Importing all the basic libraries in Python

```
import NumPy as np
import pandas as PD
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt, matplotlib.image as mpimg
from sklearn import svm
%matplotlib inline
```

Here the route for prediction is given and necessary steps are performed to get the predicted output.

```
    if(y_pred == 0) :
      return render_template("0.html",showcase =  str(y_pred))
    elif(y_pred == 1) :
      return render_template("1.html",showcase =  str(y_pred))
    elif(y_pred == 2) :
      return render_template("2.html",showcase =  str(y_pred))
    elif(y_pred == 3) :
      return render_template("3.html",showcase =  str(y_pred))
    elif(y_pred == 4) :
      return render_template("4.html",showcase =  str(y_pred))
    elif(y_pred == 5) :
      return render_template("5.html",showcase =  str(y_pred))
    elif(y_pred == 6) :
      return render_template("6.html",showcase =  str(y_pred))
    elif(y_pred == 7) :
      return render_template("7.html",showcase =  str(y_pred))
    elif(y_pred == 8) :
      return render_template("8.html",showcase =  str(y_pred))
    else :
      return render_template("9.html",showcase =  str(y_pred))
else:
     return None
```

Importing the train and test data

```
train_df=pd.read_csv('trainDG.csv')
test_df=pd.read_csv('testDG.csv')
```

For now, let's try it out on the first 5000 images

```
= train_df.iloc[0:5000,1:]
labels = train_df.iloc[0:5000,:1]
train_images, test_images,train_labels, test_labels =
train_test_split(images, labels, train_size=0.8,
random_state=0)
```

Applying the SVM model

```
 clf = SVM.SVC()
clf.fit(train_images, train_labels.values.ravel())
clf.score(test_images,test_labels)
```

```
In [9]: clf = svm.SVC()
        clf.fit(train_images, train_labels.values.ravel())
        clf.score(test_images,test_labels)

        C:\Users\DELL\Anaconda3\lib\site-packages\sklearn\s
        'auto' to 'scale' in version 0.22 to account better
        his warning.
          "avoid this warning.", FutureWarning)

Out[9]: 0.887
```

Prediction for test data

```
 est_data=pd.read_csv('testDG.csv')
test_data[test_data>0]=1
results=clf.predict(test_data[0:5000])
```

Generating the output file(if required)

```
df = PD.DataFrame(results)
df. index.name='ImageId'
df.index+=1
df.columns=['Label']
df.to_csv('sample_submission.csv', header=True)
```

The accuracy for this model is 88.7% which is better compared to K-NN implementation. I have considered only the first 5000 images as it eases implementation and enables better accuracy with desired results.

Necessary conditions are given according to the input classes and the app will be returning the templates according to that.
Main Function:
This function runs your app in a web browser
Lastly, we run our app on the local host. Here we are running it on localhost:8000.