```
import
numpy
as np
         import os
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image
         from tensorflow.keras.applications.inception_v3 import preprocess_input
         from flask import Flask, request,flash, render_template, redirect,url_for
         from cloudant.client import Cloudant
         from twilio.rest import Client
         model = load_model(r"Updated-xception-diabetic-retinopathy.h5")
         app = Flask(__name__)
         app.secret_key="abc"
         app.config['UPLOAD_FOLDER'] = "User_Images"
         # Authenticate using an IAM API key
         client = Cloudant.iam('d3ffc21a-c9d1-4276-a7c3-d7a48a949e1f-bluemix',
                                 'oS6rF9Lb8-d8IyJW4VEdHx5kiIN9ehQnNoj8ygKXFjzu', connect=True)
         # Create a database using an initialized client
         my_database = client.create_database('my_db')
         if my_database.exists():
             print("Database '{0}' successfully created.".format('my_db'))
         # default home page or route
         user = ""
         @app.route('/')
         def index():
             return render_template('index.html', pred="Login", vis ="visible")
         @ app.route('/index')
         def home():
             return render_template("index.html", pred="Login", vis ="visible")
         # registration page
         @ app.route('/register',methods=["GET","POST"])
         def register():
             if request.method == "POST":
                 name =  request.form.get("name")
                 mail = request.form.get("emailid")
                 mobile = request.form.get("num")
                 pswd = request.form.get("pass")
                 data = {
                     'name': name,
                     'mail': mail,
                     'mobile': mobile,
                     'psw': pswd
                 }
```

```
        print(data)
        query = {'mail': {'$eq': data['mail']}}
        docs = my_database.get_query_result(query)
        print(docs)
        print(len(docs.all()))
        if (len(docs.all()) == 0):
            url = my_database.create_document(data)
            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 ")
    else:
        return render_template('register.html')

@ app.route('/login', methods=['GET','POST'])
def login():
    if request.method == "GET":
        user = request.args.get('mail')
        passw = request.args.get('pass')
        print(user, passw)
        query = {'mail': {'$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="")
        else:
            if ((user == docs[0][0]['mail'] and passw == docs[0][0]['psw'])):
                flash("Logged in as " + str(user))
                return render_template('index.html', pred="Logged in as "+str(user),
vis ="hidden", vis2="visible")
            else:
                return render_template('login.html', pred="The password is wrong.")
    else:
        return render_template('login.html')
@ app.route('/logout')
def logout():
    return render_template('logout.html')
@app.route("/predict",methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        f = request.files['file']
        # getting the current path 1.e where app.py is present
        basepath = os.path.dirname(__file__)
        #print ( " current path " , basepath )
        # from anywhere in the system we can give image but we want that
        filepath = os.path.join(str(basepath), 'User_Images', str(f.filename))
        #print ( " upload folder is " , filepath )
        f.save(filepath)
        img = image.load_img(filepath, target_size=(299, 299))
        x = image.img_to_array(img)  # ing to array
        x = np.expand_dims(x, axis=0)  # used for adding one more dimension
        #print ( x )
        img_data = preprocess_input(x)
        prediction = np.argmax(model.predict(img_data), axis=1)
        index = [' No Diabetic Retinopathy ', ' Mild NPDR ',
                 ' Moderate NPDR ', ' Severe NPDR ', ' Proliferative DR ']
        result = str(index[prediction[0]])
        print(result)
        account_sid = 'ACe84a385fa5539d372c1a924452f489a3'
        auth_token = '359788a4ddfb510ac8ecd2fa948b924e'

        client = Client(account_sid, auth_token)

        ''' Change the value of 'from' with the number
        received from Twilio and the value of 'to'
        with the number in which you want to send message.'''
        message = client.messages.create(
                                    from_='+17088347950',
                                    body ='Results: '+ result,
                                    to ='+919500680243'
                                )
        return render_template('prediction.html', prediction=result, fname =
filepath)
    else:
        return render_template("prediction.html")
if __name__ == "__main__":
    app.debug = True
    app.run()
```