AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

PROJECT TEAM : A.SATHYA(TL).

R.SHALINI.

R.ELAKKIYA.

S.JAGAN KUMAR.

PROJECT LINK : https://github.com/IBM-EPBL/IBM-Project-31124-1660196395

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 If a drug is suspected, it must be withdrawn as soon as possible.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:

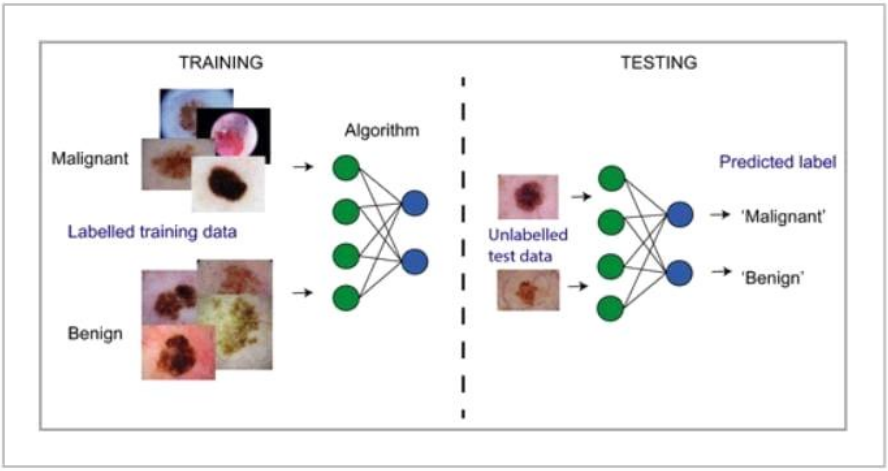
- See a doctor immediately if the redness.
- Appears suddenly or for no apparent reason.

- 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:

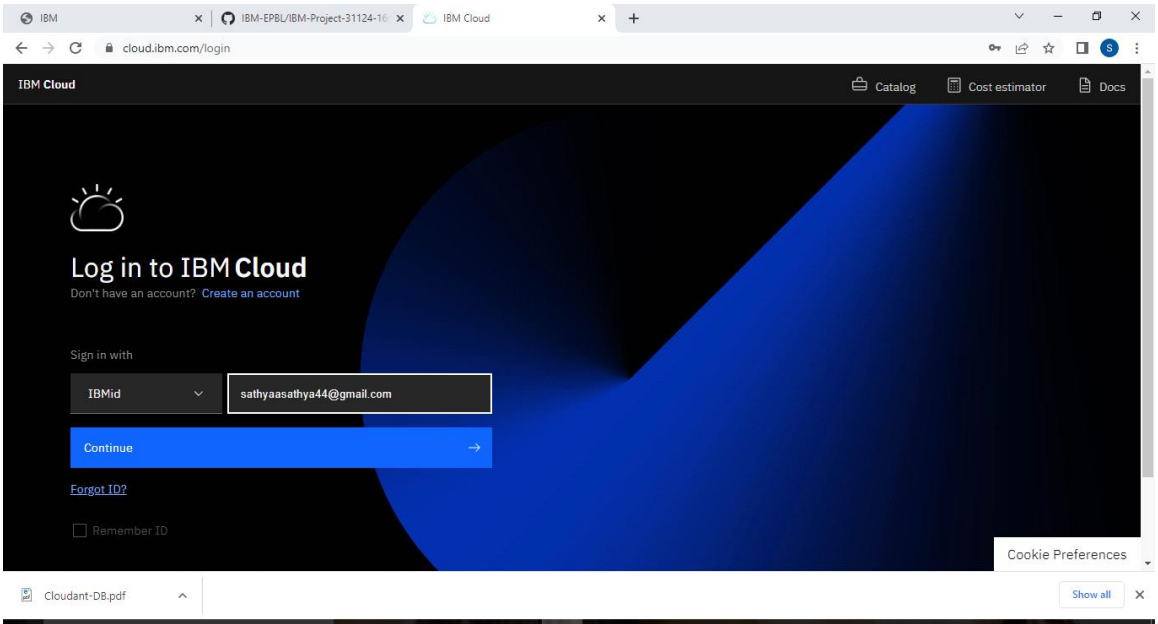
- Fatigue, flu-like symptoms (before lesions appear).
- Clusters of nodules (small round masses) and lesions on shins, forearms, thighs, and trunk.
- Red, painful lesions become soft and bluish, and fade to yellow and brown.
- Joint pain.
- Arthritis.

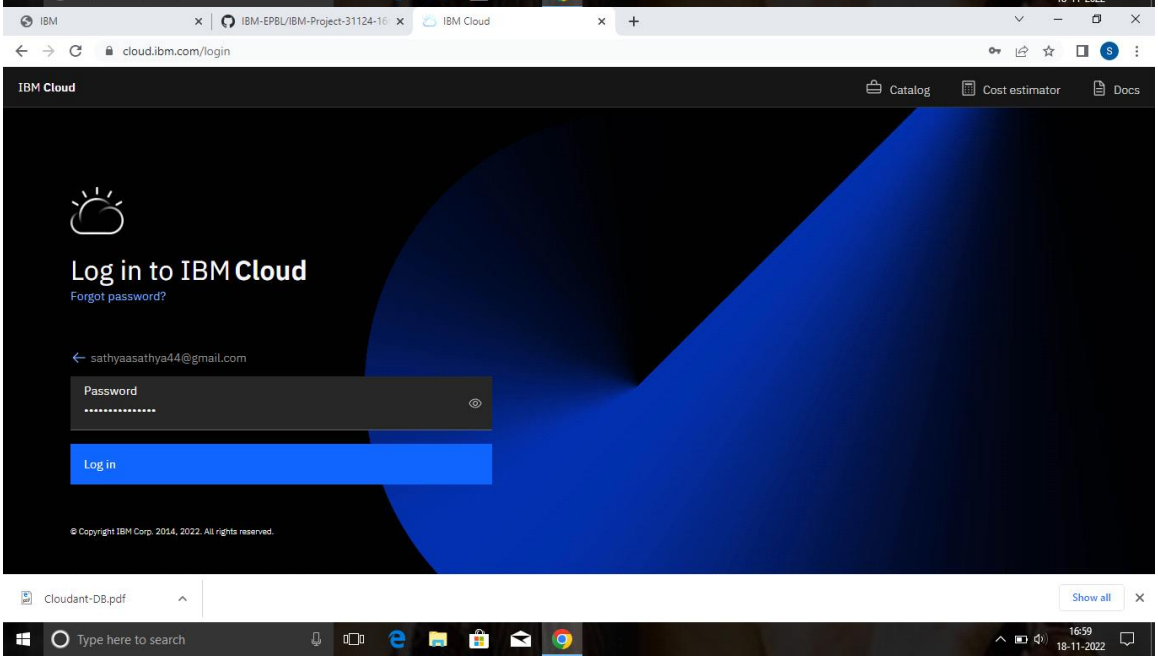
CREATE IBM SERVICE :

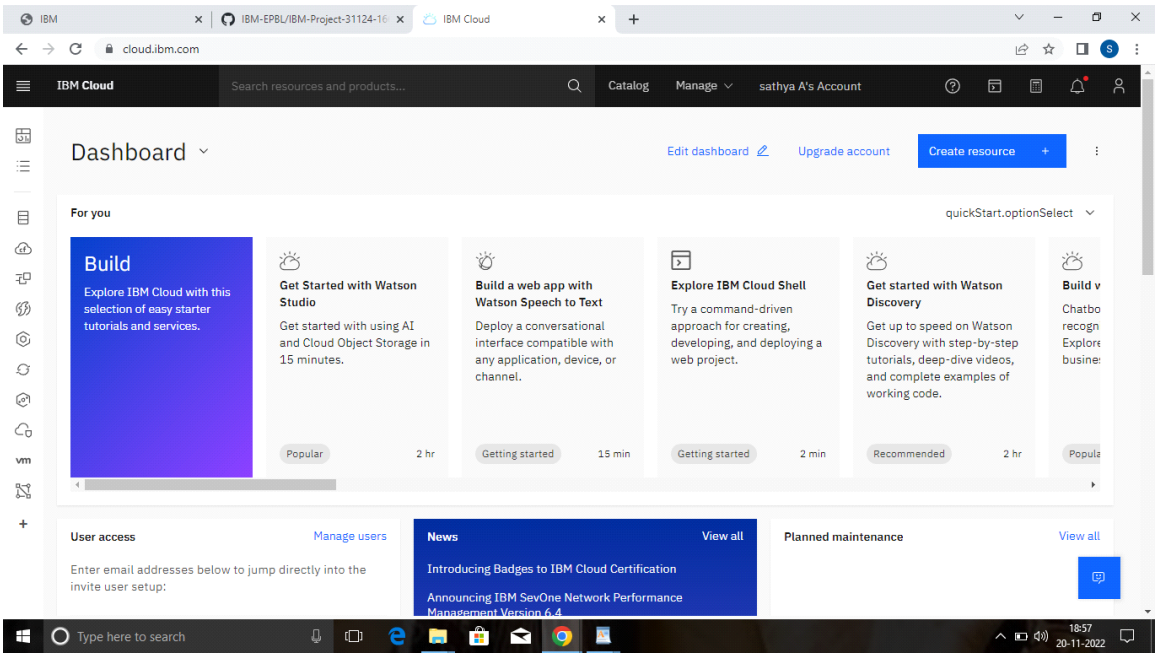
CREATE IBM CLOUD:

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

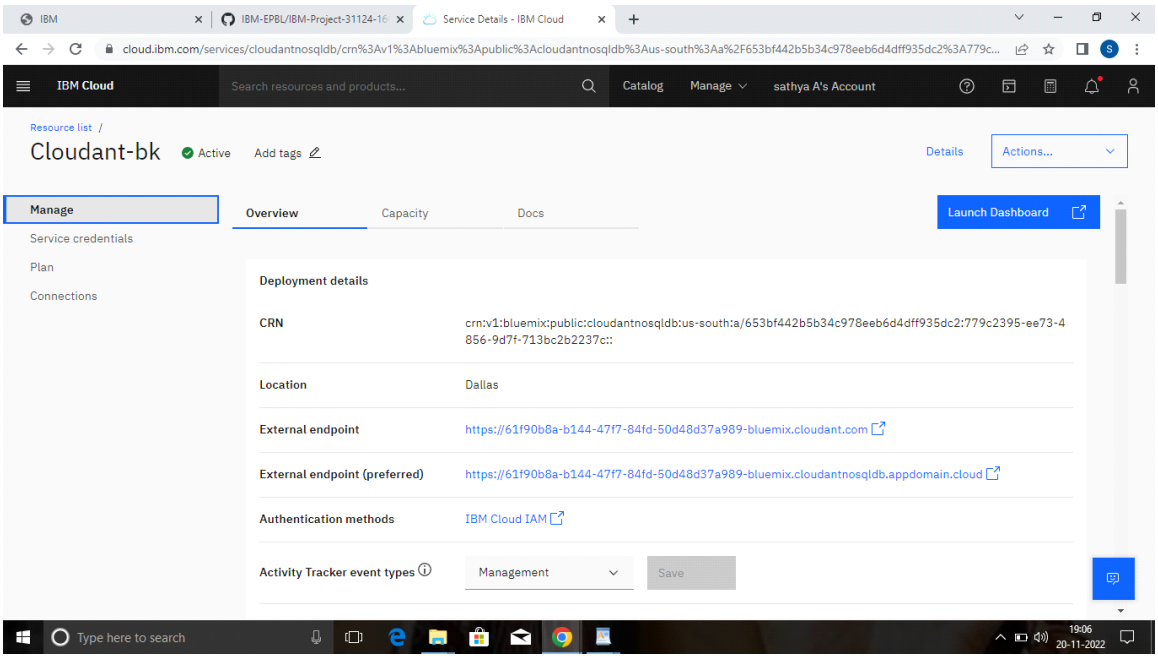
REGISTER & LOGIN IBM CLOUD :

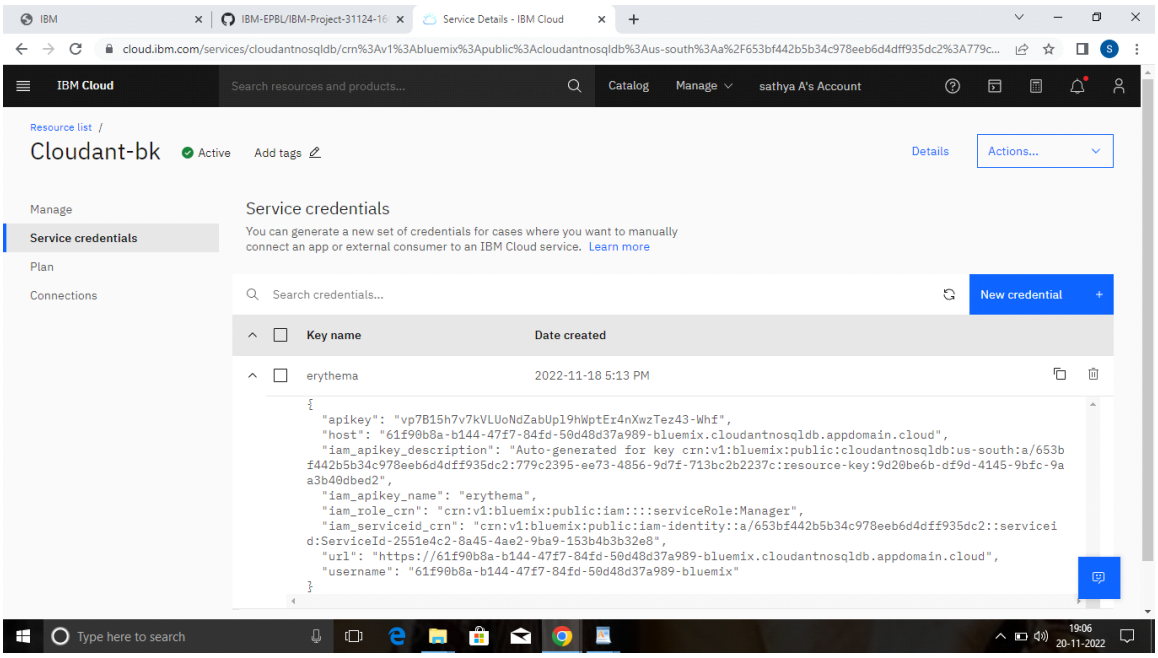






CREATE SERVICE CREDENTIALS :





BUILD PYTHON CODE :

IMPORTING LIBRARIES :

```
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
import pandas as pd
import numpy as np
import random
```

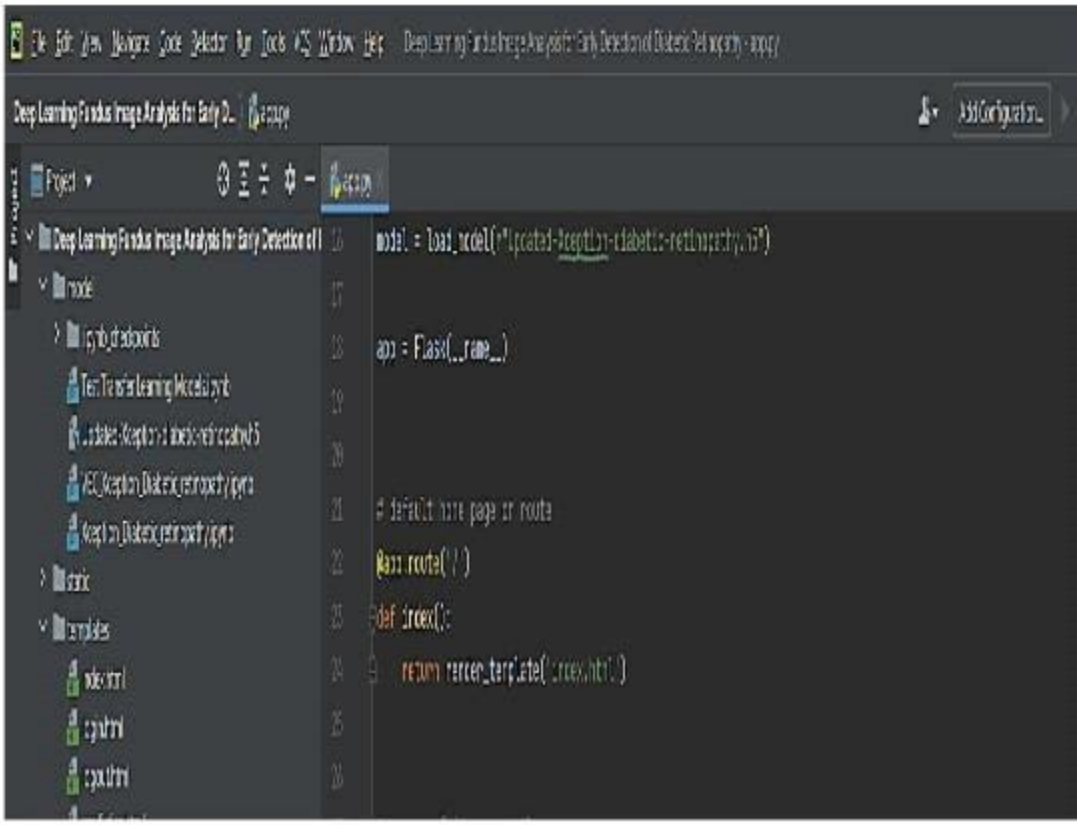
Creating a function get_parent_dir() to get the parent directory so that we can go the required directory adding path to the parent directory.

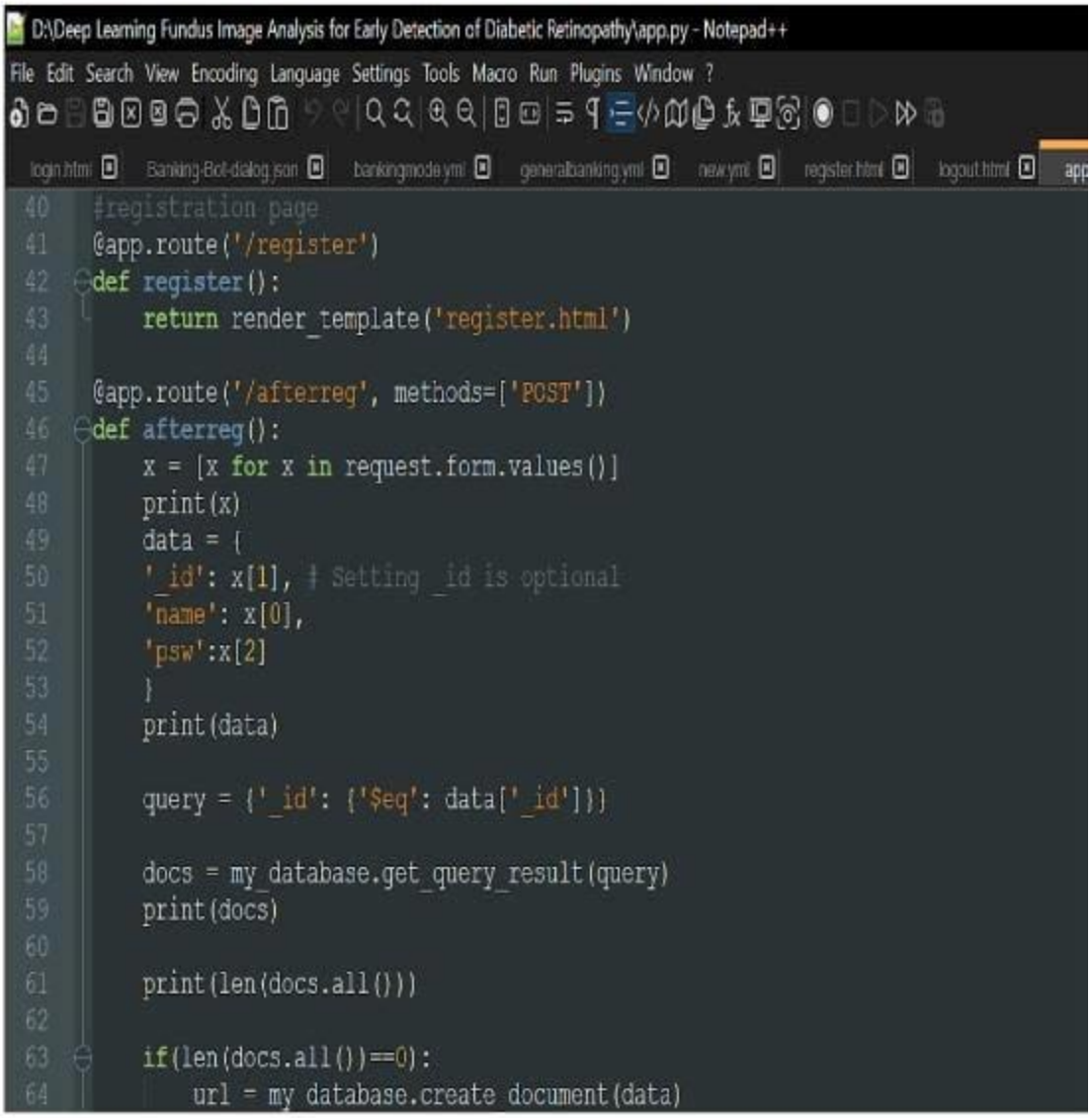
```
def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


src_path =r'C:\Users\HP\Desktop\Skin Disease-Flask\2_Training\src'
print(src_path)
utils_path = r'C:\Users\HP\Desktop\Skin Disease-Flask\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_objec
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), "Skin Disease-Flask", "Data")

image_folder = os.path.join(data_folder, "Source_Images")
```

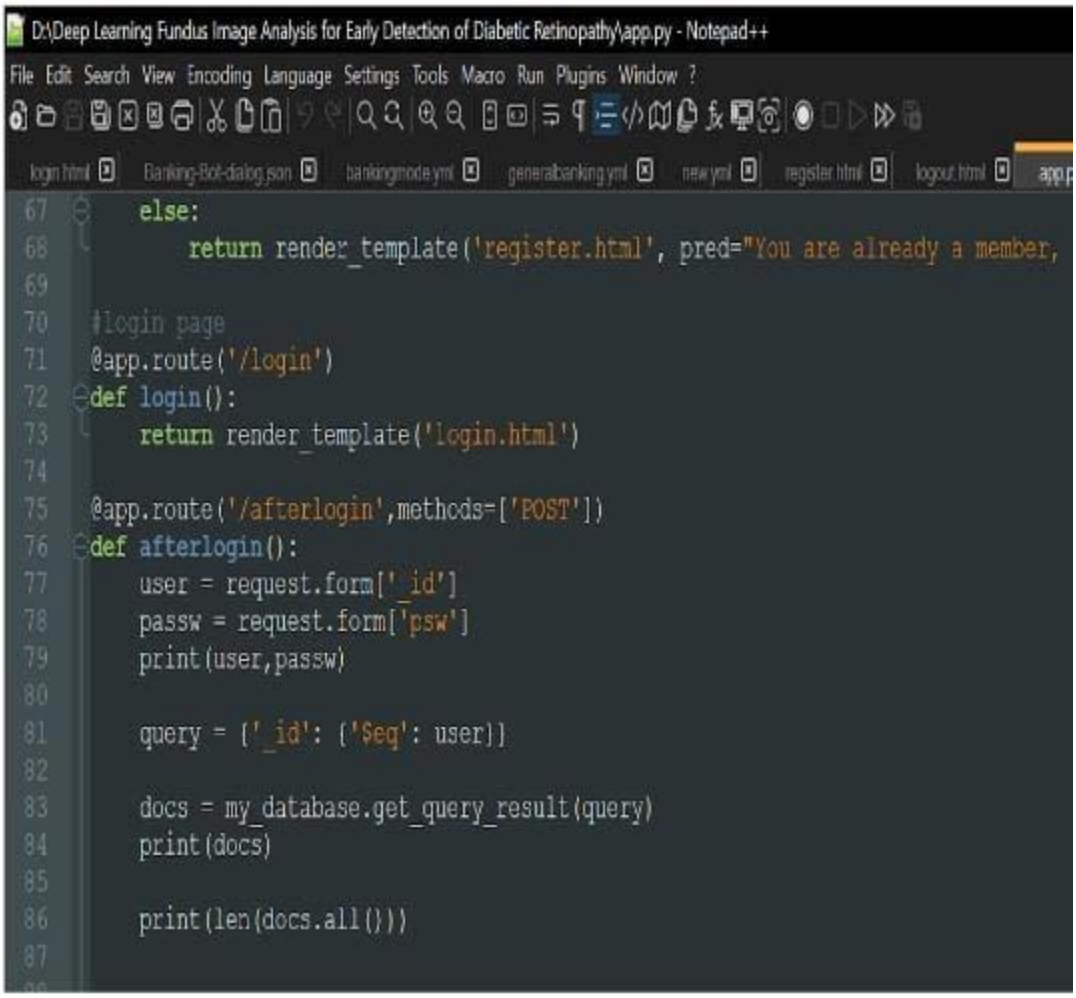
Go to the data folder in the project directory then go to test_image folder and store the path to test image folder in a variable, similarly for the detection result folder and model weights folder also.

```
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), "Skin Disease-Flask", "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
```

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


Project
app.py
model = load_model("updated-xception-diabetic-retinopathy.h5")
app = Flask(__name__)
# default home page or route
@app.route('/')
def index():
return render_template('index.html')


D:\Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy\app.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
login.html
Banking-Bot-dialog.json
bankingmode.yml
generalbanking.yml
new.yml
register.html
logout.html
app
#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)


D:\Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy\app.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
login.html
Banking-Bot-dialog.json
bankingmode.yml
generalbanking.yml
new.yml
register.html
logout.html
app.p
else:
return render_template('register.html', pred="You are already a member,
#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()))

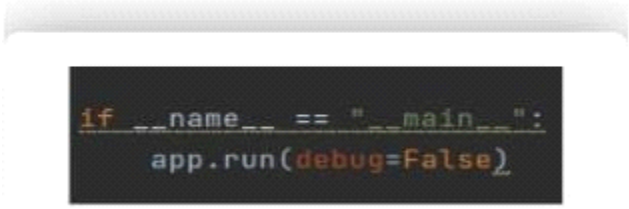
```python
@app.route('/result',methods=["GET","POST"])
def res():
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--input_path",
        type=str,
        default=image_test_folder,
        help="Path to image/video directory. All subdirectories will be included. Def
        + 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 c
    )

    parser.add_argument(
        "--file_types",
        "--names-list",
```

```
    parser.add_argument(
        "--file_types",
        "--names-list",
        nargs="*",
        default=[],
        help="Specify list of file types to include. Default is --f
    )

    parser.add_argument(
        "--yolo_model",
        type=str,
        dest="model_path",
        default=model_weights,
        help="Path to pre-trained weight files. Default is " + mode
    )

    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 " + mod
    )
```

```
if file_types:
    input_paths = GetFileList(FLAGS.input_path, endings
    print(input_paths)
else:
    input_paths = GetFileList(FLAGS.input_path)
    print(input_paths)

# Split images and videos
img_endings = (".jpg", ".jpeg", ".png")
vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")

input_image_paths = []
input_video_paths = []
for item in input_paths:
    if item.endswith(img_endings):
        input_image_paths.append(item)
    elif item.endswith(vid_endings):
        input_video_paths.append(item)

output_path = FLAGS.output
if not os.path.exists(output_path):
    os.makedirs(output_path)

# define YOLO detector
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),
    }
)
```

```
            )
    end = timer()
    print(
        "Processed {} images in {:.1f}sec - {:.1f}FPS".f
            len(input_image_paths),
            end - start,
            len(input_image_paths) / (end - start),
        )
    )
    out_df.to_csv(FLAGS.box, index=False)

# This is for videos
if input_video_paths:
    print(
        "Found {} input videos: {} ...".format(
            len(input_video_paths),
            [os.path.basename(f) for f in input_video_pa
        )
    )
    start = timer()
    for i, vid_path in enumerate(input_video_paths):
        output_path = os.path.join(
            FLAGS.output,
            os.path.basename(vid_path).replace(".", FLAG
        )
        detect_video(yolo, vid_path, output_path=output_

    end = timer()
    print(
        "Processed {} videos in {:.1f}sec".format(
            len(input_video_paths), end - start
        )
    )
# Close the current yolo session
yolo.close_session()
return render_template('prediction.html')
```

Main Function:



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 "
```

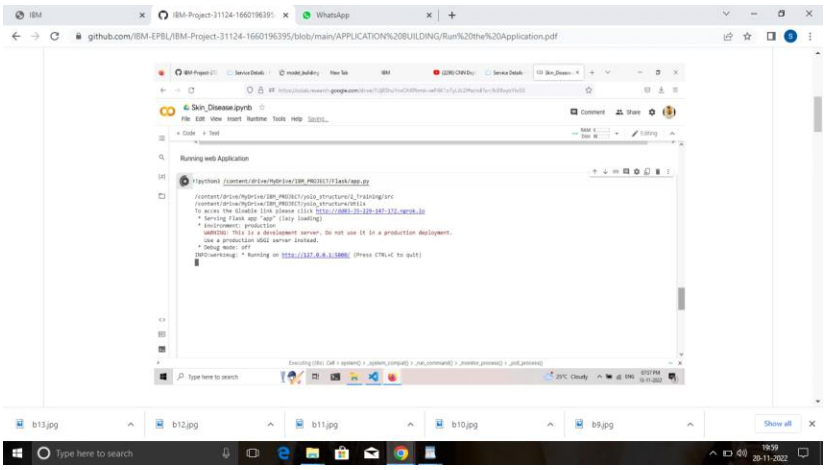
```
            + image_test_folder,
        )

        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,
```

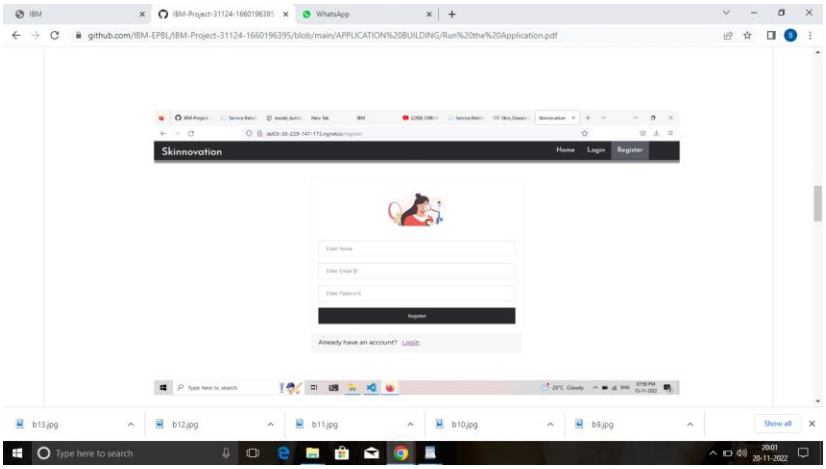
```
        dest="model_path",
        default=model_weights,
        help="Path to pre-trained weight files. Default is " + model_weights,
)

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,
```

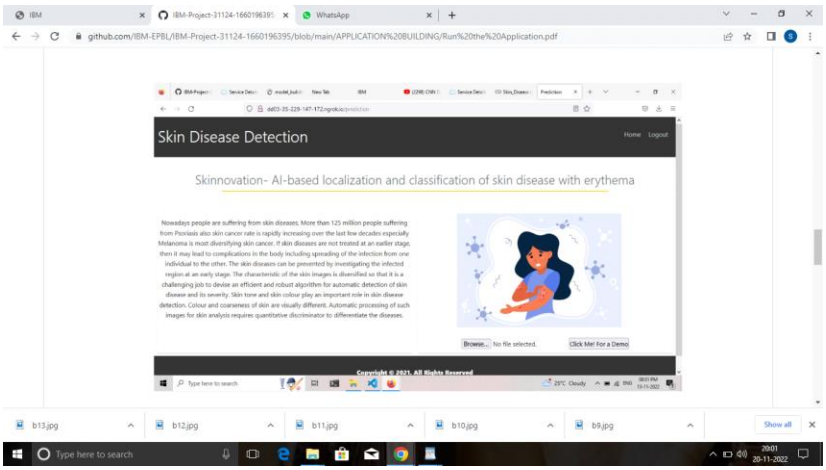
```
        help="Threshold for YOLO object confidence score to show predictions. Default is 0.25.",
    )

    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,
```

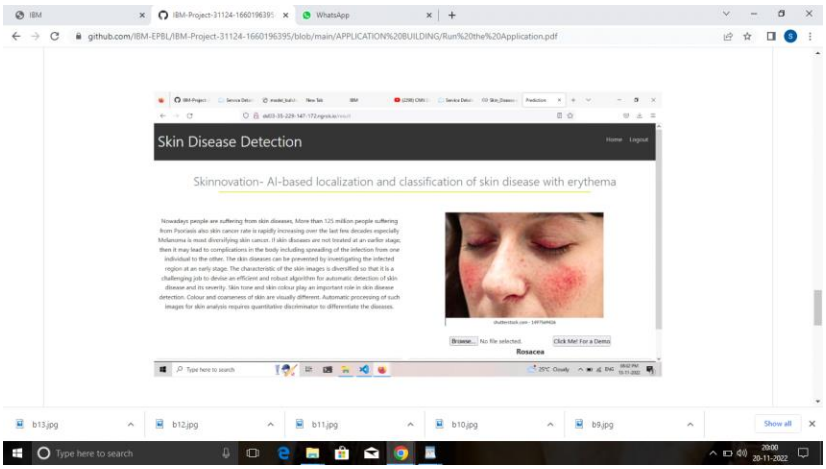
```
                "gpu_num": FLAGS.gpu_num,
                "model_image_size": (416, 416),
            }
        )


        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:


github.com/IBM-EPBL/IBM-Project-31124-1660196395/blob/main/APPLICATION%20BUILDING/Run%20the%20Application.pdf
Skin_Disease.ipynb
Running web Application
b13.jpg
b12.jpg
b11.jpg
b10.jpg
b9.jpg
Show all
Type here to search
19:59
20-11-2022


github.com/IBM-EPBL/IBM-Project-31124-1660196395/blob/main/APPLICATION%20BUILDING/Run%20the%20Application.pdf
Skinnovation
Home
Login
Register
Already have an account? Login
b13.jpg
b12.jpg
b11.jpg
b10.jpg
b9.jpg
Show all
Type here to search
20:01
20-11-2022


github.com/IBM-EPBL/IBM-Project-31124-1660196395/blob/main/APPLICATION%20BUILDING/Run%20the%20Application.pdf
Skin Disease Detection
Home
Logout
Skinnovation- AI-based localization and classification of skin disease with erythema
Browse...
No file selected.
Click Me! For a Demo
b13.jpg
b12.jpg
b11.jpg
b10.jpg
b9.jpg
Show all
Type here to search
20:01
20-11-2022


IBM
IBM-Project-31124-1660196395
WhatsApp
github.com/IBM-EPBL/IBM-Project-31124-1660196395/blob/main/APPLICATION%20BUILDING/Run%20the%20Application.pdf
Skin Disease Detection
Home
Logout
Skinnovation- AI-based localization and classification of skin disease with erythema
Nowadays people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.
Browse...
No file selected.
Click Me! For a Demo
Rosacea
Type here to search
b13.jpg
b12.jpg
b11.jpg
b10.jpg
b9.jpg
Show all
Type here to search
20:00
20-11-2022