Application Building

Here, building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building server-side script

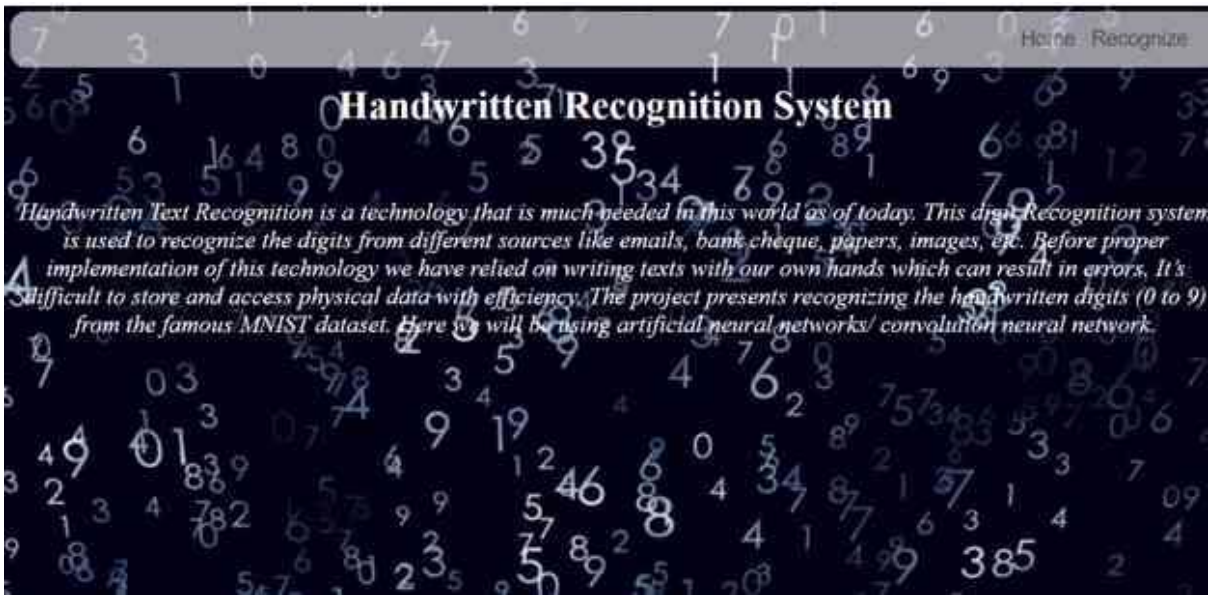
Create An HTML File

- We use HTML to create the front end part of the web page.
- Here, we created 2 html pages- index.html, web.html.
- index.html displays home page.
- web.html accepts the values from the input and displays the prediction.
- For more information regarding HTML refer the link below

Please refer to the link for HTML code files

Let's see how our index.html file looks like

This is the main page which describes about the project and summarizes it.



This is the prediction page where we get to choose the image from our local system and predict the output.

This is the prediction page where we get to choose the image from our local system and predict the output.

Build Python Code (Part 1)

Let us build the flask file 'app.py' which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application.

- App starts running when the "__name__" constructor is called in main.
- render_template is used to return HTML file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.

Import Libraries:

```
from flask import Flask, render_template, request# Flask-It is our framework which we are going to use to
run/serve our application.
#request-for accessing file which was uploaded by the user on our application.

from PIL import Image #used for manipulating image uploaded by the user.
import numpy as np #used for numerical analysis
from tensorflow.keras.models import load_model#to load our model trained with MNIST data
import tensorflow as tf#to run our model.
```

Libraries required for the app to run are to be imported.

Routing to the html Page

```
@app.route('/') #default route
def upload_file():
    return render_template('main.html') #rendering html page
@app.route('/about') #Main page route
def upload_file1():
    return render_template('main.html') #rendering html page
@app.route('/upload') #main page route
def upload_file2():
    return render_template('index6.html')
```

We are routing the app to the HTML templates which we want to render.
Firstly we are rendering the main.html template and from there we are navigating to our prediction page that is index6.html

Returning the prediction on UI:

```
@app.route('/predict', methods = ['POST']) #route for our prediction
def upload_image_file():
    if request.method == 'POST':
        img = Image.open(request.files['file'].stream).convert("L")# convert image to monochrome
        img = img.resize((28,28))# resizing of input image
        im2arr = np.array(img)#converting to image
        im2arr = im2arr.reshape(1,28,28,1) #reshaping according to our requirement
        y_pred = model.predict_classes(im2arr) #predicting the results
        print(y_pred) #printing our result in prompt
        #return 'Predicted Number: ' + str(y_pred) #returning our output
```

Build Python Code (Part 2)

Here the route for prediction is given and necessary steps are performed in order 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
```

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 localhost. Here we are running it on localhost:8000

```
    else:
        return None
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000,debug=True)
    #app.run(debug = True) #running our flask app
```

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

```
(base) C:\Users\DELL>cd C:\Users\DELL\Hand_Written_Final

(base) C:\Users\DELL\Hand_Written_Final>python app.py
```

Navigate to the localhost where you can view your web page
Upload an image and see the predicted output on UI your page and output looks like: