Team ID	PNT2022TMID52603
Project Name	A Novel Method for Handwritten Digit Recognition System

Bulid python PART-1

MODEL CREATION:

```
from keras.datasets import mnist
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Dense,Flatten from
tensorflow.keras.optimizers import Adam (X_train,y_train),(

X_test,y_test) =mnist.load_data() print(X_train.shape) print(X_test.shape)
print(y_test.shape) print(y_train.shape)

print("The label value is ",y_test[10]) #Value in y_test plt.imshow(X_test[10])
print("The label value is ",y_test[65]) #Value in y_test plt.imshow(X_test[65])

X_train.shape X_test.shape

X_train1 = X_train.reshape(60000, 28, 28, 1).astype('float32') X_test1 =
X_test.reshape(10000, 28, 28, 1).astype('float32') number_of_classes= 10
y_train1 = np_utils.to_categorical(y_train,number_of_classes)
```

```
y_test1 = np_utils.to_categorical(y_test,number_of_classes) print("After
encoding the value",y_test[10] ,"become", y_test1[10]) print("After encoding the
value",y_test[100] ,"become", y_test1[100])

print("After encoding the value",y_test[65] ,"become", y_test1[65]) model =
Sequential()
```

```
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
model.add(Conv2D(32, (3, 3), activation="relu"))

model.add(Flatten())
model.add(Dense(number_of_classes, activation="softmax"))

model.compile(loss='categorical_crossentropy', optimizer="Adam",
metrics=["accuracy"])

model.fit(X_train1, y_train1, batch_size=32, epochs=5,
validation_data=(X_test1,y_test1)) metrics = model.evaluate(X_test1, y_test1,
verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)

prediction = model.predict(X_test1[:4]) print(prediction)
import numpy as np print(np.argmax(prediction, axis=1)) print(y_test1[:4])
model.save("model.h5")

from tensorflow.keras.models import load_model
model=load_model("model.h5")

model.summary()
```

FLASK APP:

```
import pickle

import sklearn

from flask import Flask, render_template,request,redirect,url_for,flash

from flask_bootstrap import Bootstrap

from flask_sqlalchemy import SQLAlchemy

import numpy as np

from werkzeug.security import generate_password_hash,check_password_hash

import os

import cv2

from skimage import feature
```

```
from PIL import Image, ImageOps
from keras.preprocessing import image
from keras import models
from keras.models import load_model
from keras.preprocessing import image
from flask_login import
login_user,logout_user,LoginManager,UserMixin,current_user,login_required
app = Flask(_name_)
app.config['SECRET_KEY'] = '8BYkEfBA6O6donzWlSihBXox7C0sKR6b'
app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///database.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
Bootstrap(app)
login_manager = LoginManager()
login_manager.init_app(app)
class users(UserMixin,db.Model):
    id = db.Column(db.Integer,primary_key=True)
    email= db.Column(db.String(200),nullable=False)
    password = db.Column(db.String(300),nullable=False)
    name = db.Column(db.String(100),nullable=False)
@login_manager.user_loader
def user_load(id):
    return users.query.get(int(id))
@app.route("/")
def home():
```

```
    return render_template("index1.html")
@app.route("/register",methods=['GET','POST'])
def register():
    if request.method == 'POST':
        if users.query.filter_by(email=request.form['email']).first():
            flash('User already registered')
            return redirect(url_for('login'))
        else:
         password =
generate_password_hash(request.form['password'],method="pbkdf2:sha256",sa
lt_length=8)
         user = users(
            email = request.form['email'],
            password = password,
            name = request.form['name']
         )
         db.session.add(user)
         db.session.commit()
         return redirect(url_for('home'))
    return render_template('register.html')
@app.route("/login",methods=['GET','POST'])
def login():
    if request.method == 'POST':
         email= request.form['email']
         password = request.form['password']
```

```
        k=users.query.filter_by(email=email).first()
        if not k:
            flash('User not registered')
            return redirect(url_for('login'))
        elif check_password_hash(k.password,password):
            login_user(k)
            return redirect(url_for('model'))
        else:
            flash('Wrong password')
            return redirect(url_for('login'))

    return render_template('login.html')
@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('home'))
@app.route("/digit")
def model():
    return render_template('index.html')
def quantify_image(image):
  features = feature.hog(image,orientations=9,

pixels_per_cell=(10,10),cells_per_block=(2,2),transform_sqrt=True,block_norm="L1")
```

```
  return features
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']  # requesting the file
        basepath = os.path.dirname(os.path.realpath('_file_'))  # storing the file directory
        filepath = os.path.join(basepath, "uploads", f.filename)  # storing the file in uploads folder
        f.save(filepath)
        image = cv2.imread(filepath)
        model = load_model("MNIST_model.h5")
        img = Image.open(filepath).convert("L")
        img = img.resize((28, 28))
        img2arr = np.array(img)
        img2arr = img2arr.reshape(1, 28, 28, 1)
        results = model.predict(img2arr)
        results = np.argmax(results, axis=1)
        result =str(results)
        return result

    return None
admin=[1]
if _name_ == '_main_':
    app.run(debug=True)
```