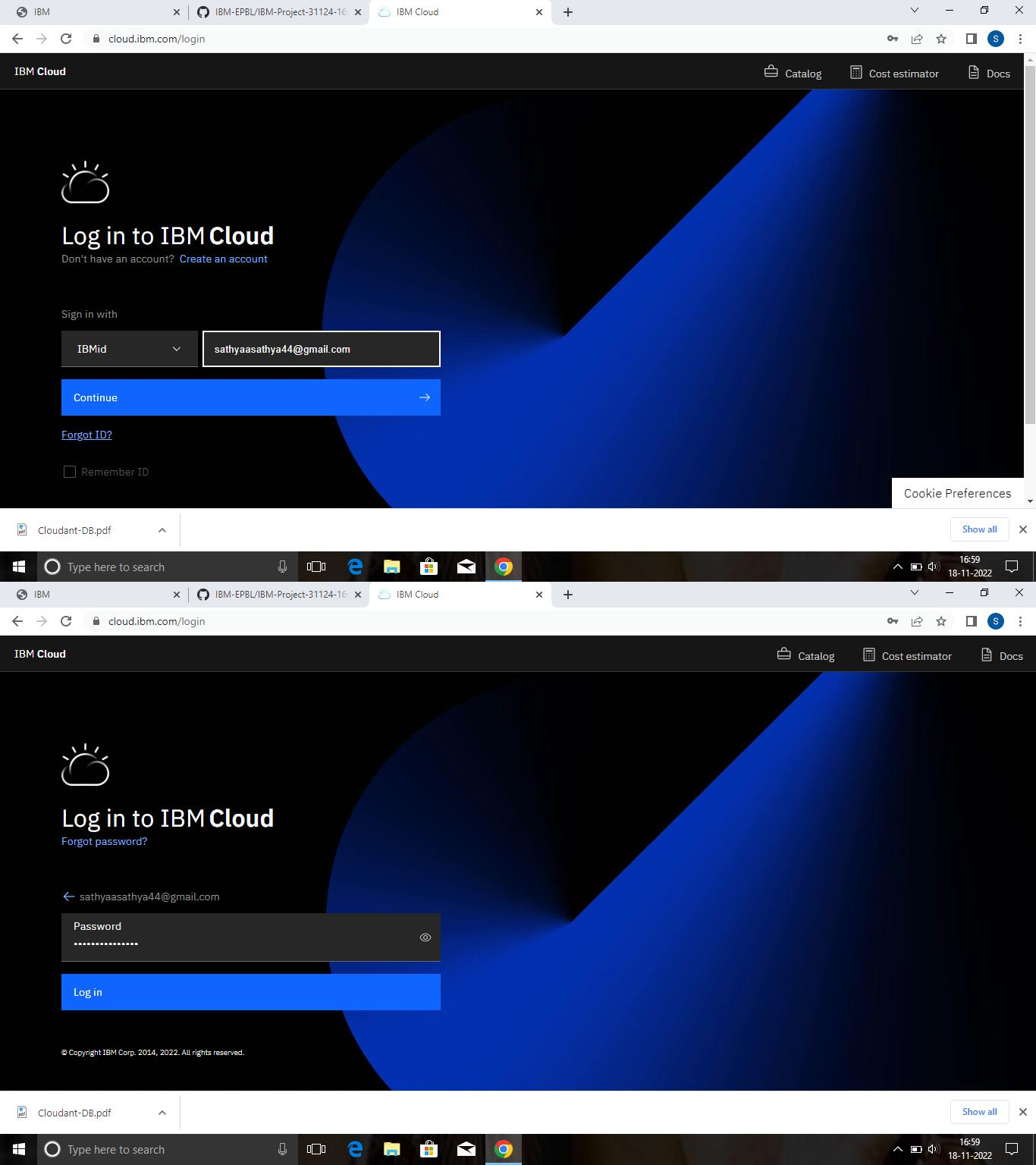
* Fatigue, flu-like symptoms (before lesions appear).
* Clusters of nodules (small round masses) and lesions on shins, forearms, thighs, and trunk.
* Red, painful lesions become soft and bluish, and fade to yellow and brown.
* Joint pain.
* Arthritis.

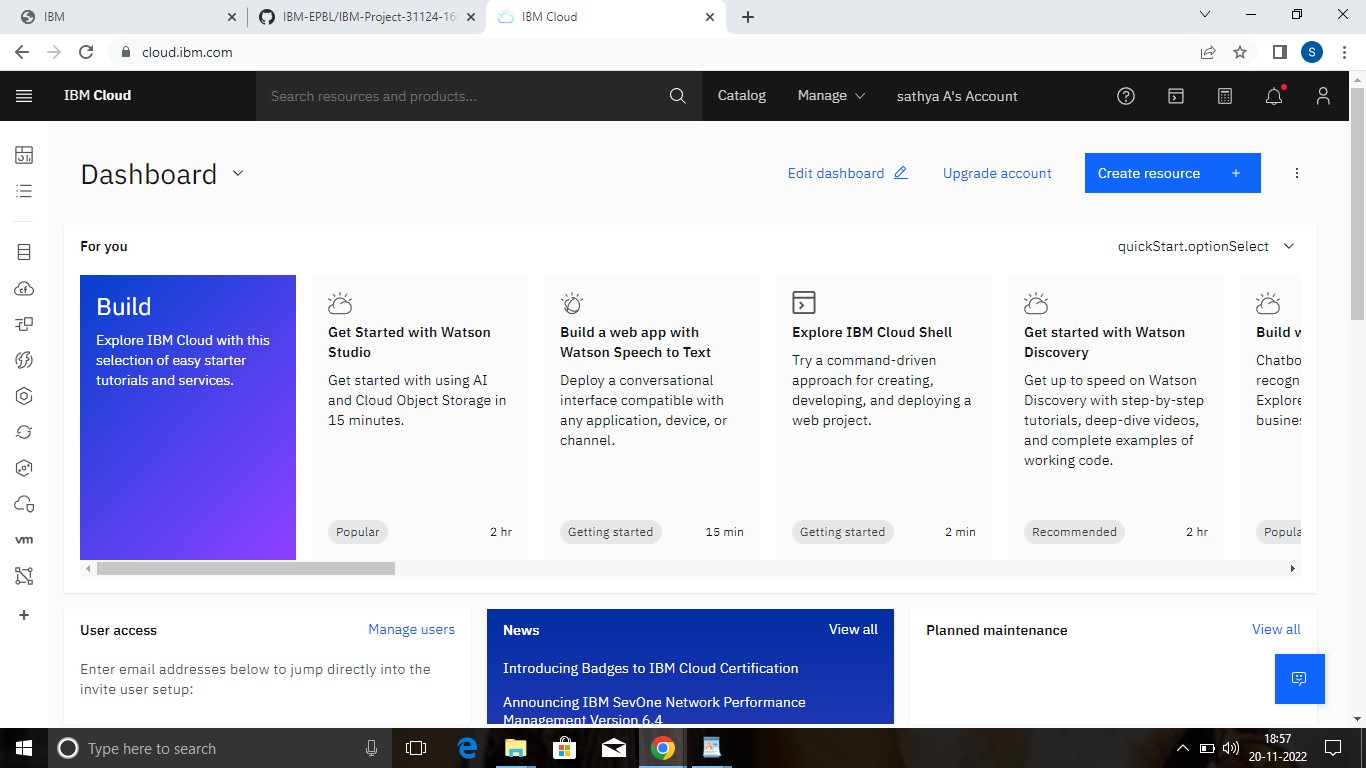
CREATE IBM SERVICE :

# CREATE IBM CLOUD:

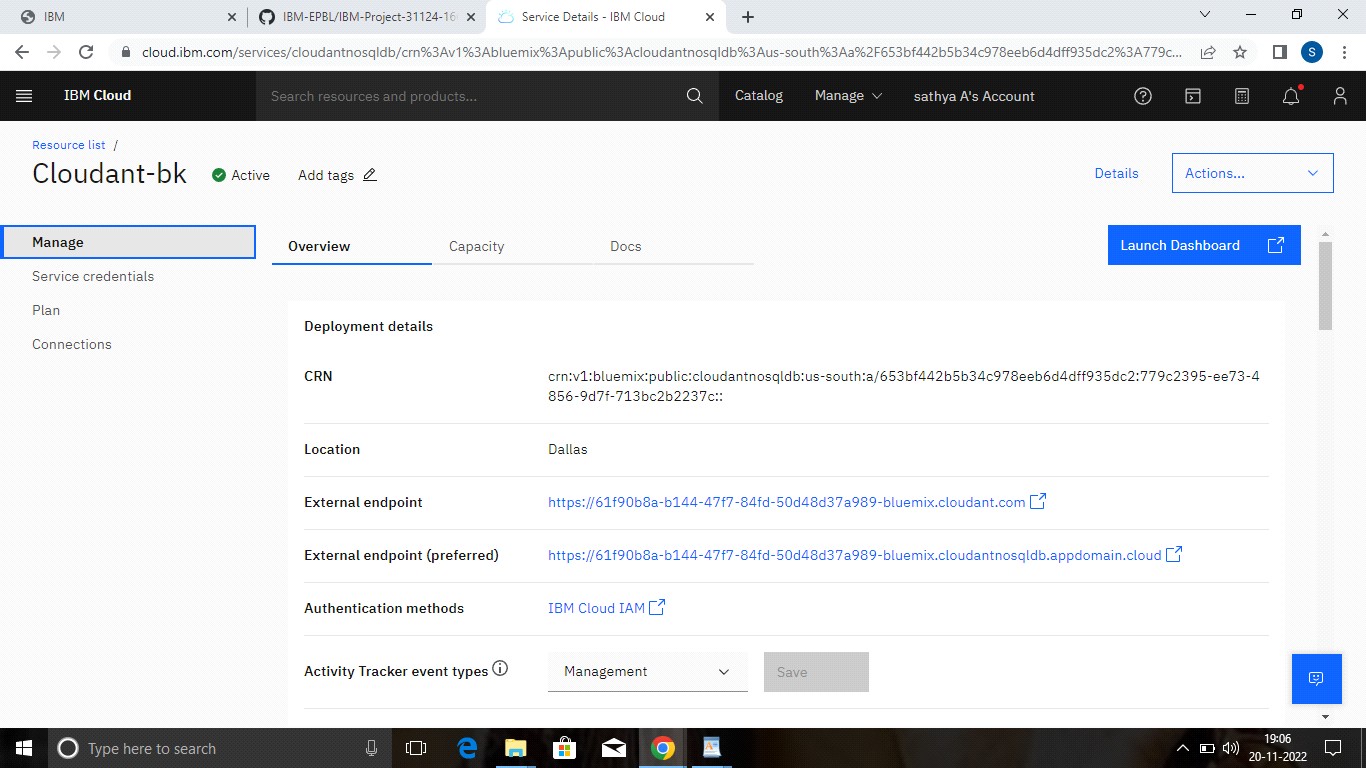
In this activity, you will be creating the Necessary IBM service. The following are the service that you have to create.

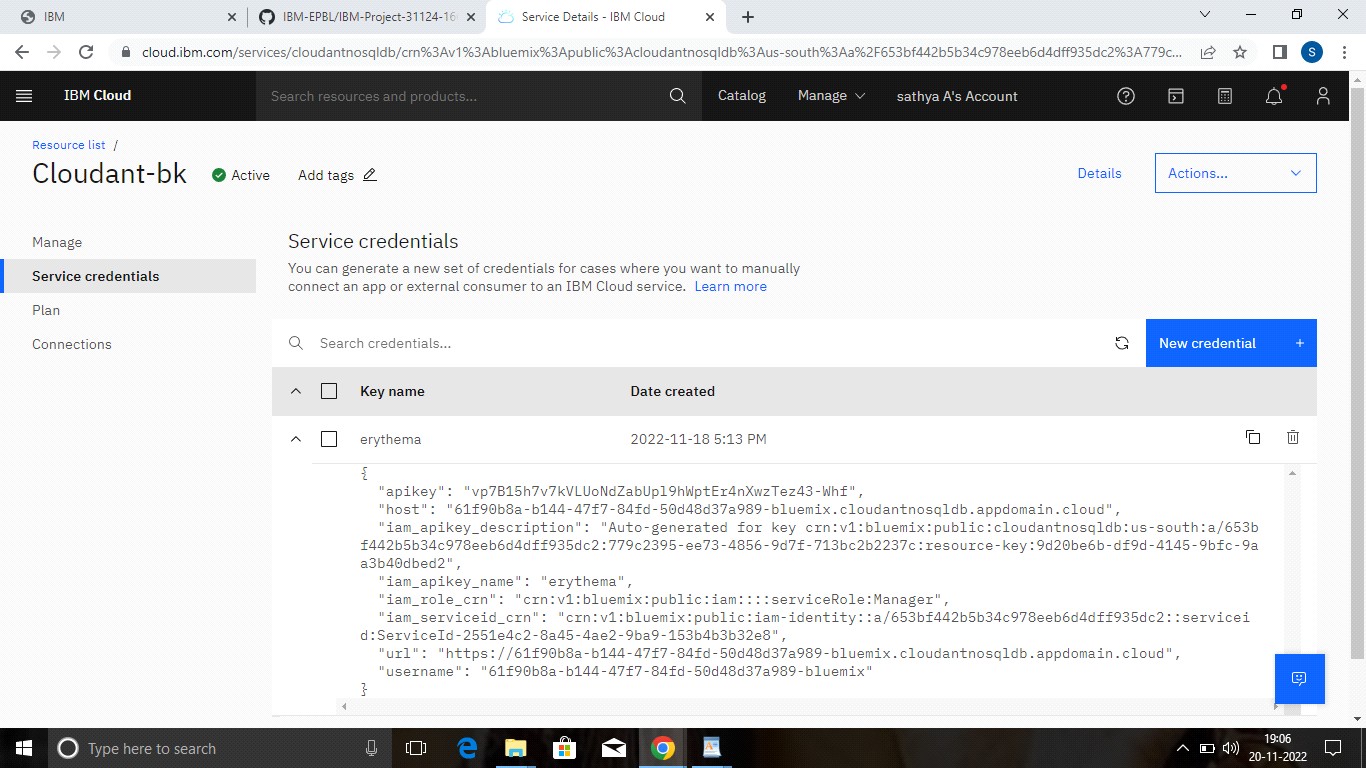
**REGISTER & LOGIN IBM CLOUD** :





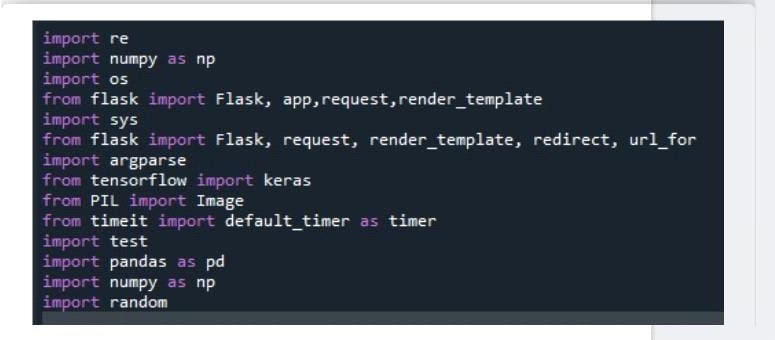
CREATE SERVICE CREDENTIALS :





BUILD PYTHON CODE :

### IMPORTING LIBRARIES :



Creating a function get\_parent\_dir() to get the parent directory so that we can go the required directory adding path to the parent directory.

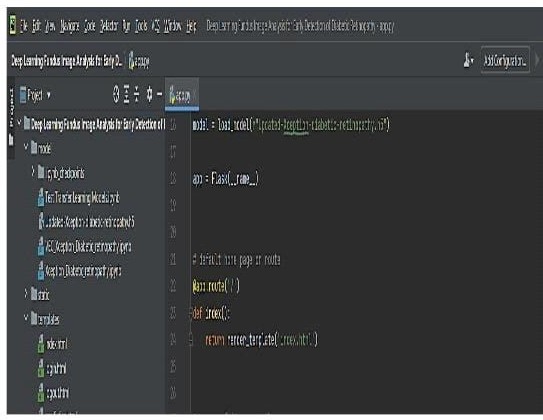


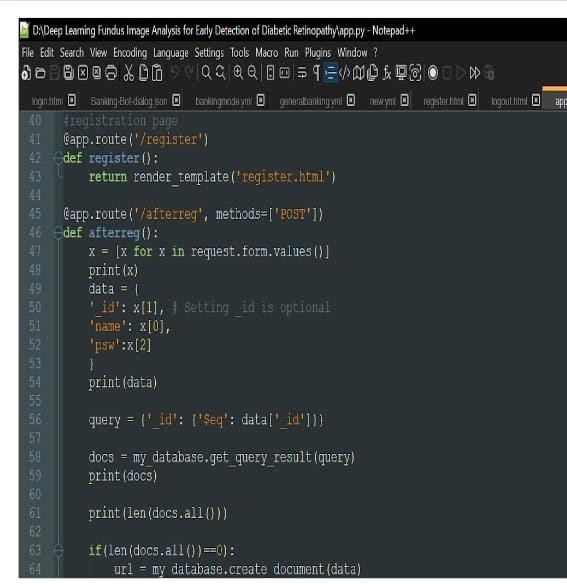
Go to the data folder in the project directory then go to test\_image folder and store the path to test image folder in a variable, similarly for the detection result folder and model weights folder also.

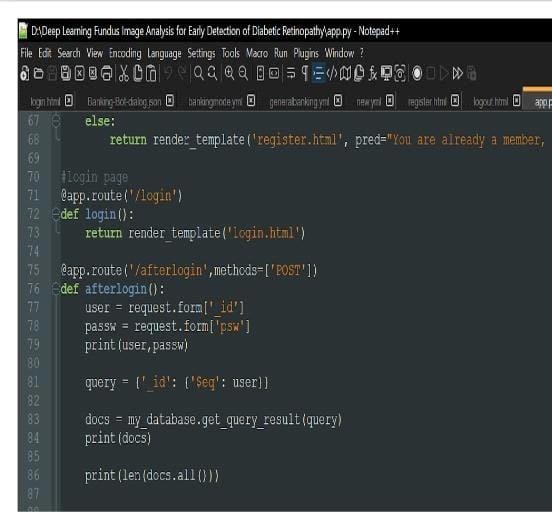


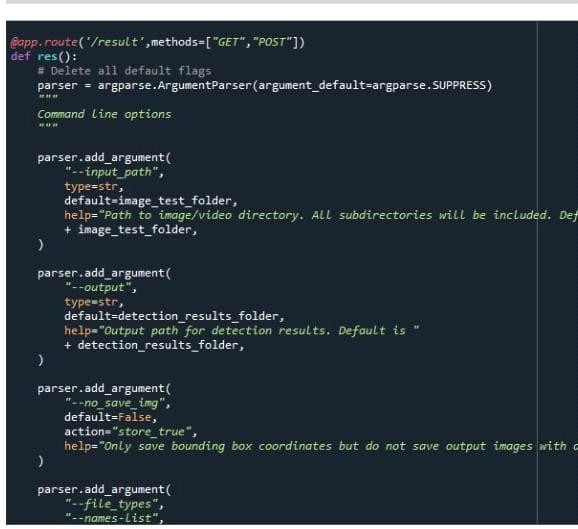
An object of Flask class is our WSGI application. Flask constructor takes the name of the current module ( name ) as argument.

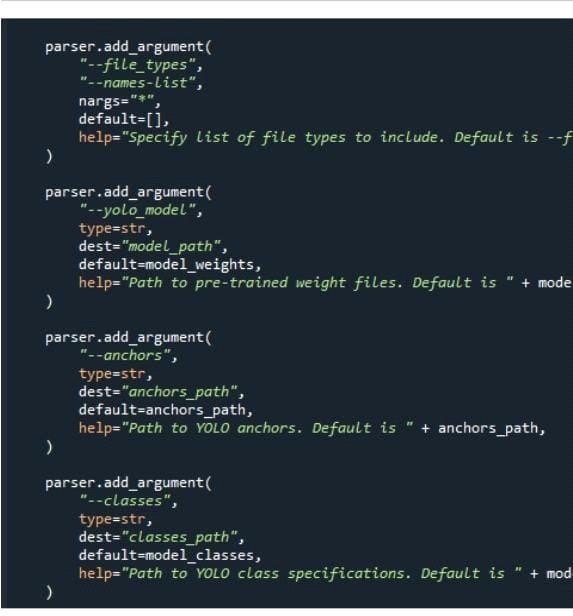


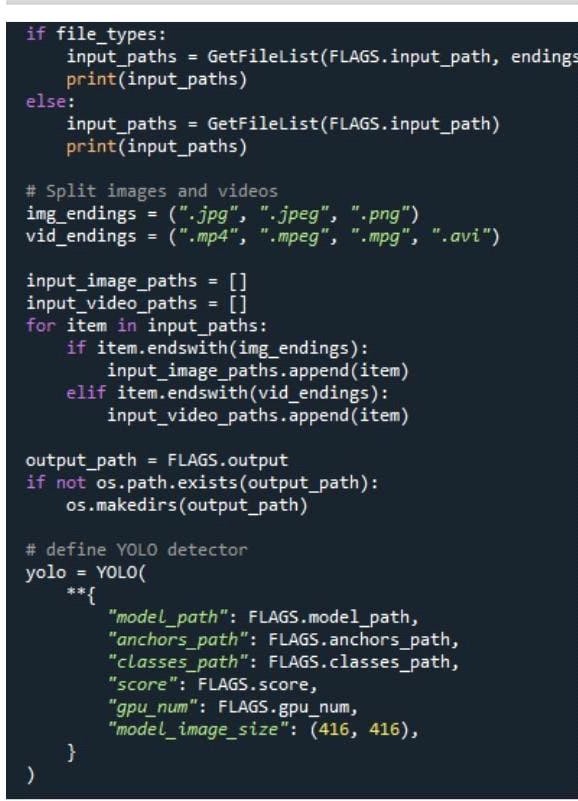


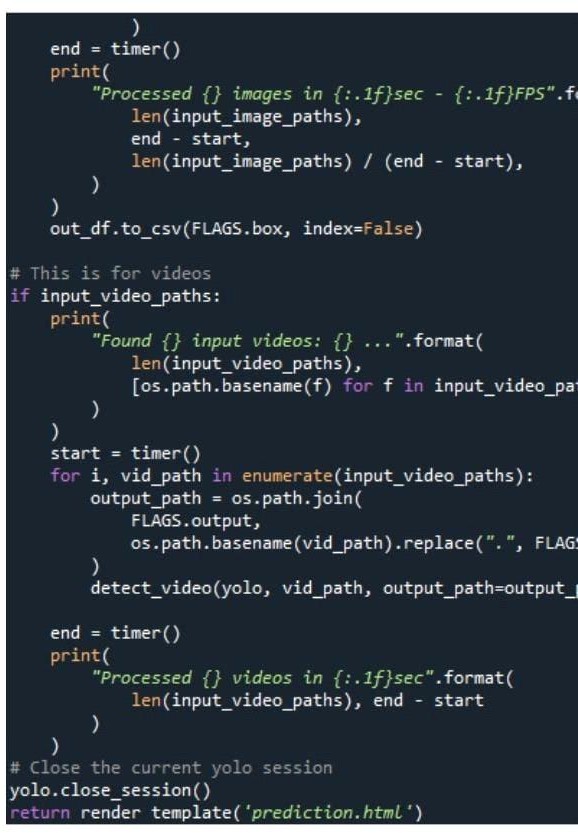














### BUILD HTML CODE:

* We use HTML to create the front-end part of the web page.
* Here, we have created 1 HTML page-Erythema.html.
* Erythema.html displays the home page.
* A simple HTML page is created.

### Run the application:

* Open anaconda prompt from the start menu.
* Navigate to the folder where your python script is.
* Now type “python app.py” command.
* Navigate to the localhost where you can view your web page.
* Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

## SOURCE CODE:

# -\*- coding: utf-8 -\*- """Untitled0.ipynb

Automatically generated by Colaboratory.

Original file is located at <https://colab.research.google.com/drive/1PYFZ7zKhWpFF5YilnguhZ8X1EgtSIJN4>

"""

import re

import numpy as np

import os

from flask import Flask, app,request,render\_template import sys

from flask import Flask, request, render\_template, redirect, url\_for import argparse

from tensorflow import keras from PIL import Image

from timeit import default\_timer as timer import test

from pyngrok import ngrok import pandas as pd import numpy as np import random

def get\_parent\_dir(n=1):

""" returns the n-th parent dicrectory of the current working directory """

current\_path = os.path.dirname(os.path.abspath( file )) for k in range(n):

current\_path = os.path.dirname(current\_path) return current\_path

src\_path=r'/content/drive/MyDrive/IBM\_PROJECT/yolo\_structure/2\_Training/src' print(src\_path) utils\_path=r'/content/drive/MyDrive/IBM\_PROJECT/yolo\_structure/Utils' print(utils\_path)

sys.path.append(src\_path) sys.path.append(utils\_path)

import argparse

from keras\_yolo3.yolo import YOLO, detect\_video from PIL import Image

from timeit import default\_timer as timer

from utils import load\_extractor\_model, load\_features, parse\_input, detect\_object import test

import utils

import pandas as pd import numpy as np

from Get\_File\_Paths import GetFileList import random

os.environ["TF\_CPP\_MIN\_LOG\_LEVEL"] = "3"

# Set up folder names for default values

data\_folder = os.path.join(get\_parent\_dir(n=1), "yolo\_structure", "Data")

image\_folder = os.path.join(data\_folder, "Source\_Images")

image\_test\_folder = os.path.join(image\_folder, "Test\_Images")

detection\_results\_folder = os.path.join(image\_folder, "Test\_Image\_Detection\_Results") detection\_results\_file = os.path.join(detection\_results\_folder, "Detection\_Results.csv")

model\_folder = os.path.join(data\_folder, "Model\_Weights")

model\_weights = os.path.join(model\_folder, "trained\_weights\_final.h5") model\_classes = os.path.join(model\_folder, "data\_classes.txt")

anchors\_path = os.path.join(src\_path, "keras\_yolo3", "model\_data", "yolo\_anchors.txt")

FLAGS = None

from cloudant.client import Cloudant

# Authenticate using an IAM API key client =

Cloudant.iam('ef7f4729-2486-45c5-a7fa-f4140373e2e6-bluemix','6GfFjs3engXLnSJB8Kp4f bs7HTKwrJpWJE7wNPGzZPVW', connect=True)

# Create a database using an initialized client my\_database = client.create\_database('my\_database')

app=Flask( name ) port\_no=5000

ngrok.set\_auth\_token("2H7aM94zEuTa40t3J6jKpIqWAc3\_B2UxzZs6qxetntgadxQW") public\_url = ngrok.connect(port\_no).public\_url

print(f"To acces the Gloable link please click {public\_url}")

#default home page or route @app.route('/')

def index():

return render\_template('index.html')

@app.route('/index.html')

def home():

return render\_template("index.html")

#registration page @app.route('/register') def register():

return render\_template('register.html')

@app.route('/afterreg', methods=['POST']) def afterreg():

x = [x for x in request.form.values()] print(x)

data = {

'\_id': x[1], # Setting \_id is optional 'name': x[0],

'psw':x[2]

}

print(data)

query = {'\_id': {'$eq': data['\_id']}}

docs = my\_database.get\_query\_result(query) print(docs)

print(len(docs.all()))

if(len(docs.all())==0):

url = my\_database.create\_document(data) #response = requests.get(url)

return render\_template('register.html', pred="Registration Successful, please login using your details")

else:

return render\_template('register.html', pred="You are already a member, please login using your details")

#login page @app.route('/login') def login():

return render\_template('login.html')

@app.route('/afterlogin',methods=['POST']) def afterlogin():

user = request.form['\_id'] passw = request.form['psw'] print(user,passw)

query = {'\_id': {'$eq': user}}

docs = my\_database.get\_query\_result(query) print(docs)

print(len(docs.all()))

if(len(docs.all())==0):

return render\_template('login.html', pred="The username is not found.")

else:

if((user==docs[0][0]['\_id'] and passw==docs[0][0]['psw'])): return redirect(url\_for('prediction'))

else:

print('Invalid User')

@app.route('/logout') def logout():

return render\_template('logout.html')

@app.route('/prediction') def prediction():

return render\_template('prediction.html',path="../static/img/6623.jpg",)

@app.route('/result',methods=["GET","POST"]) def res():

# Delete all default flags

parser = argparse.ArgumentParser(argument\_default=argparse.SUPPRESS) """

Command line options """

f = request.files['file'] f.save("./drive/MyDrive/IBM\_PROJECT/Flask/static/img/"+f.filename)

parser.add\_argument( "--input\_path", type=str,

default=image\_test\_folder,

help="Path to image/video directory. All subdirectories will be included.

Default is "

+ image\_test\_folder,

)

parser.add\_argument( "--output", type=str,

default=detection\_results\_folder,

help="Output path for detection results. Default is "

+ detection\_results\_folder,

)

parser.add\_argument( "--no\_save\_img", default=False,

action="store\_true",

help="Only save bounding box coordinates but do not save output images with annotated boxes. Default is False.",

)

parser.add\_argument( "--file\_types",

"--names-list", nargs="\*", default=[],

help="Specify list of file types to include. Default is --file\_types .jpg .jpeg .png

.mp4",

)

parser.add\_argument( "--yolo\_model", type=str,

dest="model\_path", default=model\_weights,

help="Path to pre-trained weight files. Default is " + model\_weights,

)

parser.add\_argument( "--anchors", type=str,

dest="anchors\_path", default=anchors\_path,

help="Path to YOLO anchors. Default is " + anchors\_path,

)

parser.add\_argument( "--classes", type=str,

dest="classes\_path", default=model\_classes,

help="Path to YOLO class specifications. Default is " + model\_classes,

)

parser.add\_argument(

"--gpu\_num", type=int, default=1, help="Number of GPU to use. Default is 1"

)

parser.add\_argument( "--confidence", type=float, dest="score", default=0.25,

help="Threshold for YOLO object confidence score to show predictions. Default

is 0.25.",

)

parser.add\_argument( "--box\_file", type=str, dest="box",

default=detection\_results\_file,

help="File to save bounding box results to. Default is "

+ detection\_results\_file,

)

parser.add\_argument( "--postfix", type=str, dest="postfix",

default="\_disease",

help='Specify the postfix for images with bounding boxes. Default is "\_disease"',

)

yolo = YOLO(

\*\*{

"model\_path": FLAGS.model\_path, "anchors\_path": FLAGS.anchors\_path, "classes\_path": FLAGS.classes\_path, "score": FLAGS.score,

"gpu\_num": FLAGS.gpu\_num, "model\_image\_size": (416, 416),

}

)

img\_path="/drive/MyDrive/IBM\_PROJECT/Flask/static/img/"+f.filename prediction, image,lat,lon= detect\_object(

yolo, img\_path,

save\_img=save\_img, save\_img\_path=FLAGS.output, postfix=FLAGS.postfix,

)

yolo.close\_session() return

render\_template('prediction.html',prediction=str(prediction),path="../static/img/"+f.filena me)

""" Running our application """ if name == " main ":

app.run(port=port\_no)

## OUTPUT:

