PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

Submitted by

PNT2022TMID43387

PRAKALYA E – 715519106035

ASWATHI R – 715519106006

YAZHINI KANNAN – 715519106060

SHAKTHI K – 715519106044

INDEX

13. APPENDIX

Source Code

GitHub & Project Demo Link

INTRODUCTION

1.1 PROJECT OVERVIEW

Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. Because of the progress in the field of science and technology, everything is being digitized to reduce human effort. Hence, there comes a need for handwritten digit recognition in many real-time applications. The MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. We use Artificial neural networks to train these images and build a deep learning model. Web application is created where the user can upload an image of a handwritten digit. This image is analyzed by the model and the detected result is returned to the UI.

1.2 PURPOSE

Digit recognition systems can recognize digits from a variety of sources. They have many real time applications. Some of them are

- Form data entry
- Mail sorting
- Bank cheque processing
- Automatic license plate recognition

LITERATURE SURVEY

2.1 PROBLEM STATEMENT

With the large number of the hand-written documents, there is a great demand to convert the handwritten documents into digital record copies, which are more accessible through digital systems, such as digital forms and databases. To automatically accomplish the transformation from handwritten numbers to their digital version, a digit recognition system is inevitable and useful.

2.2 REFERENCES

1. **A novel hybrid CNN–SVM classifier for recognizing handwritten digits** (2017)

   Xiao-Xiao Niu, Ching Y.Suen

   Inference:

   It is a hybrid CNN–SVM model for handwritten digit recognition. This model automatically retrieves features based on the CNN architecture, and recognizes the unknown pattern using the SVM recognizer.

2. **Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization** (2018)

   Shruti R.Kulkarni, Bipin Rajendran

   Inference:

   This model is a highly compact and efficient 3-layer spiking neural network for identifying handwritten digits. It has an accuracy of 98.17% on the MNIST dataset using the NormAD learning algorithm.

3. MDig: Multi-digit Recognition using Convolutional Neural Network on Mobile (2019)

Xuan Yang, Jing Pu

Inference:

CNN tends to be the solution for any handwriting recognition. To reduce the workload, the shallow CNN is trained. Segmentation and processing are done to reduce input size fed into CNN. On the NVIDIA SHIELD tablet, the application processes a frame and extracts 32 digits in approximately 60ms, and batching the fully-connected layers reduces the CNN runtime by another 12%.

4. An efficient and improved scheme for handwritten digit recognition based on convolutional neural network (2019)

Saqib Ali, Zeeshan Shaukat, Muhammad Azeem, Zareen Sakhawat, Tariq Mahmood, Khalil ur Rehman

Inference:

The conventional algorithms used for handwriting recognition uses character recognition and feature extraction, but has very low accuracy and low computational speed. With the use of CNN(Convolutional Neural Networks) as classifier, MNIST as data set and DL4J for testing, the above system has proven to increase the accuracy of the system by 99.21% and also increases the computational speed.

5. Hybrid CNN-SVM Classifier for Handwritten Digit Recognition (2020)

Savita Ahlawat, Amit Choudhary

Inference:

The conventional algorithms used for handwriting recognition uses character recognition and feature extraction, but has very low accuracy and low computational speed. With the use of CNN(Convolutional Neural Networks) as classifier, MNIST as data set and DL4J for testing, the above system has proven to increase the accuracy of the system by 99.21% and also increases the computational speed.

6. **An adaptive deep Q-learning strategy for handwritten digit recognition**(2018)

Junfei Qiao, Gongming Wang, Wenjing Li, Min Chen

Inference:

To increase the accuracy and decrease the running time, we employ an adaptive deep Q-learning strategy. The mentioned strategy uses feature extraction and decision making to form a Q-deep belief Network(Q-ADBN). Q-ADBN is responsible for the feature extraction from the handwriting.

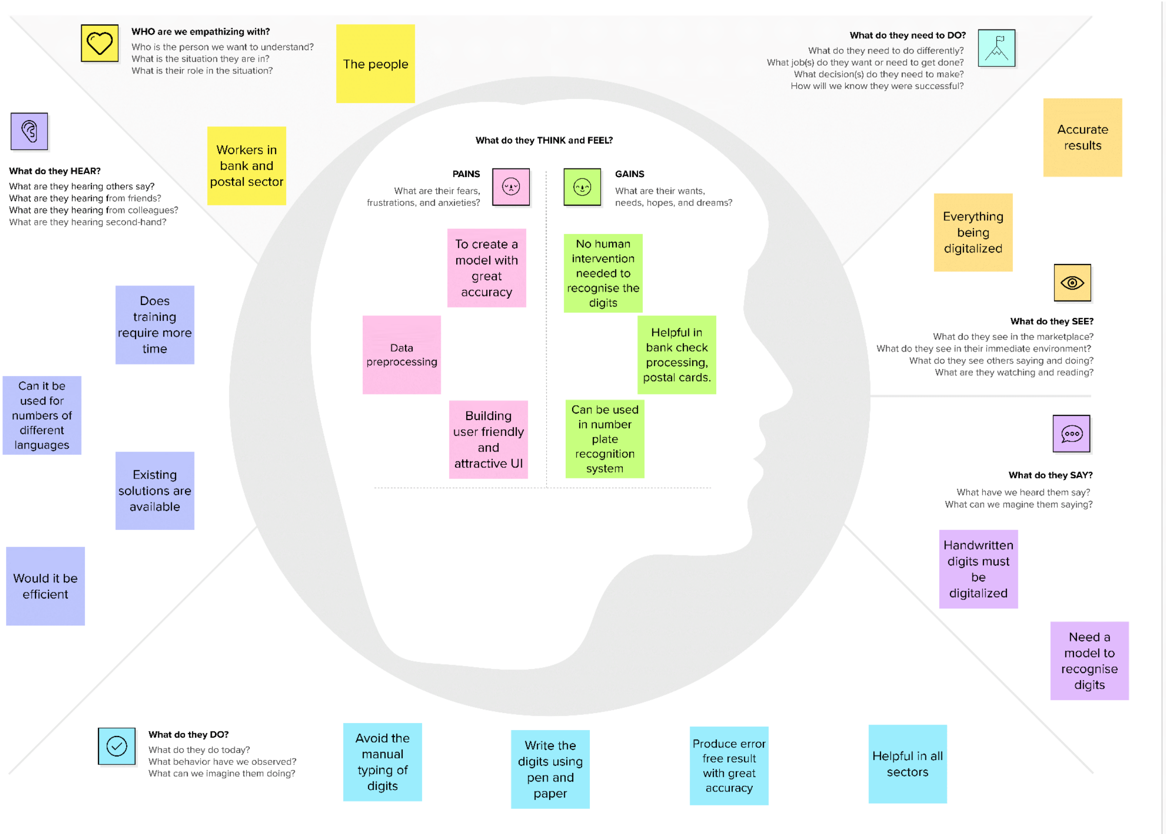
7. **Recognition of Handwritten Digit using Convolutional Neural Network (CNN)**(2019)

Fathma Siddique; Shadman Sakib; Md. Abu Bakr Siddique

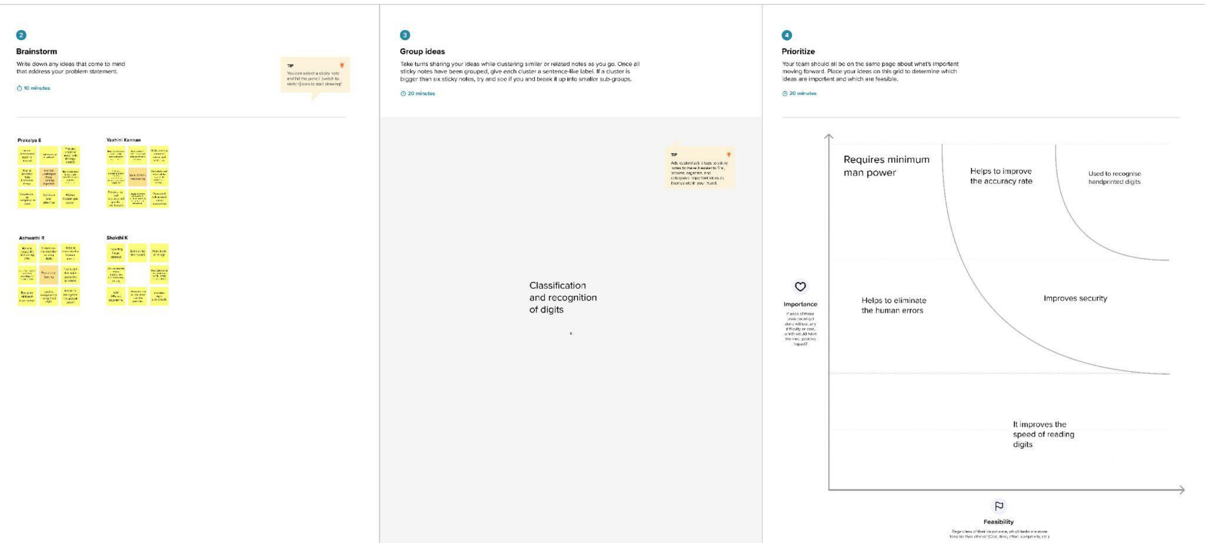
This model can be used to recognise the handwritten digits. The model is done using CNN, MNIST and is well trained with an accuracy of 99.15%.

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP


WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
The people
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Accurate results
Workers in bank and postal sector
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
To create a model with great accuracy
No human intervention needed to recognise the digits
Data preprocessing
Helpful in bank check processing, postal cards.
Building user friendly and attractive UI
Can be used in number plate recognition system
Everything being digitalized
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
Does training require more time
Can it be used for numbers of different languages
Existing solutions are available
Would it be efficient
What do they SAY?
What have we heard them say?
What can we imagine them saying?
Handwritten digits must be digitalized
Need a model to recognise digits
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Avoid the manual typing of digits
Write the digits using pen and paper
Produce error free result with great accuracy
Helpful in all sectors

3.2 IDEATION AND BRAINSTORMING



2 **Brainstorm**

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP: You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

3 **Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP: Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4 **Prioritize**

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance: If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility: Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

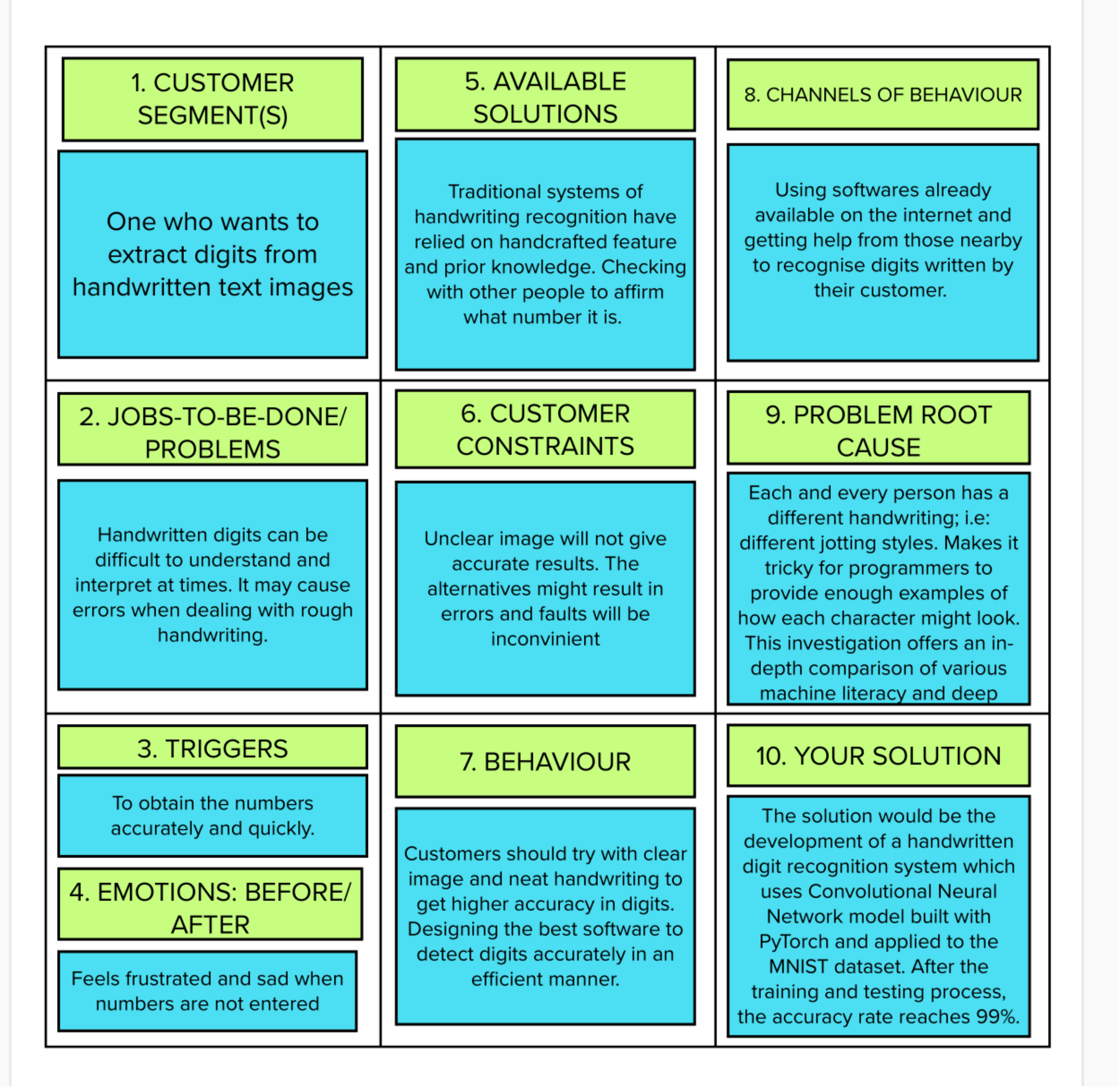
Requires minimum man power
Helps to improve the accuracy rate
Used to recognise handprinted digits
Helps to eliminate the human errors
Improves security
It improves the speed of reading digits

3.3 PROPOSED SOLUTION

S.NO	PARAMETERS	DESCRIPTION
1.	Problem Statement (Problem to be solved)	Digit recognition is essential in the modern world. It has the capacity to resolve problems that are getting harder and easier while facilitating human work. One instance is the recognition of handwritten digits. This is a technique that is used globally to identify zip codes or postal codes for mail sorting. A variety of methods can be used to recognise handwritten digits. Because handwritten numerals are not always accurate and can be produced in a variety of ways, the machine has a challenging task. Handwritten digit identification, which uses a picture of a digit to identify the digit represented in the image, offers a solution to this problem.
2.	Idea / Solution description	With 60,000 training photos of handwritten digits from 0 to 9 and 10,000 test images, the MNIST dataset is used to conduct handwritten digit recognition. The MNIST dataset therefore includes 10 distinct classifications. We're going to put into practise a Convolutional Neural Networks model trained application for handwritten digit recognition in this project. The user enters the handwritten digit into a GUI, which recognises it, and the answer is shown instantly.
3.	Novelty / Uniqueness	In the field of handwriting visual recognition, this project presents a practical method for addressing novelty. In addition to identifying any aesthetic differences that could exist inside or across texts, the ideal transcription agent would be able to discriminate between known and unknown characters in an image. The novelty is brought in by making use of tools and algorithms that generate multiple copies of the image with different types of altercations in width, height, skew, etc. This makes the model more accurate and reliable

4.	Social Impact / Customer Satisfaction	With handwriting recognition technology come a lot of advantages. In addition to reading postal addresses and bank check amounts, it is also helpful for reading forms. As a result of how simple it is to compare two texts and establish whether one is a copy, it is also employed in the detection of fraud. This suggested approach ought to be capable of detecting those digits as well, given that users in rural locations will speak their own regional language. Given that it is intended to address real life issues, it must be completely trustworthy and extremely dependable in all respects, and it must be used by people all over the world.
5.	Business Model (Revenue Model)	The major revenue generating sectors are banking, healthcare, retail, tourism, logistics, transportation, government, manufacturing, and other sectors. All procedures are now quicker and easier to access as a result of digitalization in commercial organisations. Data is becoming an essential component for success as businesses experience technological breakthroughs. When information is transformed into digital form, it can be processed by computers and other computing devices, making it simple to distribute, access, and store. Hence the market value of this technology is very high.
6.	Scalability of the Solution	Scaling of the model can be achieved by expanding the dataset to regional languages. This makes it very useful especially in rural areas where people are prone to writing in the local text. Another method is to use IBM Cloud AI to optimize, train and improve the efficiency of the working model. The high accuracy and reliability makes it more desirable to the market.

3.4 PROBLEM SOLUTION FIT



1. CUSTOMER SEGMENT(S) One who wants to extract digits from handwritten text images	5. AVAILABLE SOLUTIONS Traditional systems of handwriting recognition have relied on handcrafted feature and prior knowledge. Checking with other people to affirm what number it is.	8. CHANNELS OF BEHAVIOUR Using softwares already available on the internet and getting help from those nearby to recognise digits written by their customer.
2. JOBS-TO-BE-DONE/ PROBLEMS Handwritten digits can be difficult to understand and interpret at times. It may cause errors when dealing with rough handwriting.	6. CUSTOMER CONSTRAINTS Unclear image will not give accurate results. The alternatives might result in errors and faults will be inconvinient	9. PROBLEM ROOT CAUSE Each and every person has a different handwriting; i.e: different jotting styles. Makes it tricky for programmers to provide enough examples of how each character might look. This investigation offers an in-depth comparison of various machine literacy and deep
3. TRIGGERS To obtain the numbers accurately and quickly. 4. EMOTIONS: BEFORE/ AFTER Feels frustrated and sad when numbers are not entered	7. BEHAVIOUR Customers should try with clear image and neat handwriting to get higher accuracy in digits. Designing the best software to detect digits accurately in an efficient manner.	10. YOUR SOLUTION The solution would be the development of a handwritten digit recognition system which uses Convolutional Neural Network model built with PyTorch and applied to the MNIST dataset. After the training and testing process, the accuracy rate reaches 99%.

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Website	The code, graphics, and other components of a website are made available online through web hosting.
FR-2	Digit Classifier Model	Packages - tensorflow, keras
FR-3	MNIST Dataset	MNIST is a handwritten digits dataset which can be used for training various image processing systems. It has 60,000 training and 10,000 testing examples.

4.2 NON-FUNCTIONAL REQUIREMENTS

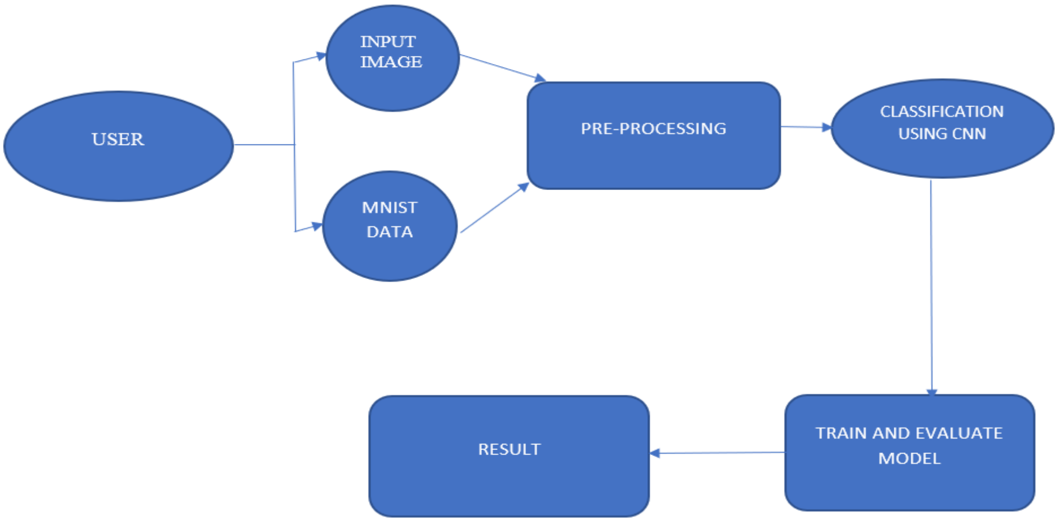
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Users should be able to understand and use the system easily. In addition, it ought to be simple for users of all skill levels to navigate.
NFR-2	Reliability	The web application must give an accurate result as much as possible. It also indicates the probability that the developed model will perform its function without any failures.

NFR-3	Performance	The delay in providing the information when hundreds of requests are given should be minimum i.e, the model should be fast enough.
NFR-4	Availability	Access to information is restricted to each user.
NFR-5	Scalability	Ten thousand concurrent site visitors should be manageable for the system.

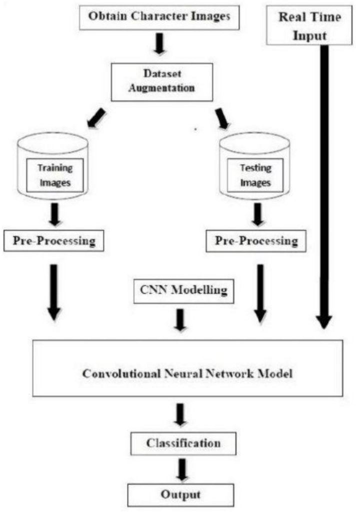
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Technical Architecture:



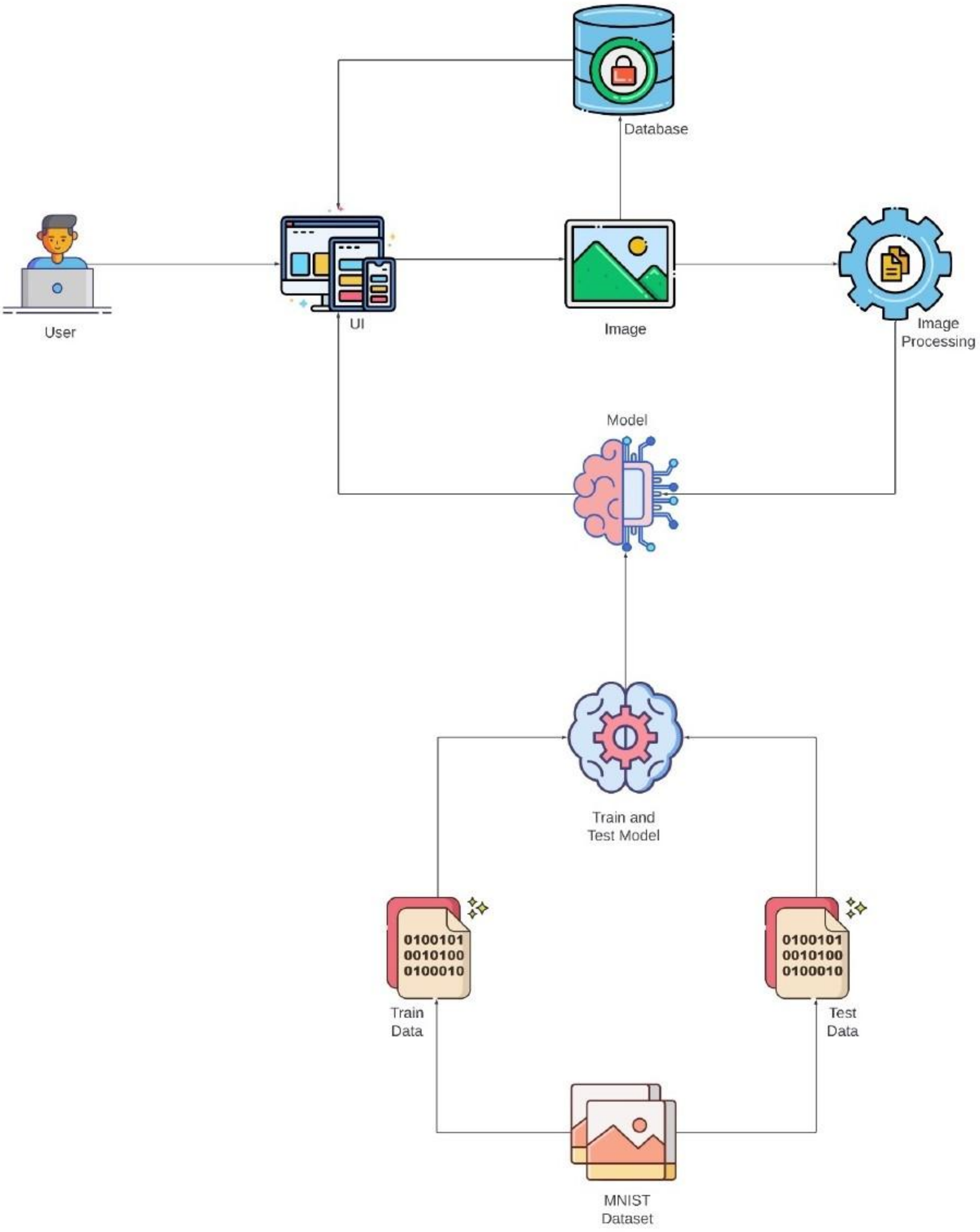

Database
User
UI
Image
Image Processing
Model
Train and Test Model
0100101
0010100
0100010
Train Data
0100101
0010100
0100010
Test Data
MNIST Dataset

Table-1 : Components, technologies and description:

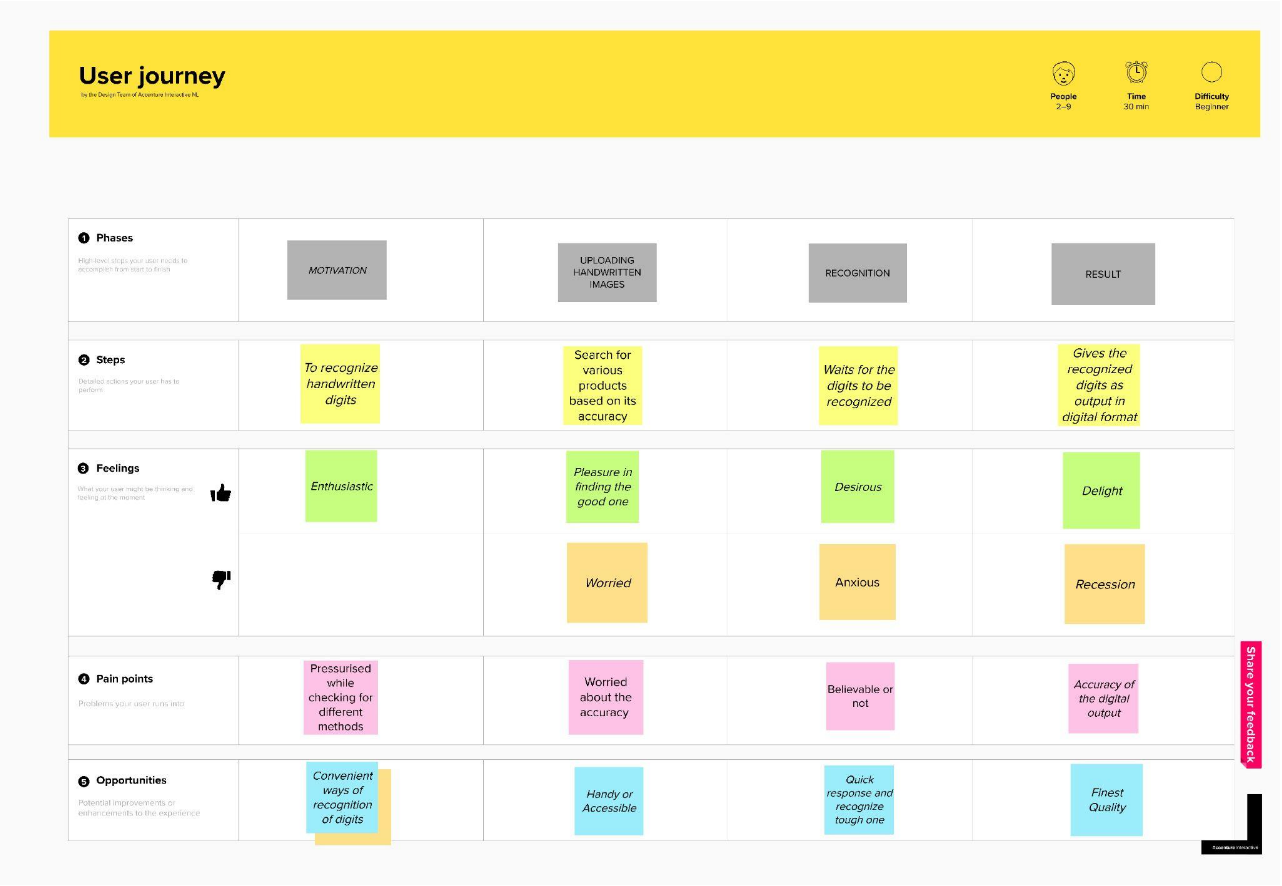
S.No	Component	Description	Technology
1.	User Interface	How the user interacts with applications e.g. Web UI	HTML, CSS, Flask
2.	Application Logic-1	Logic for the model that is being built	Python
3.	Application Logic-2	IBM model is trained using the Python code developed	IBM Watson service
4.	Deep Learning Model	Purpose of this model is to recognise the digits from the image uploaded by the user	CNN model
5.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration	IBM cloud

Table-2: Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Deep learning frameworks can help you upload data and train a deep learning model that would lead to accurate and intuitive predictive analysis.	Tensorflow, Visual code or pycharm
2.	Scalable Architecture	The system should be able to handle 10000 users accessing the site at the same time.	Python with IBM cloud model

3.	Availability	Information is restricted to each user's limited access.	IBM cloud
4.	Performance	Should reduce the delay in information when hundreds of requests are given. The system should be fast.	Python and flask

5.3 USER STORIES



User journey

by the Design Team of Accenture Interactive NL

People 2–9 | Time 30 min | Difficulty Beginner

❶ **Phases** High-level steps your user needs to accomplish from start to finish	MOTIVATION	UPLOADING HANDWRITTEN IMAGES	RECOGNITION	RESULT
❷ **Steps** Detailed actions your user has to perform	To recognize handwritten digits	Search for various products based on its accuracy	Waits for the digits to be recognized	Gives the recognized digits as output in digital format
❸ **Feelings** What your user might be thinking and feeling at the moment 👍	Enthusiastic	Pleasure in finding the good one	Desirous	Delight
👎		Worried	Anxious	Recession
❹ **Pain points** Problems your user runs into	Pressurised while checking for different methods	Worried about the accuracy	Believable or not	Accuracy of the digital output
❺ **Opportunities** Potential improvements or enhancements to the experience	Convenient ways of recognition of digits	Handy or Accessible	Quick response and recognize tough one	Finest Quality

Share your feedback

Accenture Interactive

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection & pre processing	USN-1	As a user, I can upload any kind of image that has gone through the pre-processing step.	10	High	Prakalya E, Aswathi R, Yazhini Kannan
Sprint-1		USN-2	As a user, I can upload the image in any resolution.	10	High	Shakthi K, Aswathi R, Yazhini Kannan
Sprint-2	Building the Machine learning model	USN-3	As a user, I will receive a web application with a machine learning model that is highly accurate at recognizing digits written by hand.	5	Medium	Yazhini Kannan, Shakthi K
Sprint-2		USN-4	As a user, I can show the image of the handwritten number for them to recognize.	5	Medium	Prakalya E, Aswathi R
Sprint-2		USN-5	As a user, I am able to obtain the	10	High	Prakalya E, Aswathi R, Shakthi K

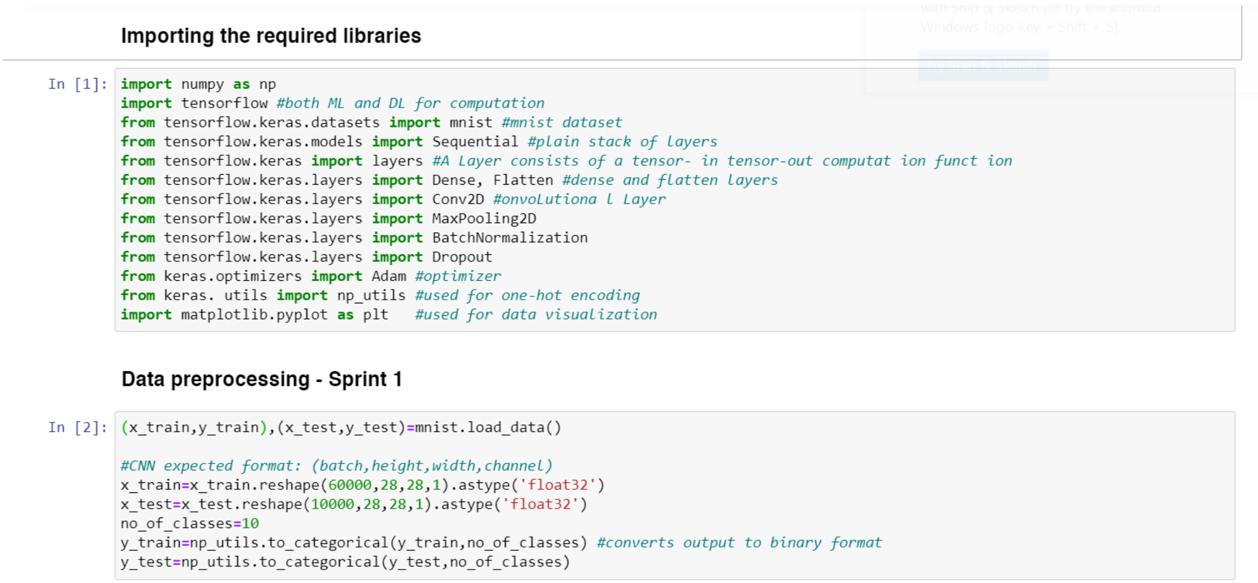
			ideal recognized digit.			
Sprint-3	Building User Interface for Web Application	USN-6	As a user, I will use an upload button to add the image of the handwritten digits to the web page.	5	Medium	Aswathi R, Shakthi K
Sprint-3		USN-7	As a user, I can be aware of the web page's fundamental usage in detail.	3	Low	Prakalya E
Sprint-3		USN-8	As a user, in the web page, I may see the anticipated or recognised digits.	5	Medium	Yazhini Kannan, Prakalya E
Sprint-4	Train and deployment of model in IBM Cloud	USN-9	I can utilize the product, and access the web application from anywhere.	10	High	Shakthi K, Aswathi R, Yazhini Kannan

6.2 SPRINT DELIVERY SCHEDULE:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

CODING AND SOLUTIONING

DEVELOPMENT OF MODEL

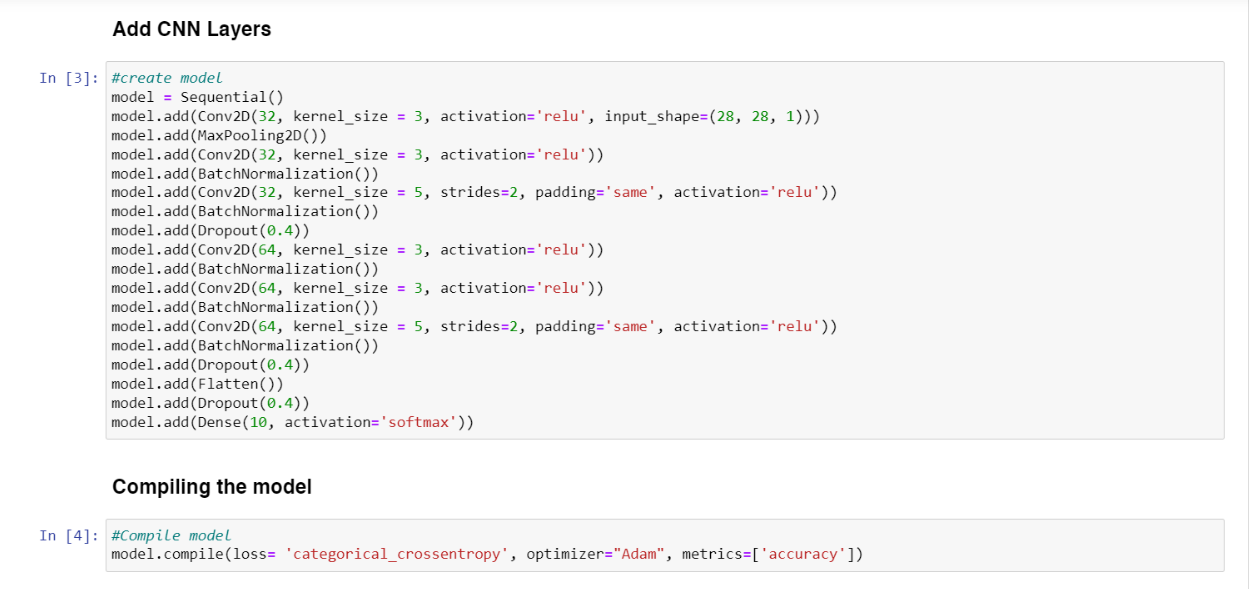

with Snip & Sketch (or try the shortcut
Windows logo key + Shift + S).
Try Snip & Sketch

Importing the required libraries

```
In [1]: import numpy as np
        import tensorflow #both ML and DL for computation
        from tensorflow.keras.datasets import mnist #mnist dataset
        from tensorflow.keras.models import Sequential #plain stack of layers
        from tensorflow.keras import layers #A Layer consists of a tensor- in tensor-out computat ion funct ion
        from tensorflow.keras.layers import Dense, Flatten #dense and flatten layers
        from tensorflow.keras.layers import Conv2D #onvoLutiona l Layer
        from tensorflow.keras.layers import MaxPooling2D
        from tensorflow.keras.layers import BatchNormalization
        from tensorflow.keras.layers import Dropout
        from keras.optimizers import Adam #optimizer
        from keras. utils import np_utils #used for one-hot encoding
        import matplotlib.pyplot as plt   #used for data visualization
```

Data preprocessing - Sprint 1

```
In [2]: (x_train,y_train),(x_test,y_test)=mnist.load_data()

        #CNN expected format: (batch,height,width,channel)
        x_train=x_train.reshape(60000,28,28,1).astype('float32')
        x_test=x_test.reshape(10000,28,28,1).astype('float32')
        no_of_classes=10
        y_train=np_utils.to_categorical(y_train,no_of_classes) #converts output to binary format
        y_test=np_utils.to_categorical(y_test,no_of_classes)
```

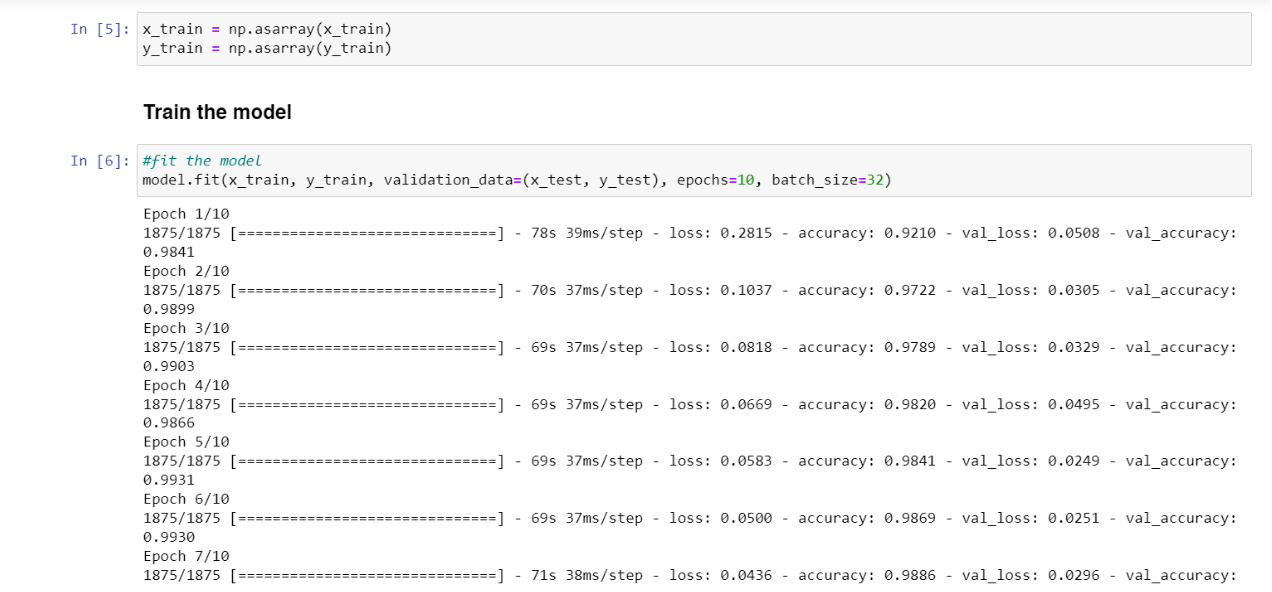


Add CNN Layers

```
In [3]: #create model
        model = Sequential()
        model.add(Conv2D(32, kernel_size = 3, activation='relu', input_shape=(28, 28, 1)))
        model.add(MaxPooling2D())
        model.add(Conv2D(32, kernel_size = 3, activation='relu'))
        model.add(BatchNormalization())
        model.add(Conv2D(32, kernel_size = 5, strides=2, padding='same', activation='relu'))
        model.add(BatchNormalization())
        model.add(Dropout(0.4))
        model.add(Conv2D(64, kernel_size = 3, activation='relu'))
        model.add(BatchNormalization())
        model.add(Conv2D(64, kernel_size = 3, activation='relu'))
        model.add(BatchNormalization())
        model.add(Conv2D(64, kernel_size = 5, strides=2, padding='same', activation='relu'))
        model.add(BatchNormalization())
        model.add(Dropout(0.4))
        model.add(Flatten())
        model.add(Dropout(0.4))
        model.add(Dense(10, activation='softmax'))
```

Compiling the model

```
In [4]: #Compile model
        model.compile(loss= 'categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
```

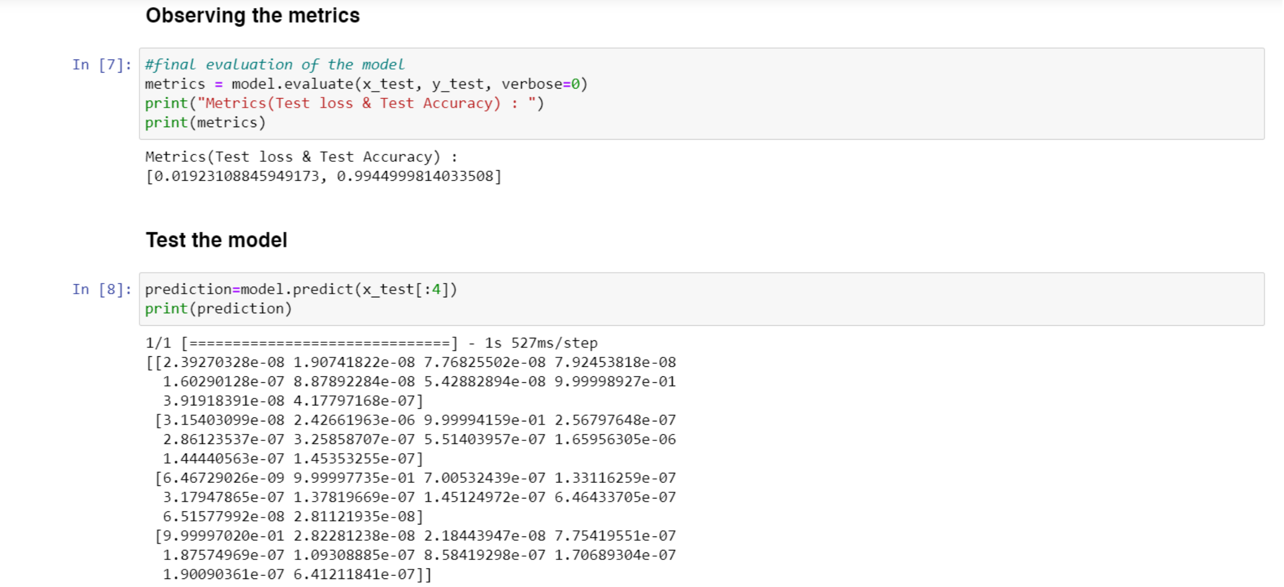


```
In [5]: x_train = np.asarray(x_train)
        y_train = np.asarray(y_train)
```

Train the model

```
In [6]: #fit the model
        model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=32)
```

```
Epoch 1/10
1875/1875 [==============================] - 78s 39ms/step - loss: 0.2815 - accuracy: 0.9210 - val_loss: 0.0508 - val_accuracy:
0.9841
Epoch 2/10
1875/1875 [==============================] - 70s 37ms/step - loss: 0.1037 - accuracy: 0.9722 - val_loss: 0.0305 - val_accuracy:
0.9899
Epoch 3/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0818 - accuracy: 0.9789 - val_loss: 0.0329 - val_accuracy:
0.9903
Epoch 4/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0669 - accuracy: 0.9820 - val_loss: 0.0495 - val_accuracy:
0.9866
Epoch 5/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0583 - accuracy: 0.9841 - val_loss: 0.0249 - val_accuracy:
0.9931
Epoch 6/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0500 - accuracy: 0.9869 - val_loss: 0.0251 - val_accuracy:
0.9930
Epoch 7/10
1875/1875 [==============================] - 71s 38ms/step - loss: 0.0436 - accuracy: 0.9886 - val_loss: 0.0296 - val_accuracy:
```



Observing the metrics

```
In [7]: #final evaluation of the model
        metrics = model.evaluate(x_test, y_test, verbose=0)
        print("Metrics(Test loss & Test Accuracy) : ")
        print(metrics)
```

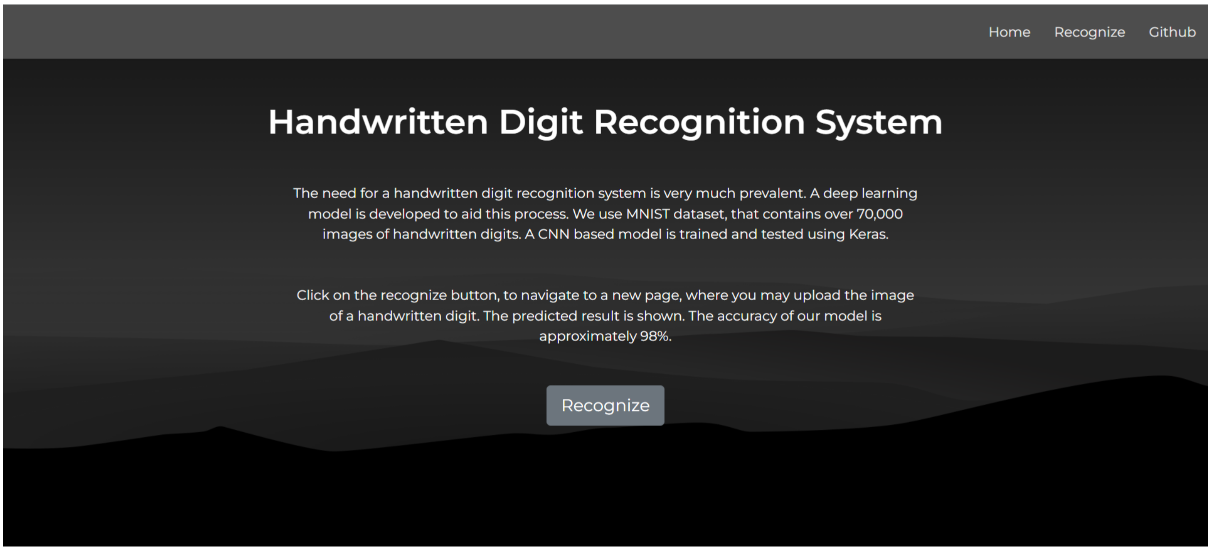
```
Metrics(Test loss & Test Accuracy) :
[0.019231088459491730, 0.9944999814033508]
```

Test the model

```
In [8]: prediction=model.predict(x_test[:4])
        print(prediction)
```

```
1/1 [==============================] - 1s 527ms/step
[[2.39270328e-08 1.90741822e-08 7.76825502e-08 7.92453818e-08
  1.60290128e-07 8.87892284e-08 5.42882894e-08 9.99998927e-01
  3.91918391e-08 4.17797168e-07]
 [3.15403099e-08 2.42661963e-06 9.99994159e-01 2.56797648e-07
  2.86123537e-07 3.25858707e-07 5.51403957e-07 1.65956305e-06
  1.44440563e-07 1.45353255e-07]
 [6.46729026e-09 9.99997735e-01 7.00532439e-07 1.33116259e-07
  3.17947865e-07 1.37819669e-07 1.45124972e-07 6.46433705e-07
  6.51577992e-08 2.81121935e-08]
 [9.99997020e-01 2.82281238e-08 2.18443947e-08 7.75419551e-07
  1.87574969e-07 1.09308885e-07 8.58419298e-07 1.70689304e-07
  1.90090361e-07 6.41211841e-07]]
```

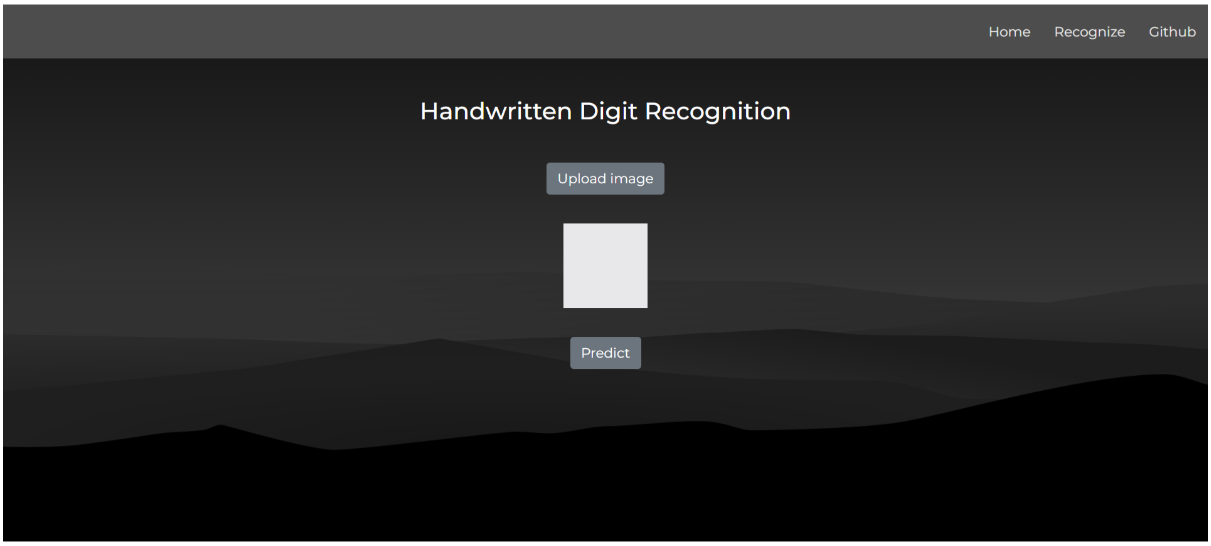
OUTPUT



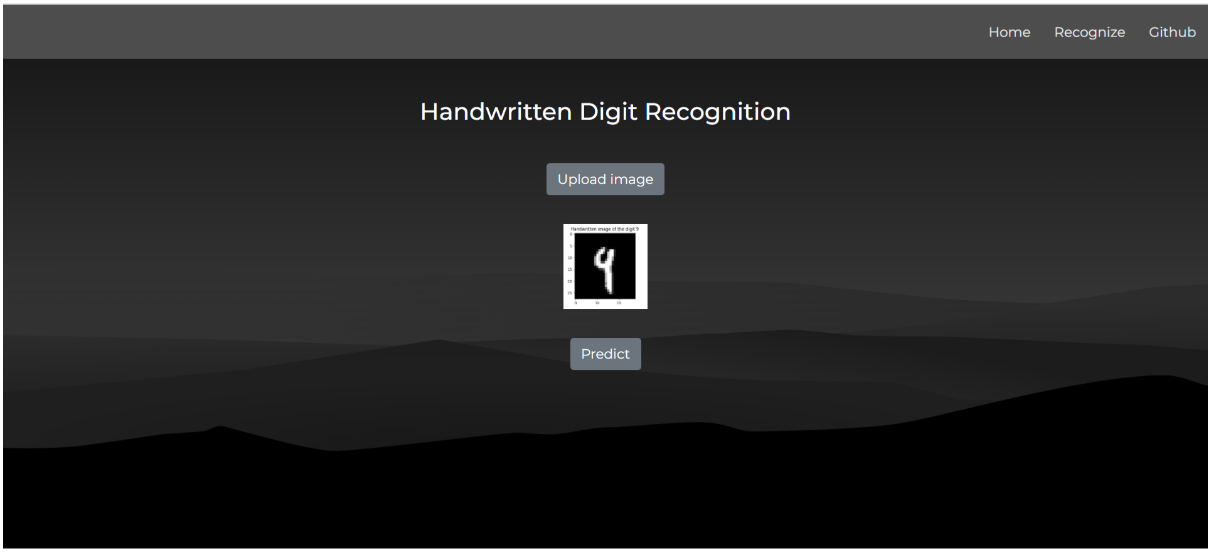
Handwritten Digit Recognition System

The need for a handwritten digit recognition system is very much prevalent. A deep learning model is developed to aid this process. We use MNIST dataset, that contains over 70,000 images of handwritten digits. A CNN based model is trained and tested using Keras.

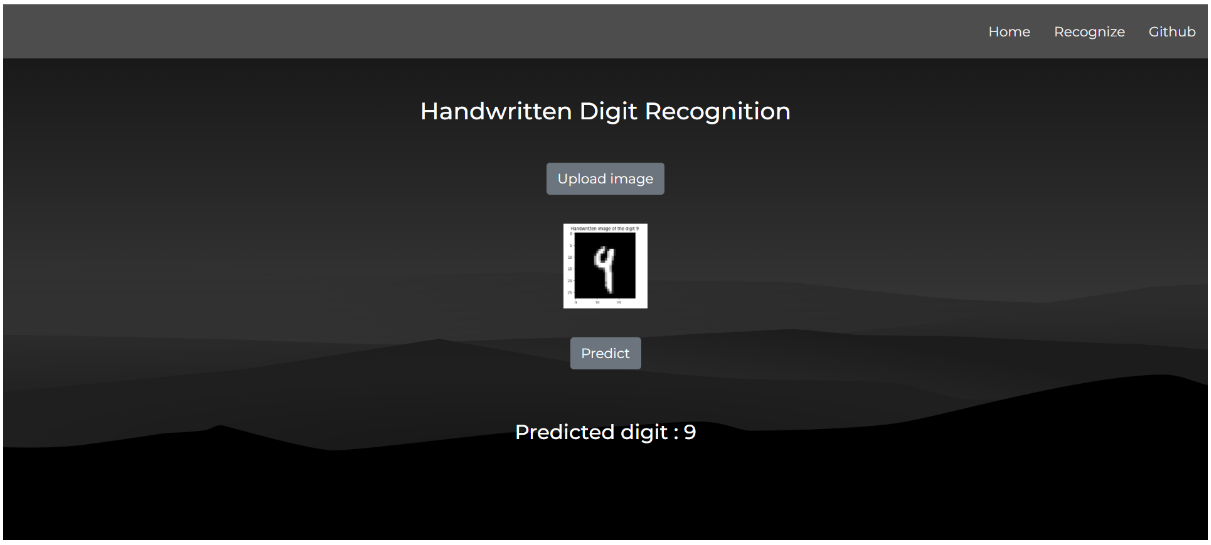
Click on the recognize button, to navigate to a new page, where you may upload the image of a handwritten digit. The predicted result is shown. The accuracy of our model is approximately 98%.



Handwritten Digit Recognition



Handwritten Digit Recognition

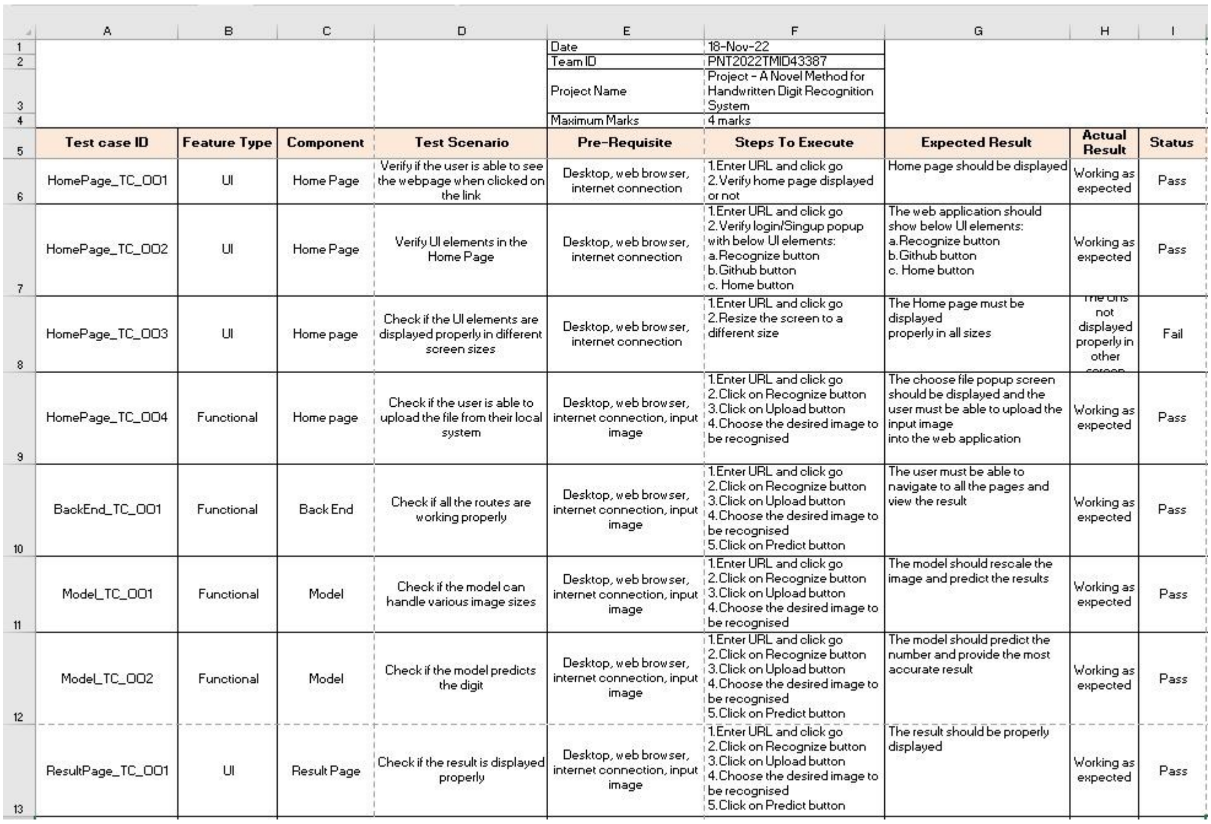


Handwritten Digit Recognition

Predicted digit : 9

TESTING

TEST CASE REPORT



Date	18-Nov-22
Team ID	PNT2022TMID43387
Project Name	Project - A Novel Method for Handwritten Digit Recognition System
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Expected Result	Actual Result	Status
HomePage_TC_OO1	UI	Home Page	Verify if the user is able to see the webpage when clicked on the link	Desktop, web browser, internet connection	1.Enter URL and click go 2.Verify home page displayed or not	Home page should be displayed	Working as expected	Pass
HomePage_TC_OO2	UI	Home Page	Verify UI elements in the Home Page	Desktop, web browser, internet connection	1.Enter URL and click go 2.Verify login/Singup popup with below UI elements: a.Recognize button b.Github button c. Home button	The web application should show below UI elements: a.Recognize button b.Github button c. Home button	Working as expected	Pass
HomePage_TC_OO3	UI	Home page	Check if the UI elements are displayed properly in different screen sizes	Desktop, web browser, internet connection	1.Enter URL and click go 2.Resize the screen to a different size	The Home page must be displayed properly in all sizes	The UI's not displayed properly in other screen	Fail
HomePage_TC_OO4	Functional	Home page	Check if the user is able to upload the file from their local system	Desktop, web browser, internet connection, input image	1.Enter URL and click go 2.Click on Recognize button 3.Click on Upload button 4.Choose the desired image to be recognised	The choose file popup screen should be displayed and the user must be able to upload the input image into the web application	Working as expected	Pass
BackEnd_TC_OO1	Functional	Back End	Check if all the routes are working properly	Desktop, web browser, internet connection, input image	1.Enter URL and click go 2.Click on Recognize button 3.Click on Upload button 4.Choose the desired image to be recognised 5.Click on Predict button	The user must be able to navigate to all the pages and view the result	Working as expected	Pass
Model_TC_OO1	Functional	Model	Check if the model can handle various image sizes	Desktop, web browser, internet connection, input image	1.Enter URL and click go 2.Click on Recognize button 3.Click on Upload button 4.Choose the desired image to be recognised	The model should rescale the image and predict the results	Working as expected	Pass
Model_TC_OO2	Functional	Model	Check if the model predicts the digit	Desktop, web browser, internet connection, input image	1.Enter URL and click go 2.Click on Recognize button 3.Click on Upload button 4.Choose the desired image to be recognised 5.Click on Predict button	The model should predict the number and provide the most accurate result	Working as expected	Pass
ResultPage_TC_OO1	UI	Result Page	Check if the result is displayed properly	Desktop, web browser, internet connection, input image	1.Enter URL and click go 2.Click on Recognize button 3.Click on Upload button 4.Choose the desired image to be recognised 5.Click on Predict button	The result should be properly displayed	Working as expected	Pass

USER ACCEPTANCE TESTING

1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of **A Novel Method for Handwritten Digit Recognition System** project at the time of the release to User Acceptance Testing (UAT).

2.Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	1	2	3	8
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	6	2	1	8	17
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	1	2	2	5
Totals	11	7	10	15	43

3.Test Case Analysis

This report shows the number of test cases that have passed, failed,and untested

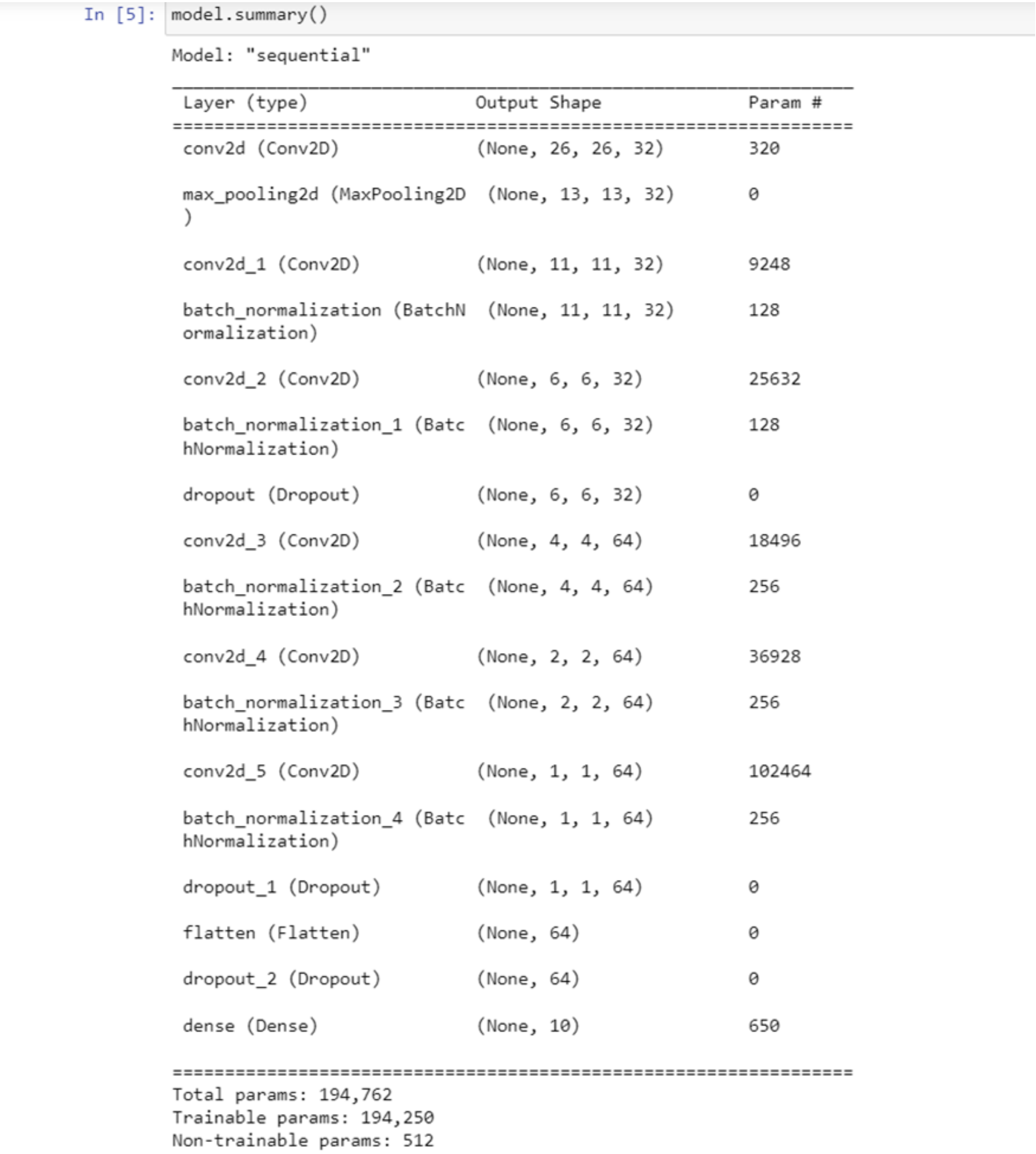
Section	Total Cases	Not Tested	Fail	Pass
Home Page	4	0	1	3
Back End	1	0	0	1
Model	2	0	0	2

Result page	1	0	0	1

RESULTS

Model Performance Testing:

1. MODEL SUMMARY



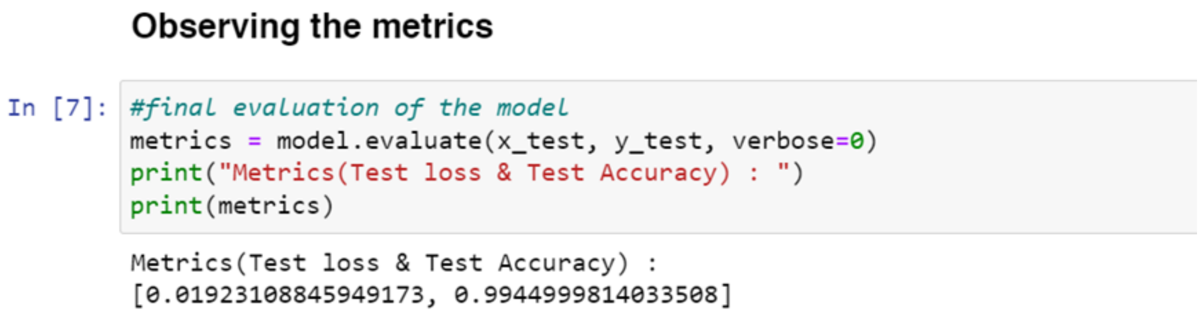
```
In [5]: model.summary()

        Model: "sequential"
        _________________________________________________________________
         Layer (type)                Output Shape              Param #
        =================================================================
         conv2d (Conv2D)             (None, 26, 26, 32)        320

         max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
         )

         conv2d_1 (Conv2D)           (None, 11, 11, 32)        9248

         batch_normalization (BatchN  (None, 11, 11, 32)       128
         ormalization)

         conv2d_2 (Conv2D)           (None, 6, 6, 32)          25632

         batch_normalization_1 (Batc  (None, 6, 6, 32)         128
         hNormalization)

         dropout (Dropout)           (None, 6, 6, 32)          0

         conv2d_3 (Conv2D)           (None, 4, 4, 64)          18496

         batch_normalization_2 (Batc  (None, 4, 4, 64)         256
         hNormalization)

         conv2d_4 (Conv2D)           (None, 2, 2, 64)          36928

         batch_normalization_3 (Batc  (None, 2, 2, 64)         256
         hNormalization)

         conv2d_5 (Conv2D)           (None, 1, 1, 64)          102464

         batch_normalization_4 (Batc  (None, 1, 1, 64)         256
         hNormalization)

         dropout_1 (Dropout)         (None, 1, 1, 64)          0

         flatten (Flatten)           (None, 64)                0

         dropout_2 (Dropout)         (None, 64)                0

         dense (Dense)               (None, 10)                650

        =================================================================
        Total params: 194,762
        Trainable params: 194,250
        Non-trainable params: 512
```

2. ACCURACY

Values:

Test loss: 0.01923

Test Accuracy: 0.99449



Observing the metrics

In [7]:
```
#final evaluation of the model
metrics = model.evaluate(x_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy) : ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy) :
[0.01923108845949173, 0.9944999814033508]
```

ADVANTAGES AND DISADVANTAGES

Any system of its own has both advantages and disadvantages. The advantages and disadvantages of this model are as follows.

ADVANTAGES

Using this system has many advantages. Some of them are:

- Ability to handle a lot of data
- Less manual labor
- Better accuracy than an average person
- Can handle arbitrary scalings, translations and a limited degree of image rotation.

DISADVANTAGES

The disadvantages of this model are:

- All data must be digital
- For quicker predictions, a server with high performance is required.
- Prone to occasional errors
- Unable to handle complex data
- It is not done in real time as a person writes and therefore not appropriate for immediate text input
- Requires much more computation than more standard OCR techniques

CONCLUSION

This project showcased a web application that uses machine learning to recognise handwritten numerals. Flask, HTML, CSS, and a few other technologies were used to construct this project. The model forecasts the handwritten digit using a CNN network. The suggested project is scalable and reliable.

There are many real world applications for this project, which includes reading bank cheques for amounts, processing license plate recognition data, and manually inputting data on forms like tax returns. There is a great deal of room for development that can be included into later iterations.

Accuracy can alter as it depends on the splitting of training and testing data, and this can further be improved if the number of training and testing data is provided. There is always a chance to improve accuracy if the size of data increases. Every classifier has its own accuracy and time consumption. We can also include the fact that if the power of CPU changes to GPU, the classifier can perform with better accuracy and less time and better results can be observed.

The performance of the classifier can be measured in terms of ability to identify a condition properly (sensitivity), the proportion of true results (accuracy), number of positive results from the procedure of classification as false positives (positive predictions) and ability to exclude condition correctly (specificity).

FUTURE SCOPE

This project has plenty of space for improvement. The following are some of the ways this project could be improved:

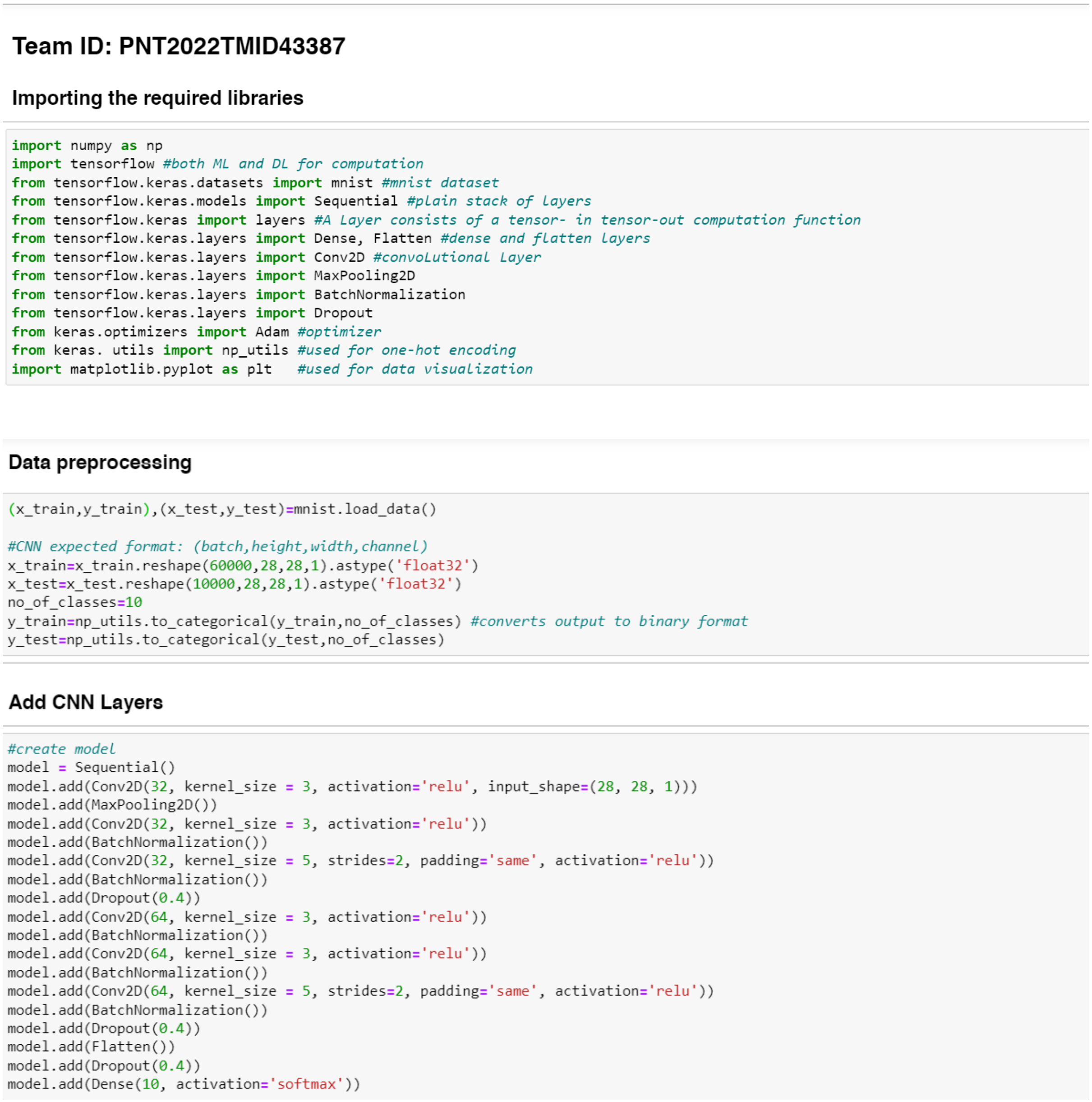
1. Include functionality for preserving the outcomes of repeated image detection from digits.
2. Add backing to recognize various digits.
3. Enhance the model to recognize digits in a variety of images.
4. Add support for various languages to assist users worldwide.

This project's potential is limitless, and it can constantly be enhanced to become better. Many sectors will benefit from implementing this approach, as will many workers, as it will cut workloads and enhance overall job efficiency.

The future of handwriting recognition is difficult to predict, but one scenario, not a very optimistic one, suggests that handwriting skills are starting to degenerate with the increased keyboard use. This trend will probably happen quite slowly though, and handwriting will probably not completely lose its significance, at least not during the next few generations.

APPENDIX

SOURCE CODE - MODEL



Team ID: PNT2022TMID43387

Importing the required libraries

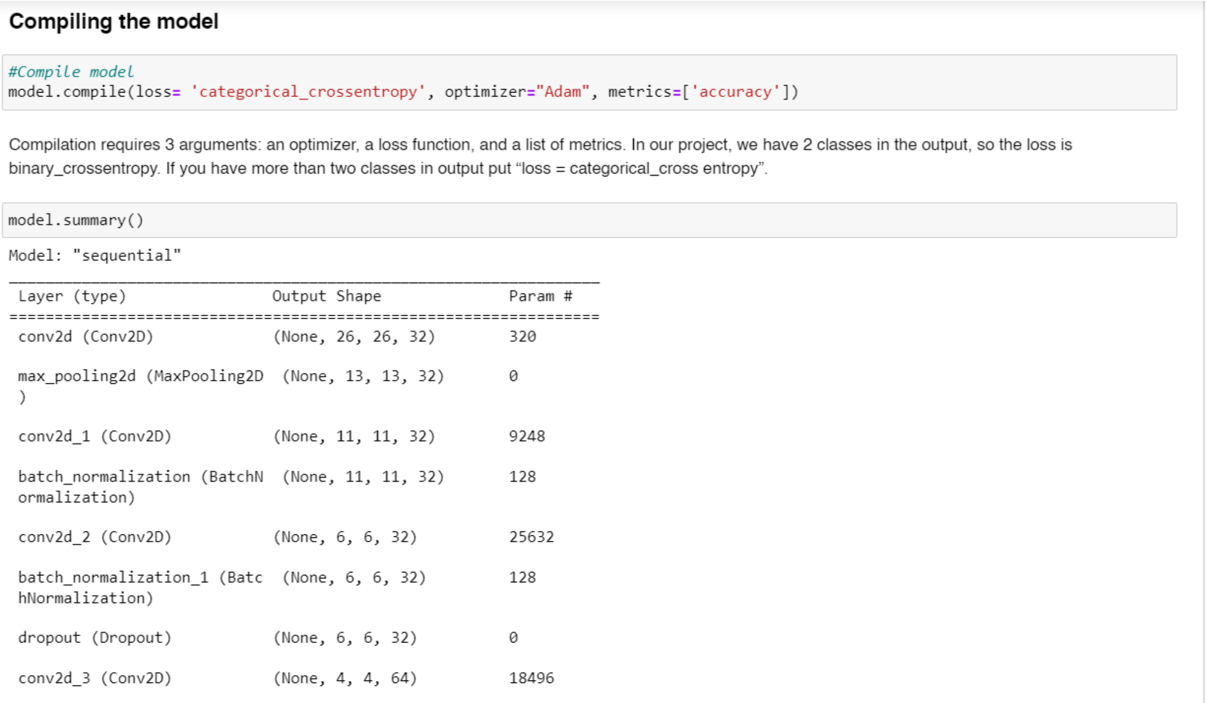
```
import numpy as np
import tensorflow #both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataset
from tensorflow.keras.models import Sequential #plain stack of layers
from tensorflow.keras import layers #A layer consists of a tensor- in tensor-out computation function
from tensorflow.keras.layers import Dense, Flatten #dense and flatten layers
from tensorflow.keras.layers import Conv2D #convolutional layer
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dropout
from keras.optimizers import Adam #optimizer
from keras. utils import np_utils #used for one-hot encoding
import matplotlib.pyplot as plt   #used for data visualization
```

Data preprocessing

```
(x_train,y_train),(x_test,y_test)=mnist.load_data()

#CNN expected format: (batch,height,width,channel)
x_train=x_train.reshape(60000,28,28,1).astype('float32')
x_test=x_test.reshape(10000,28,28,1).astype('float32')
no_of_classes=10
y_train=np_utils.to_categorical(y_train,no_of_classes) #converts output to binary format
y_test=np_utils.to_categorical(y_test,no_of_classes)
```

Add CNN Layers

```
#create model
model = Sequential()
model.add(Conv2D(32, kernel_size = 3, activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D())
model.add(Conv2D(32, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size = 5, strides=2, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Conv2D(64, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = 5, strides=2, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax'))
```



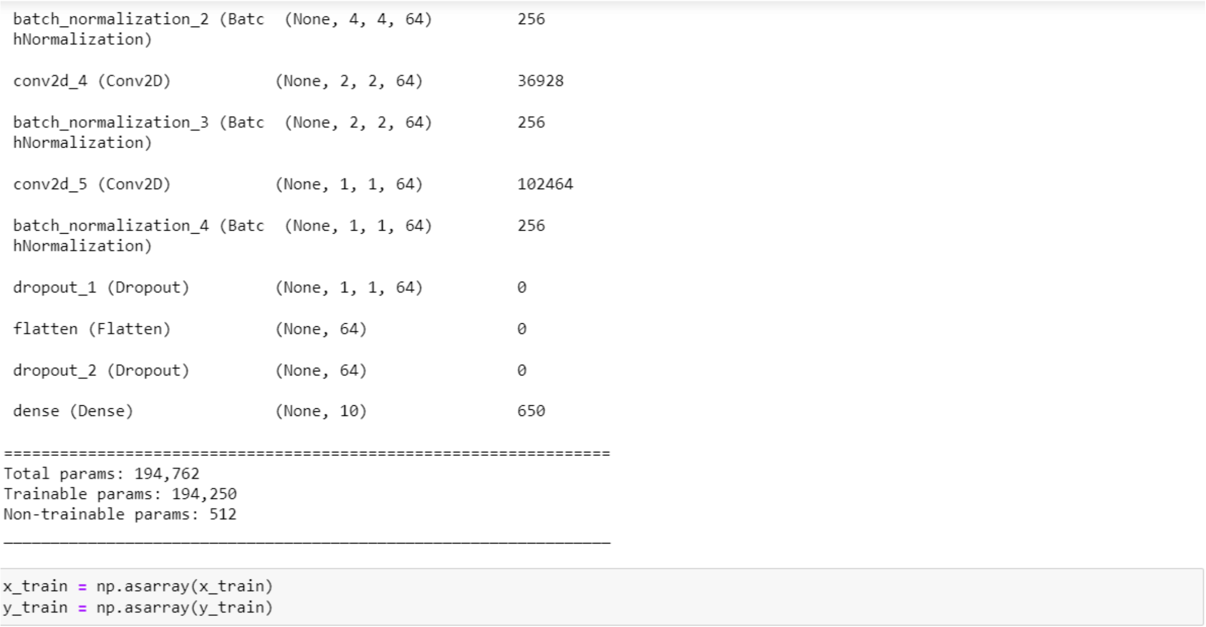
Compiling the model

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

Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics. In our project, we have 2 classes in the output, so the loss is binary_crossentropy. If you have more than two classes in output put “loss = categorical_cross entropy”.

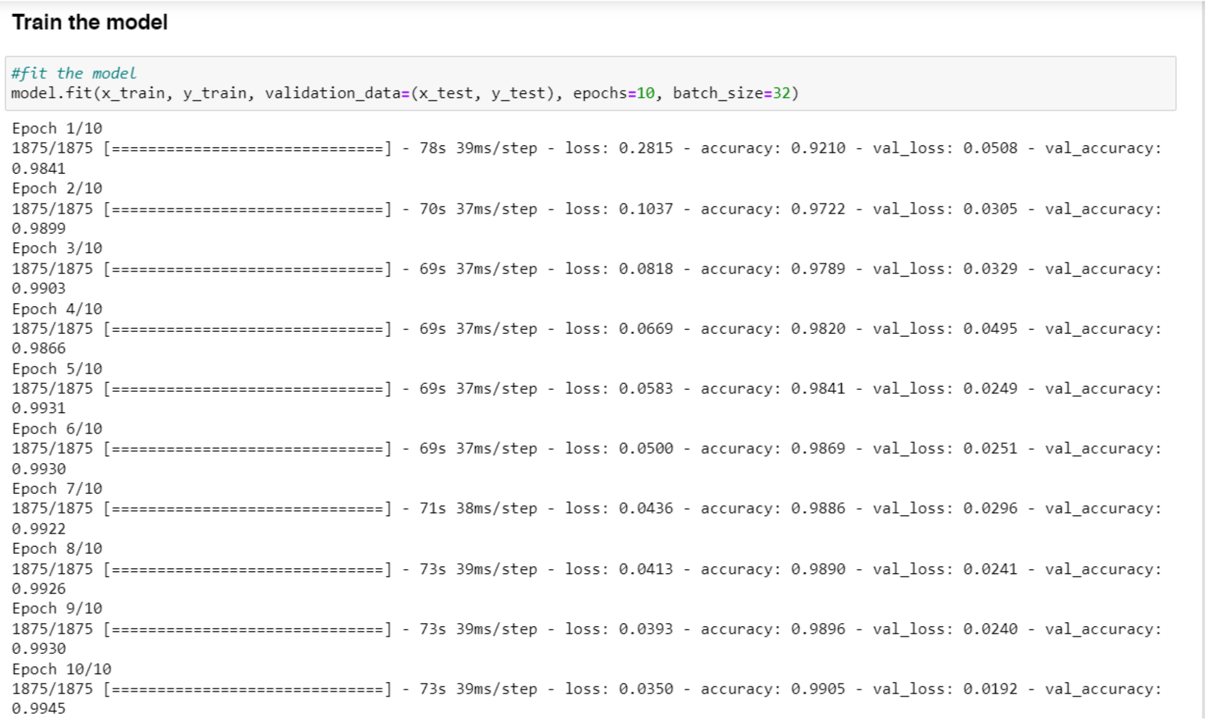
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 11, 11, 32)        9248      
                                                                 
 batch_normalization (BatchN  (None, 11, 11, 32)       128       
 ormalization)                                                   
                                                                 
 conv2d_2 (Conv2D)           (None, 6, 6, 32)          25632     
                                                                 
 batch_normalization_1 (Batc  (None, 6, 6, 32)         128       
 hNormalization)                                                 
                                                                 
 dropout (Dropout)           (None, 6, 6, 32)          0         
                                                                 
 conv2d_3 (Conv2D)           (None, 4, 4, 64)          18496     
```



```
 batch_normalization_2 (Batc  (None, 4, 4, 64)         256       
 hNormalization)                                                 
                                                                 
 conv2d_4 (Conv2D)           (None, 2, 2, 64)          36928     
                                                                 
 batch_normalization_3 (Batc  (None, 2, 2, 64)         256       
 hNormalization)                                                 
                                                                 
 conv2d_5 (Conv2D)           (None, 1, 1, 64)          102464    
                                                                 
 batch_normalization_4 (Batc  (None, 1, 1, 64)         256       
 hNormalization)                                                 
                                                                 
 dropout_1 (Dropout)         (None, 1, 1, 64)          0         
                                                                 
 flatten (Flatten)           (None, 64)                0         
                                                                 
 dropout_2 (Dropout)         (None, 64)                0         
                                                                 
 dense (Dense)               (None, 10)                650       
                                                                 
=================================================================
Total params: 194,762
Trainable params: 194,250
Non-trainable params: 512
_________________________________________________________________
```

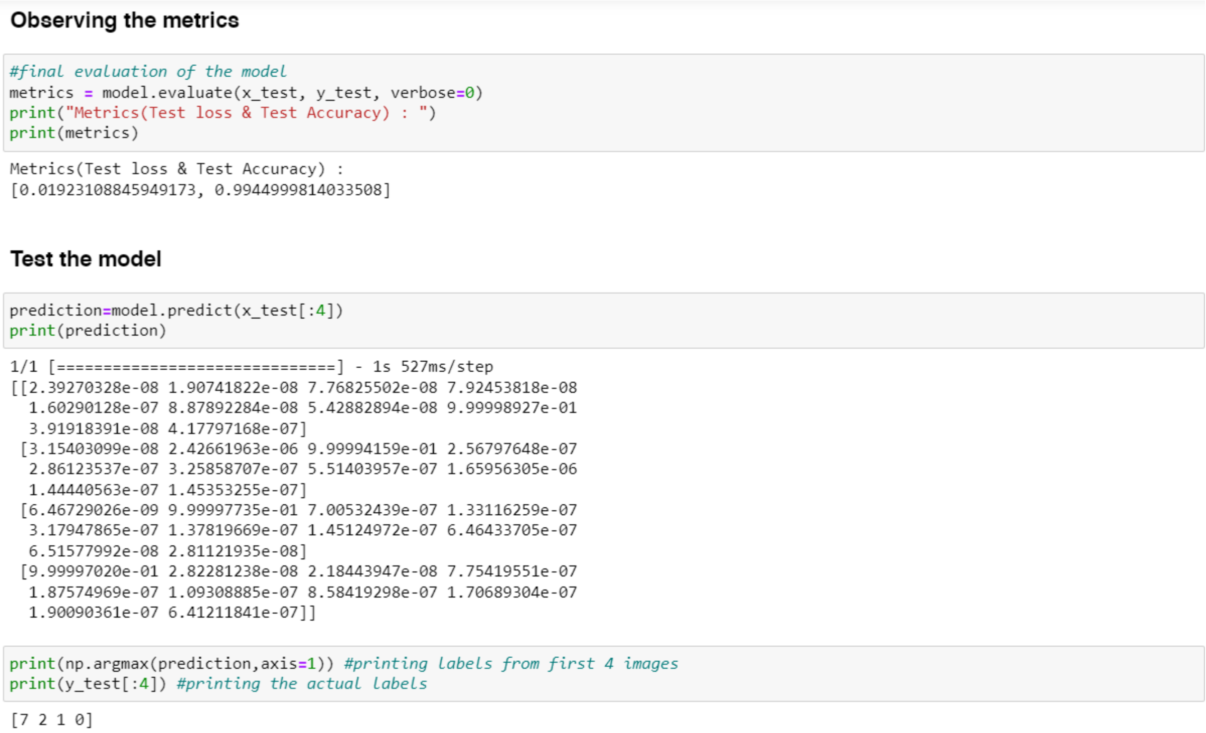
```
x_train = np.asarray(x_train)
y_train = np.asarray(y_train)
```



Train the model

```
#fit the model
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=32)
```

```
Epoch 1/10
1875/1875 [==============================] - 78s 39ms/step - loss: 0.2815 - accuracy: 0.9210 - val_loss: 0.0508 - val_accuracy:
0.9841
Epoch 2/10
1875/1875 [==============================] - 70s 37ms/step - loss: 0.1037 - accuracy: 0.9722 - val_loss: 0.0305 - val_accuracy:
0.9899
Epoch 3/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0818 - accuracy: 0.9789 - val_loss: 0.0329 - val_accuracy:
0.9903
Epoch 4/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0669 - accuracy: 0.9820 - val_loss: 0.0495 - val_accuracy:
0.9866
Epoch 5/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0583 - accuracy: 0.9841 - val_loss: 0.0249 - val_accuracy:
0.9931
Epoch 6/10
1875/1875 [==============================] - 69s 37ms/step - loss: 0.0500 - accuracy: 0.9869 - val_loss: 0.0251 - val_accuracy:
0.9930
Epoch 7/10
1875/1875 [==============================] - 71s 38ms/step - loss: 0.0436 - accuracy: 0.9886 - val_loss: 0.0296 - val_accuracy:
0.9922
Epoch 8/10
1875/1875 [==============================] - 73s 39ms/step - loss: 0.0413 - accuracy: 0.9890 - val_loss: 0.0241 - val_accuracy:
0.9926
Epoch 9/10
1875/1875 [==============================] - 73s 39ms/step - loss: 0.0393 - accuracy: 0.9896 - val_loss: 0.0240 - val_accuracy:
0.9930
Epoch 10/10
1875/1875 [==============================] - 73s 39ms/step - loss: 0.0350 - accuracy: 0.9905 - val_loss: 0.0192 - val_accuracy:
0.9945
```



Observing the metrics

```
#final evaluation of the model
metrics = model.evaluate(x_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy) : ")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy) :
[0.01923108845949173, 0.9944999814033508]
```

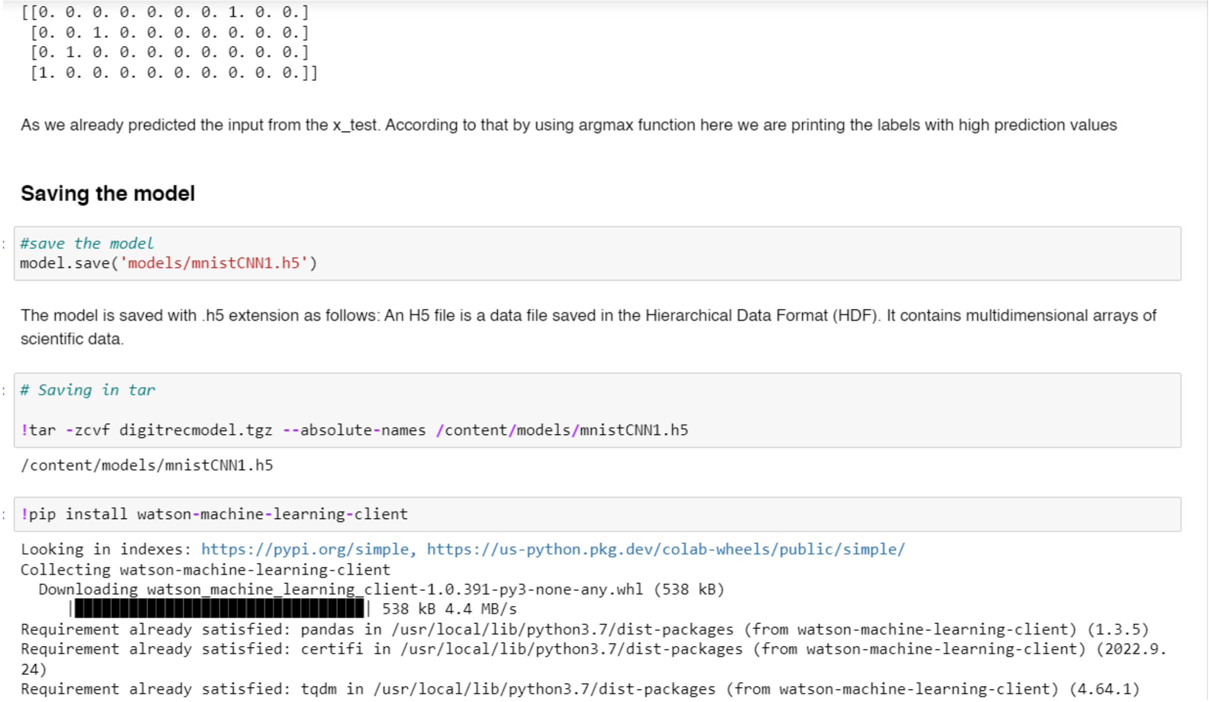
Test the model

```
prediction=model.predict(x_test[:4])
print(prediction)
```

```
1/1 [==============================] - 1s 527ms/step
[[2.39270328e-08 1.90741822e-08 7.76825502e-08 7.92453818e-08
  1.60290128e-07 8.87892284e-08 5.42882894e-08 9.99998927e-01
  3.91918391e-08 4.17797168e-07]
 [3.15403099e-08 2.42661963e-06 9.99994159e-01 2.56797648e-07
  2.86123537e-07 3.25858707e-07 5.51403957e-07 1.65956305e-06
  1.44440563e-07 1.45353255e-07]
 [6.46729026e-09 9.99997735e-01 7.00532439e-07 1.33116259e-07
  3.17947865e-07 1.37819669e-07 1.45124972e-07 6.46433705e-07
  6.51577992e-08 2.81121935e-08]
 [9.99997020e-01 2.82281238e-08 2.18443947e-08 7.75419551e-07
  1.87574969e-07 1.09308885e-07 8.58419298e-07 1.70689304e-07
  1.90090361e-07 6.41211841e-07]]
```

```
print(np.argmax(prediction,axis=1)) #printing labels from first 4 images
print(y_test[:4]) #printing the actual labels
```

```
[7 2 1 0]
```



```
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

As we already predicted the input from the x_test. According to that by using argmax function here we are printing the labels with high prediction values

Saving the model

```
#save the model
model.save('models/mnistCNN1.h5')
```

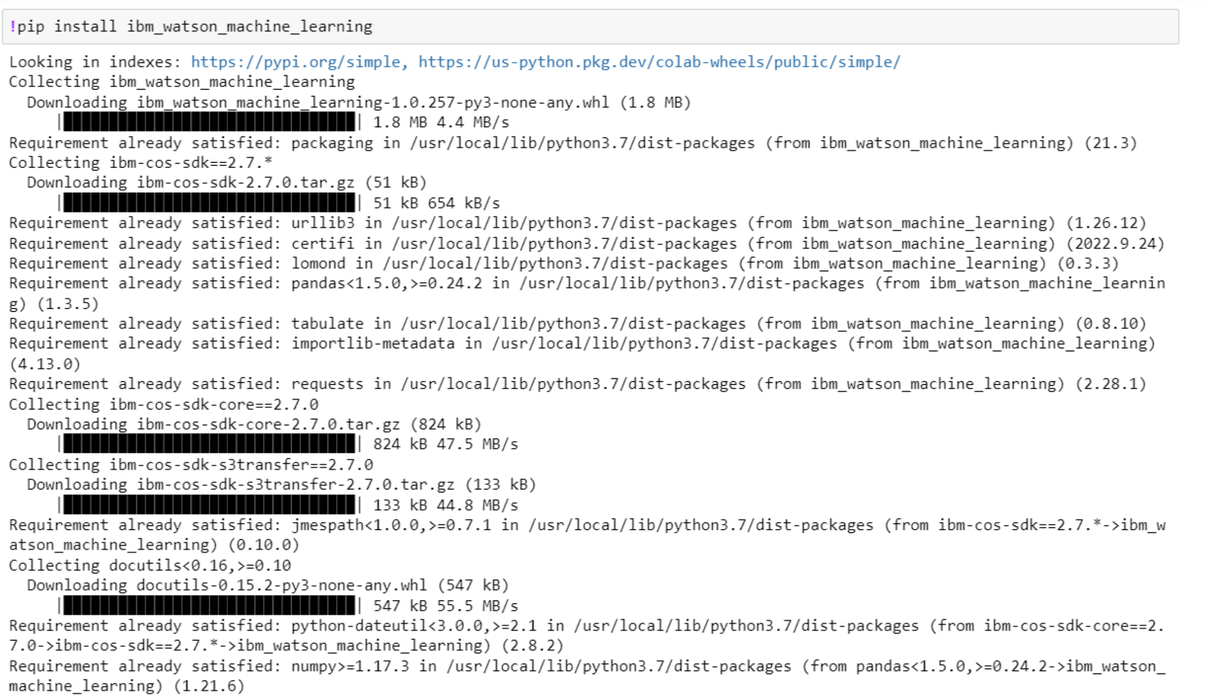
The model is saved with .h5 extension as follows: An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
# Saving in tar

!tar -zcvf digitrecmodel.tgz --absolute-names /content/models/mnistCNN1.h5
```

```
/content/models/mnistCNN1.h5
```

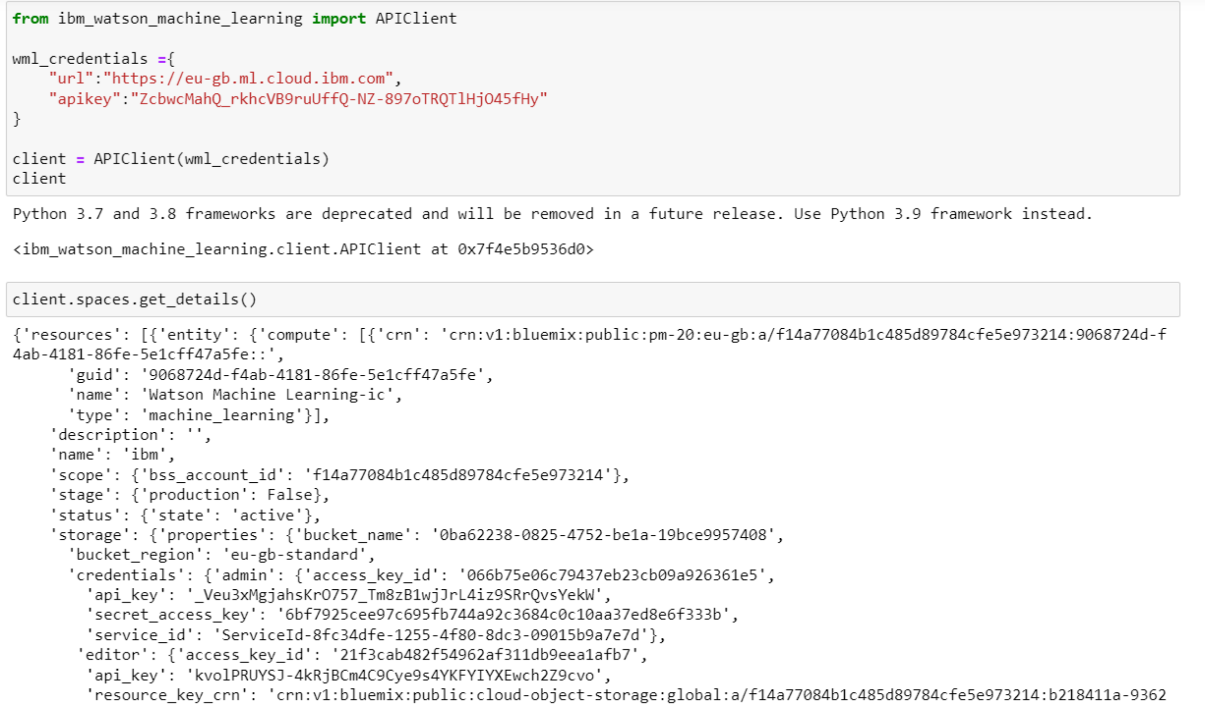
```
!pip install watson-machine-learning-client
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 4.4 MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client) (1.3.5)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client) (2022.9.
24)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client) (4.64.1)
```



```
!pip install ibm_watson_machine_learning
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ibm_watson_machine_learning
  Downloading ibm_watson_machine_learning-1.0.257-py3-none-any.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 4.4 MB/s
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning) (21.3)
Collecting ibm-cos-sdk==2.7.*
  Downloading ibm-cos-sdk-2.7.0.tar.gz (51 kB)
     |████████████████████████████████| 51 kB 654 kB/s
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning) (1.26.12)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning) (2022.9.24)
Requirement already satisfied: lomond in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learnin
g) (1.3.5)
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning) (0.8.10)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(4.13.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning) (2.28.1)
Collecting ibm-cos-sdk-core==2.7.0
  Downloading ibm-cos-sdk-core-2.7.0.tar.gz (824 kB)
     |████████████████████████████████| 824 kB 47.5 MB/s
Collecting ibm-cos-sdk-s3transfer==2.7.0
  Downloading ibm-cos-sdk-s3transfer-2.7.0.tar.gz (133 kB)
     |████████████████████████████████| 133 kB 44.8 MB/s
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.7/dist-packages (from ibm-cos-sdk==2.7.*->ibm_w
atson_machine_learning) (0.10.0)
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
     |████████████████████████████████| 547 kB 55.5 MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/dist-packages (from ibm-cos-sdk-core==2.
7.0->ibm-cos-sdk==2.7.*->ibm_watson_machine_learning) (2.8.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_
machine_learning) (1.21.6)
```

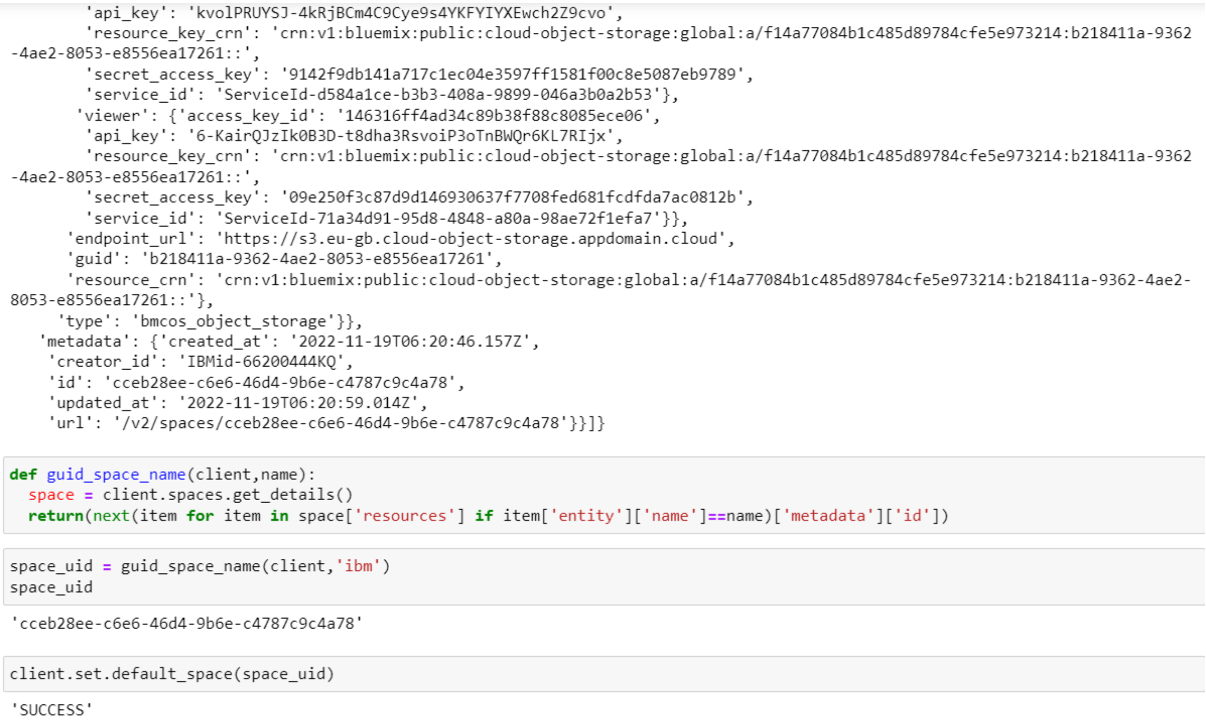


```
from ibm_watson_machine_learning import APIClient

wml_credentials ={
    "url":"https://eu-gb.ml.cloud.ibm.com",
    "apikey":"ZcbwcMahQ_rkhcVB9ruUffQ-NZ-897oTRQTlHjO45fHy"
}

client = APIClient(wml_credentials)
client
```

```
Python 3.7 and 3.8 frameworks are deprecated and will be removed in a future release. Use Python 3.9 framework instead.

<ibm_watson_machine_learning.client.APIClient at 0x7f4e5b9536d0>
```

```
client.spaces.get_details()
```

```
{'resources': [{'entity': {'compute': [{'crn': 'crn:v1:bluemix:public:pm-20:eu-gb:a/f14a77084b1c485d89784cfe5e973214:9068724d-f
4ab-4181-86fe-5e1cff47a5fe::',
      'guid': '9068724d-f4ab-4181-86fe-5e1cff47a5fe',
      'name': 'Watson Machine Learning-ic',
      'type': 'machine_learning'}],
    'description': '',
    'name': 'ibm',
    'scope': {'bss_account_id': 'f14a77084b1c485d89784cfe5e973214'},
    'stage': {'production': False},
    'status': {'state': 'active'},
    'storage': {'properties': {'bucket_name': '0ba62238-0825-4752-be1a-19bce9957408',
      'bucket_region': 'eu-gb-standard',
      'credentials': {'admin': {'access_key_id': '066b75e06c79437eb23cb09a926361e5',
        'api_key': '_Veu3xMgjahsKrO757_Tm8zB1wjJrL4iz9SRrQvsYekW',
        'secret_access_key': '6bf7925cee97c695fb744a92c3684c0c10aa37ed8e6f333b',
        'service_id': 'ServiceId-8fc34dfe-1255-4f80-8dc3-09015b9a7e7d'},
       'editor': {'access_key_id': '21f3cab482f54962af311db9eea1afb7',
        'api_key': 'kvolPRUYSJ-4kRjBCm4C9Cye9s4YKFYIYXEwch2Z9cvo',
        'resource_key_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/f14a77084b1c485d89784cfe5e973214:b218411a-9362
```



```
        'api_key': 'kvolPRUYSJ-4kRjBCm4C9Cye9s4YKFYIYXEwch2Z9cvo',
        'resource_key_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/f14a77084b1c485d89784cfe5e973214:b218411a-9362
-4ae2-8053-e8556ea17261::',
        'secret_access_key': '9142f9db141a717c1ec04e3597ff1581f00c8e5087eb9789',
        'service_id': 'ServiceId-d584a1ce-b3b3-408a-9899-046a3b0a2b53'},
       'viewer': {'access_key_id': '146316ff4ad34c89b38f88c8085ece06',
        'api_key': '6-KairQJzIk0B3D-t8dha3RsvoiP3oTnBWQr6KL7RIjx',
        'resource_key_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/f14a77084b1c485d89784cfe5e973214:b218411a-9362
-4ae2-8053-e8556ea17261::',
        'secret_access_key': '09e250f3c87d9d146930637f7708fed681fcdfda7ac0812b',
        'service_id': 'ServiceId-71a34d91-95d8-4848-a80a-98ae72f1efa7'}},
      'endpoint_url': 'https://s3.eu-gb.cloud-object-storage.appdomain.cloud',
      'guid': 'b218411a-9362-4ae2-8053-e8556ea17261',
      'resource_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/f14a77084b1c485d89784cfe5e973214:b218411a-9362-4ae2-
8053-e8556ea17261::'},
     'type': 'bmcos_object_storage'}},
   'metadata': {'created_at': '2022-11-19T06:20:46.157Z',
    'creator_id': 'IBMid-66200444KQ',
    'id': 'cceb28ee-c6e6-46d4-9b6e-c4787c9c4a78',
    'updated_at': '2022-11-19T06:20:59.014Z',
    'url': '/v2/spaces/cceb28ee-c6e6-46d4-9b6e-c4787c9c4a78'}}]}
```

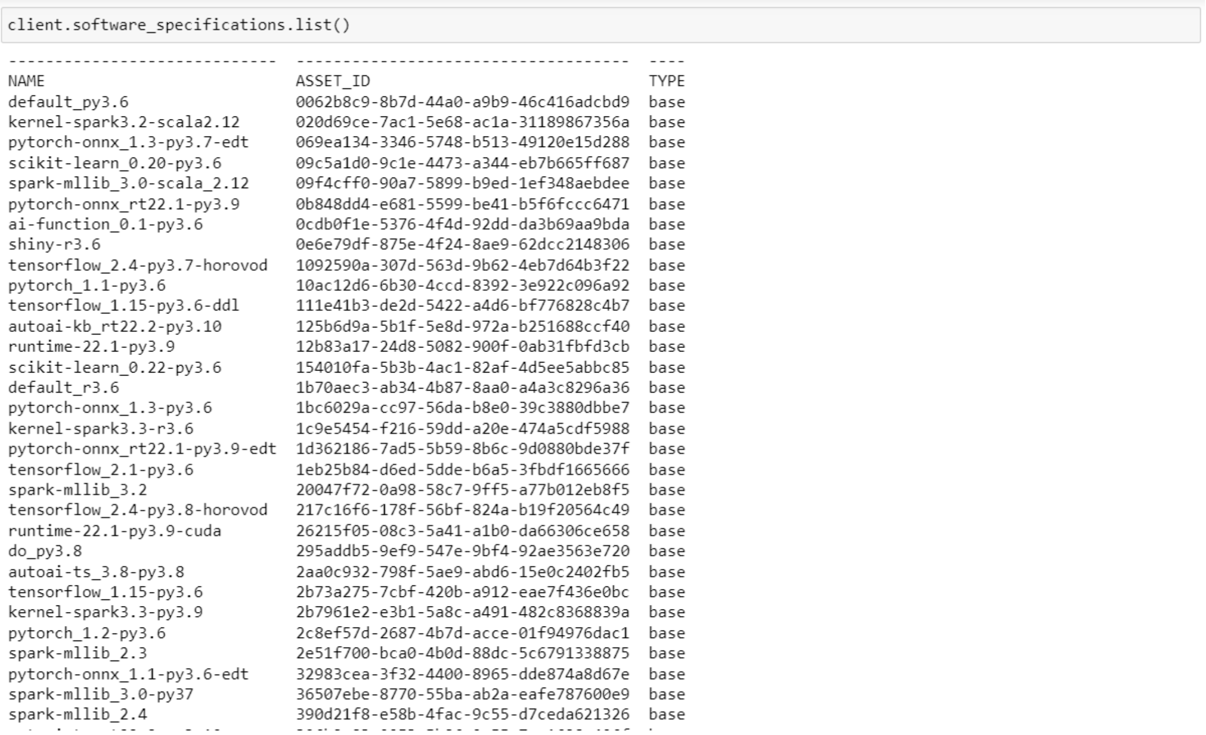
```
def guid_space_name(client,name):
  space = client.spaces.get_details()
  return(next(item for item in space['resources'] if item['entity']['name']==name)['metadata']['id'])
```

```
space_uid = guid_space_name(client,'ibm')
space_uid
```

```
'cceb28ee-c6e6-46d4-9b6e-c4787c9c4a78'
```

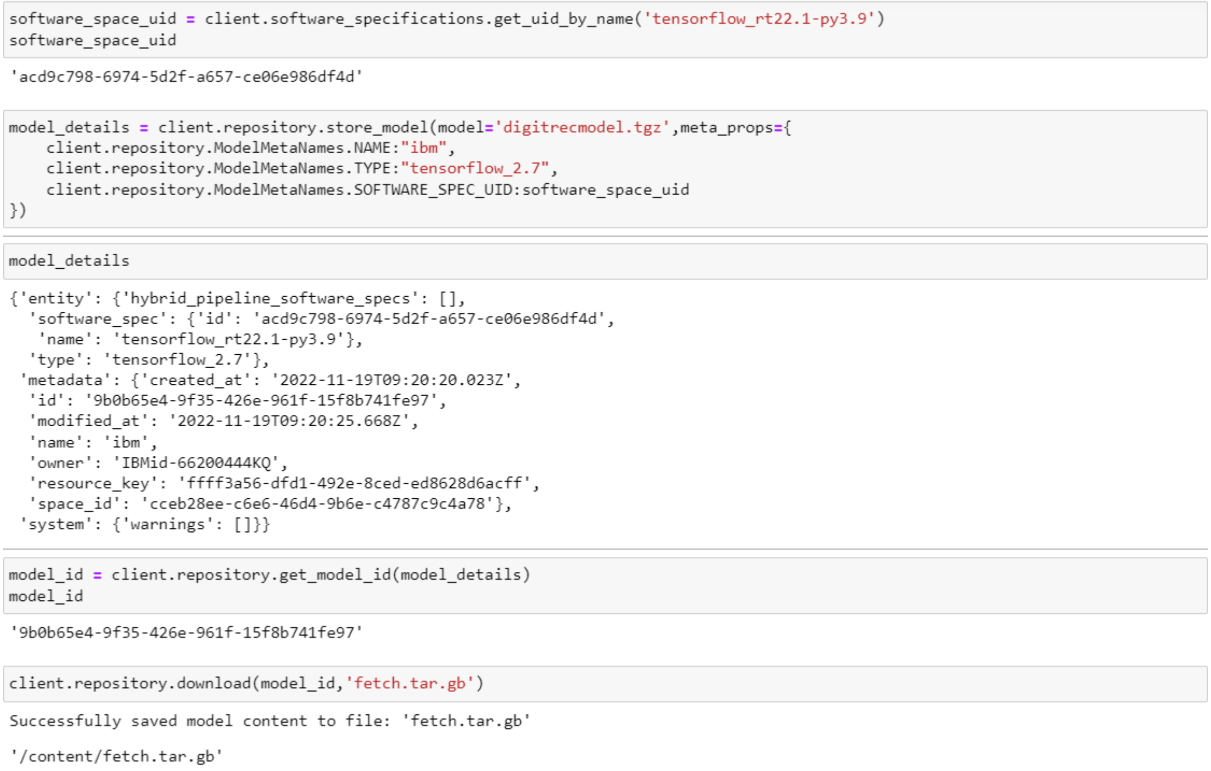
```
client.set_default_space(space_uid)
```

```
'SUCCESS'
```



```
client.software_specifications.list()
```

```
----------------------------  ------------------------------------  ----
NAME                          ASSET_ID                              TYPE
default_py3.6                 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12     020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12    09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9     0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6         0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                    0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod  1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6             10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl     111e41b3-de2d-5422-a4d6-bf776828c4b7  base
autoai-kb_rt22.2-py3.10       125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
runtime-22.1-py3.9            12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6       154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                  1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6        1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6          1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6          1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2               20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod  217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda       26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                      295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8           2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6         2b73a275-7cbf-420b-a912-eae7f436e0bc  base
kernel-spark3.3-py3.9         2b7961e2-e3b1-5a8c-a491-482c8368839a  base
pytorch_1.2-py3.6             2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3               2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt    32983cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37          36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4               390d21f8-e58b-4fac-9c55-d7ceda621326  base
```



```
software_space_uid = client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
software_space_uid
```

```
'acd9c798-6974-5d2f-a657-ce06e986df4d'
```

```
model_details = client.repository.store_model(model='digitrecmodel.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"ibm",
    client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
})
```

```
model_details
```

```
{'entity': {'hybrid_pipeline_software_specs': [],
  'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d',
   'name': 'tensorflow_rt22.1-py3.9'},
  'type': 'tensorflow_2.7'},
 'metadata': {'created_at': '2022-11-19T09:20:20.023Z',
  'id': '9b0b65e4-9f35-426e-961f-15f8b741fe97',
  'modified_at': '2022-11-19T09:20:25.668Z',
  'name': 'ibm',
  'owner': 'IBMid-66200444KQ',
  'resource_key': 'ffff3a56-dfd1-492e-8ced-ed8628d6acff',
  'space_id': 'cceb28ee-c6e6-46d4-9b6e-c4787c9c4a78'},
 'system': {'warnings': []}}
```

```
model_id = client.repository.get_model_id(model_details)
model_id
```

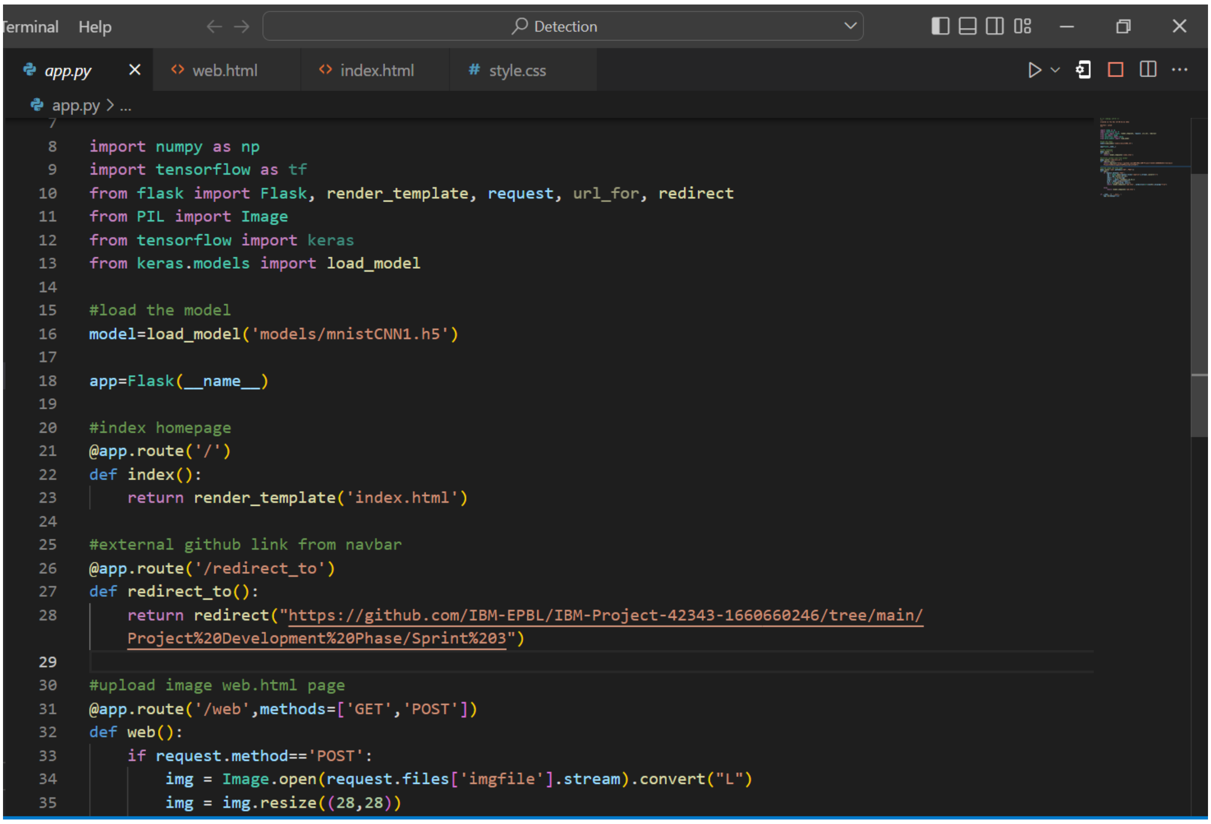
```
'9b0b65e4-9f35-426e-961f-15f8b741fe97'
```

```
client.repository.download(model_id,'fetch.tar.gb')
```

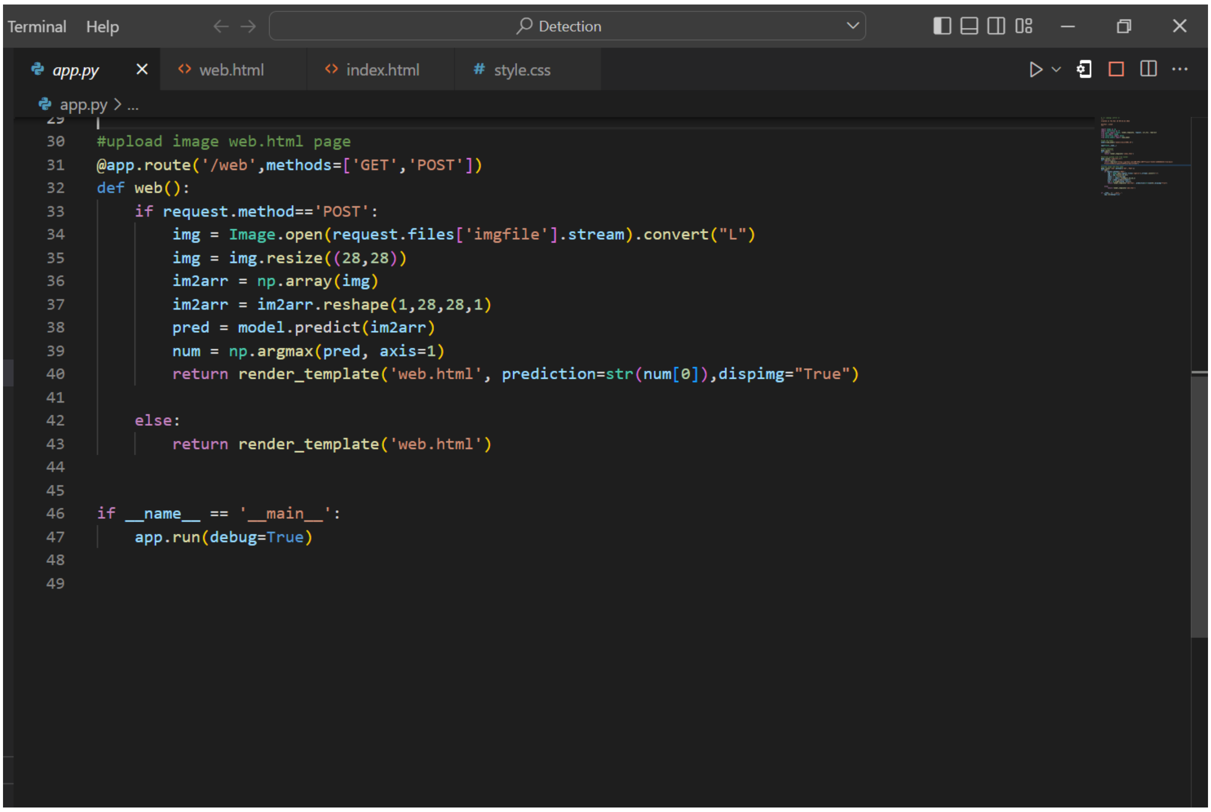
```
Successfully saved model content to file: 'fetch.tar.gb'

'/content/fetch.tar.gb'
```

UI BUILDING

FLASK APPLICATION

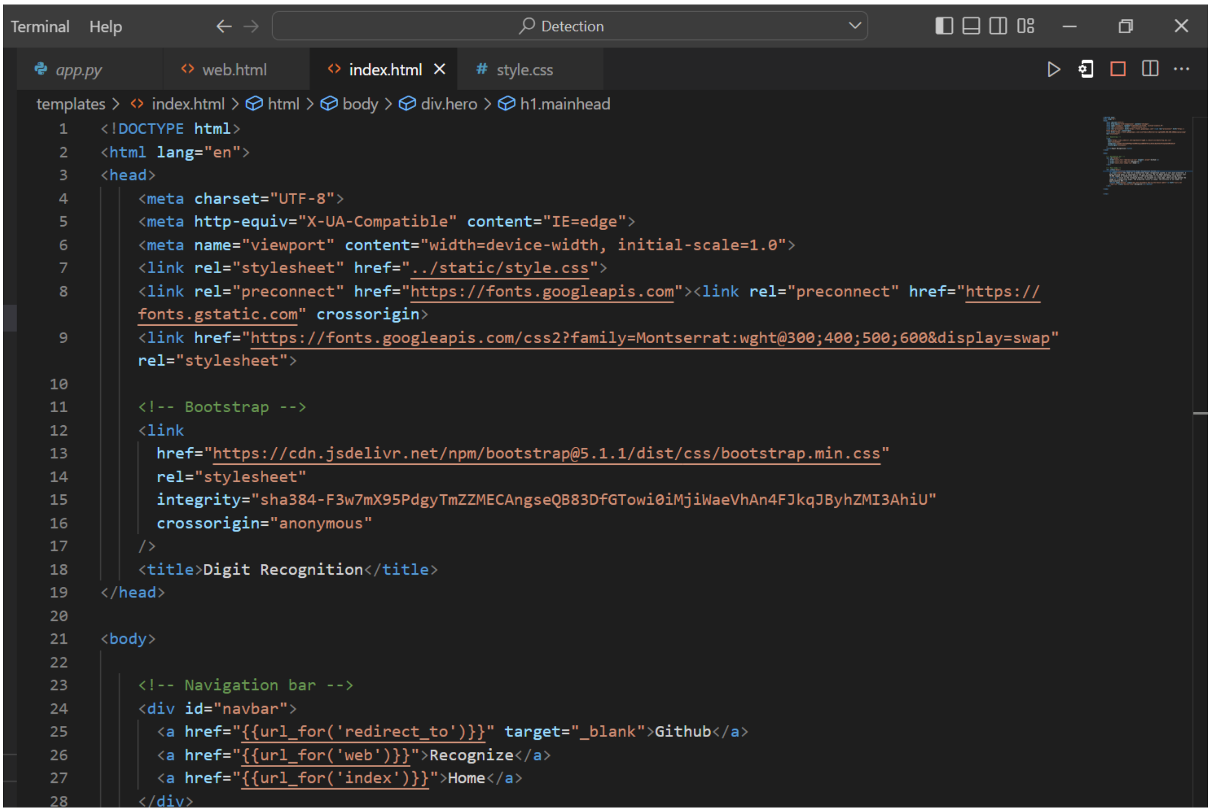


```python
import numpy as np
import tensorflow as tf
from flask import Flask, render_template, request, url_for, redirect
from PIL import Image
from tensorflow import keras
from keras.models import load_model

#load the model
model=load_model('models/mnistCNN1.h5')

app=Flask(__name__)

#index homepage
@app.route('/')
def index():
    return render_template('index.html')

#external github link from navbar
@app.route('/redirect_to')
def redirect_to():
    return redirect("https://github.com/IBM-EPBL/IBM-Project-42343-1660660246/tree/main/
    Project%20Development%20Phase/Sprint%203")

#upload image web.html page
@app.route('/web',methods=['GET','POST'])
def web():
    if request.method=='POST':
        img = Image.open(request.files['imgfile'].stream).convert("L")
        img = img.resize((28,28))
```

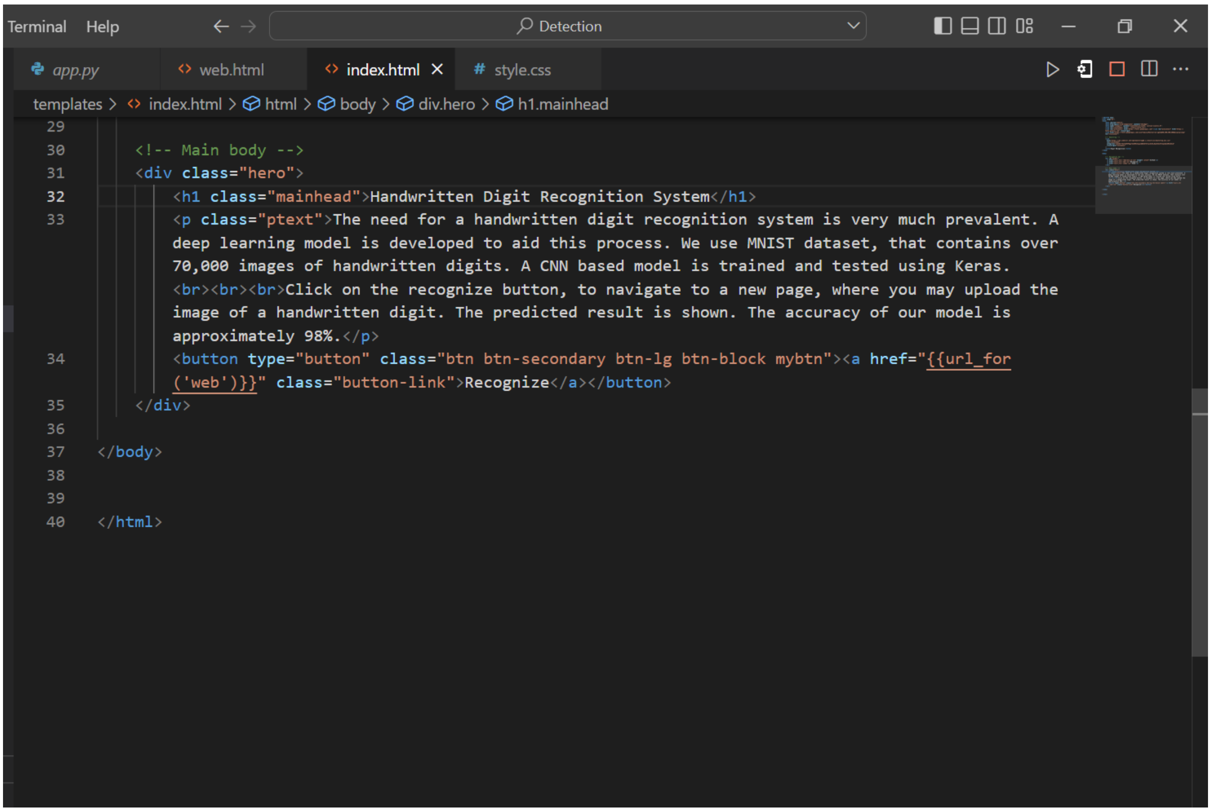


```python
#upload image web.html page
@app.route('/web',methods=['GET','POST'])
def web():
    if request.method=='POST':
        img = Image.open(request.files['imgfile'].stream).convert("L")
        img = img.resize((28,28))
        im2arr = np.array(img)
        im2arr = im2arr.reshape(1,28,28,1)
        pred = model.predict(im2arr)
        num = np.argmax(pred, axis=1)
        return render_template('web.html', prediction=str(num[0]),dispimg="True")

    else:
        return render_template('web.html')


if __name__ == '__main__':
    app.run(debug=True)
```

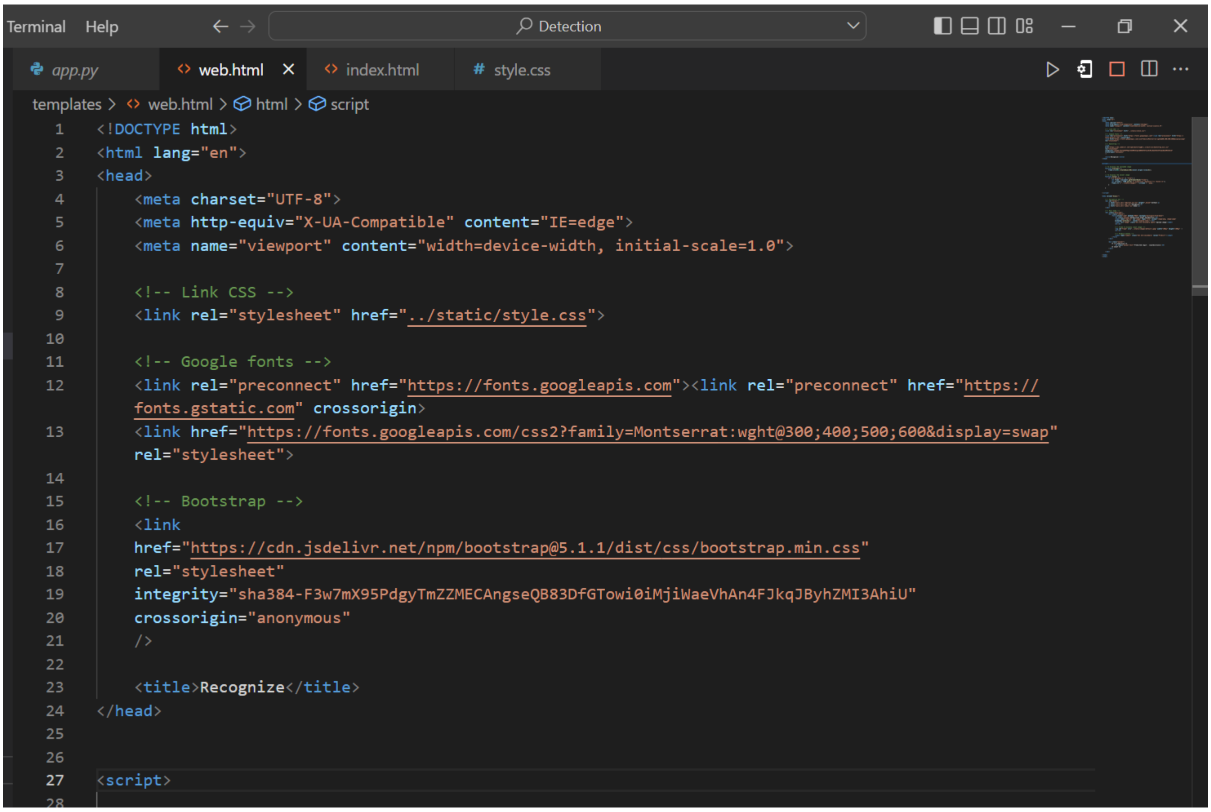
TEMPLATES - HTML FILES


Terminal
Help
Detection
app.py
web.html
index.html
style.css
templates > index.html > html > body > div.hero > h1.mainhead

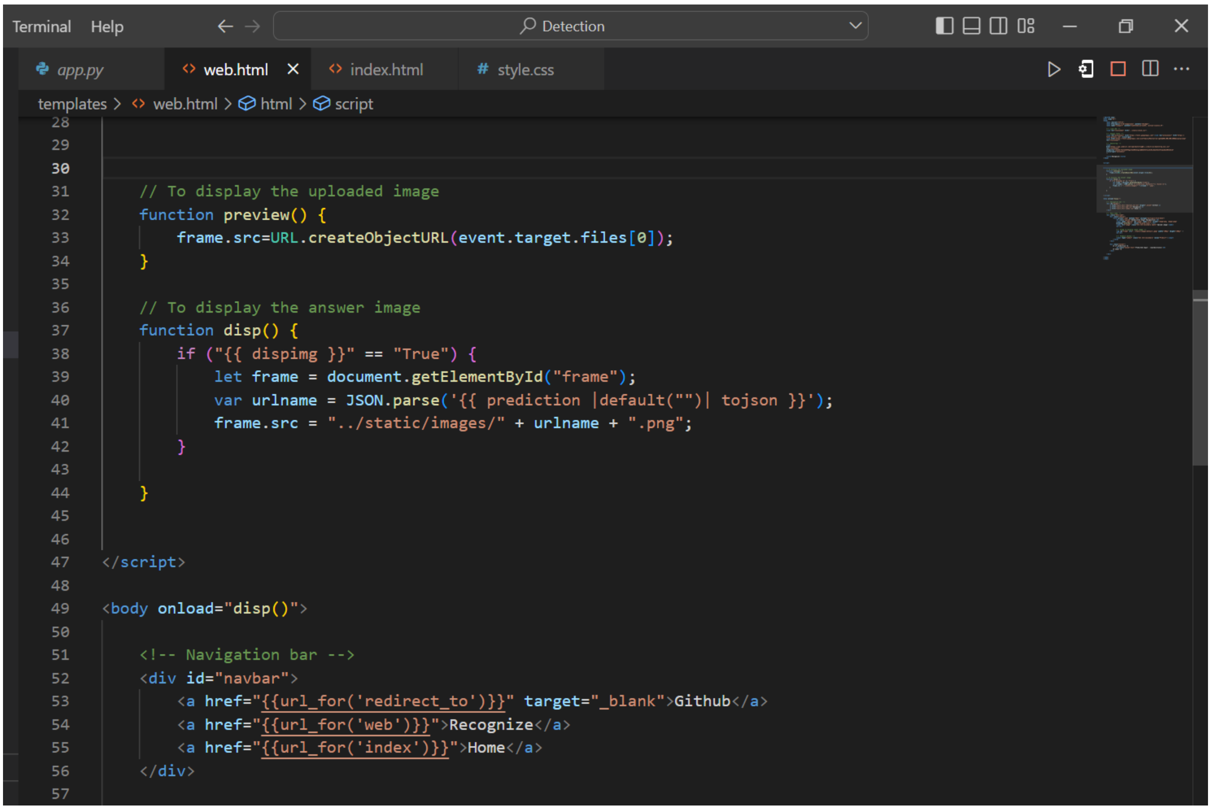
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../static/style.css">
    <link rel="preconnect" href="https://fonts.googleapis.com"><link rel="preconnect" href="https://
    fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;500;600&display=swap"
    rel="stylesheet">

    <!-- Bootstrap -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
      crossorigin="anonymous"
    />
    <title>Digit Recognition</title>
</head>

<body>

    <!-- Navigation bar -->
    <div id="navbar">
      <a href="{{url_for('redirect_to')}}" target="_blank">Github</a>
      <a href="{{url_for('web')}}">Recognize</a>
      <a href="{{url_for('index')}}">Home</a>
    </div>
```


Terminal
Help
Detection
app.py
web.html
index.html
style.css
templates > index.html > html > body > div.hero > h1.mainhead

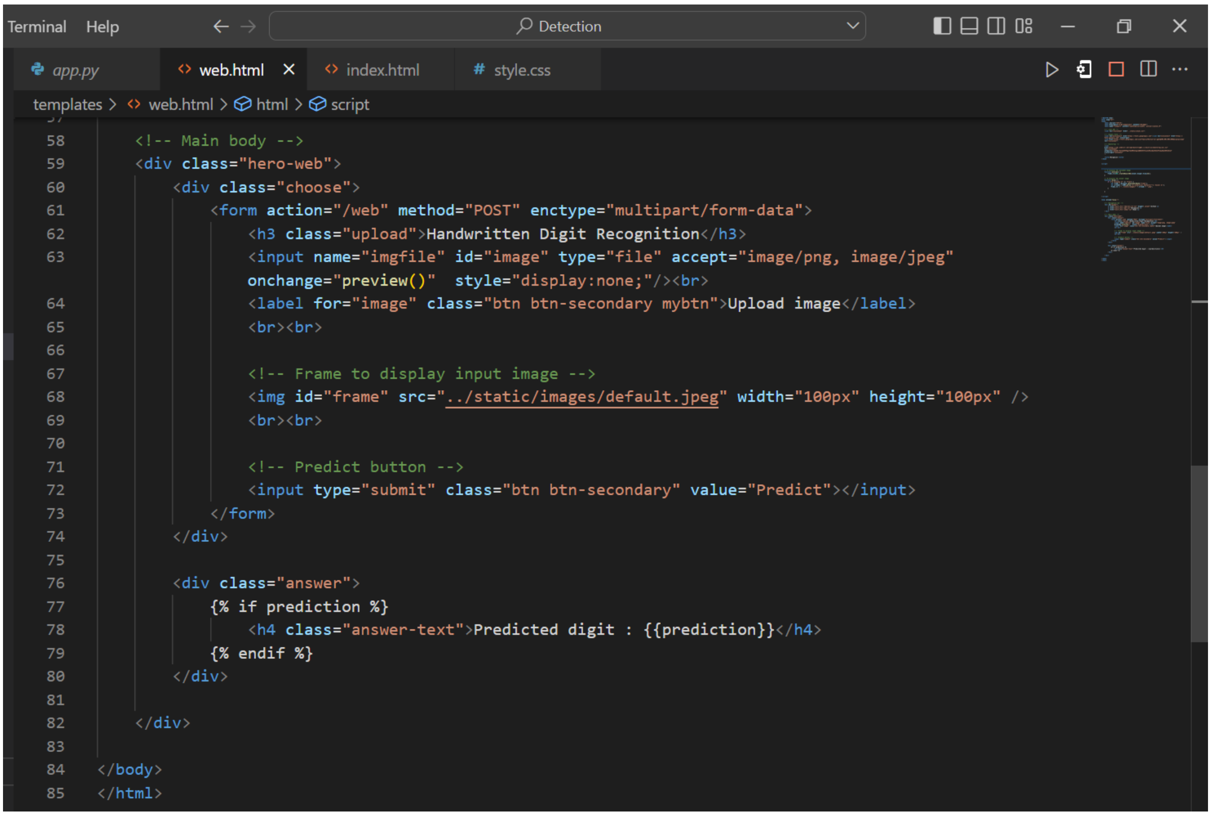
```html

    <!-- Main body -->
    <div class="hero">
        <h1 class="mainhead">Handwritten Digit Recognition System</h1>
        <p class="ptext">The need for a handwritten digit recognition system is very much prevalent. A
        deep learning model is developed to aid this process. We use MNIST dataset, that contains over
        70,000 images of handwritten digits. A CNN based model is trained and tested using Keras.
        <br><br><br>Click on the recognize button, to navigate to a new page, where you may upload the
        image of a handwritten digit. The predicted result is shown. The accuracy of our model is
        approximately 98%.</p>
        <button type="button" class="btn btn-secondary btn-lg btn-block mybtn"><a href="{{url_for
        ('web')}}" class="button-link">Recognize</a></button>
    </div>

</body>


</html>
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Link CSS -->
    <link rel="stylesheet" href="../static/style.css">

    <!-- Google fonts -->
    <link rel="preconnect" href="https://fonts.googleapis.com"><link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;500;600&display=swap" rel="stylesheet">

    <!-- Bootstrap -->
    <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
    crossorigin="anonymous"
    />

    <title>Recognize</title>
</head>


<script>
```

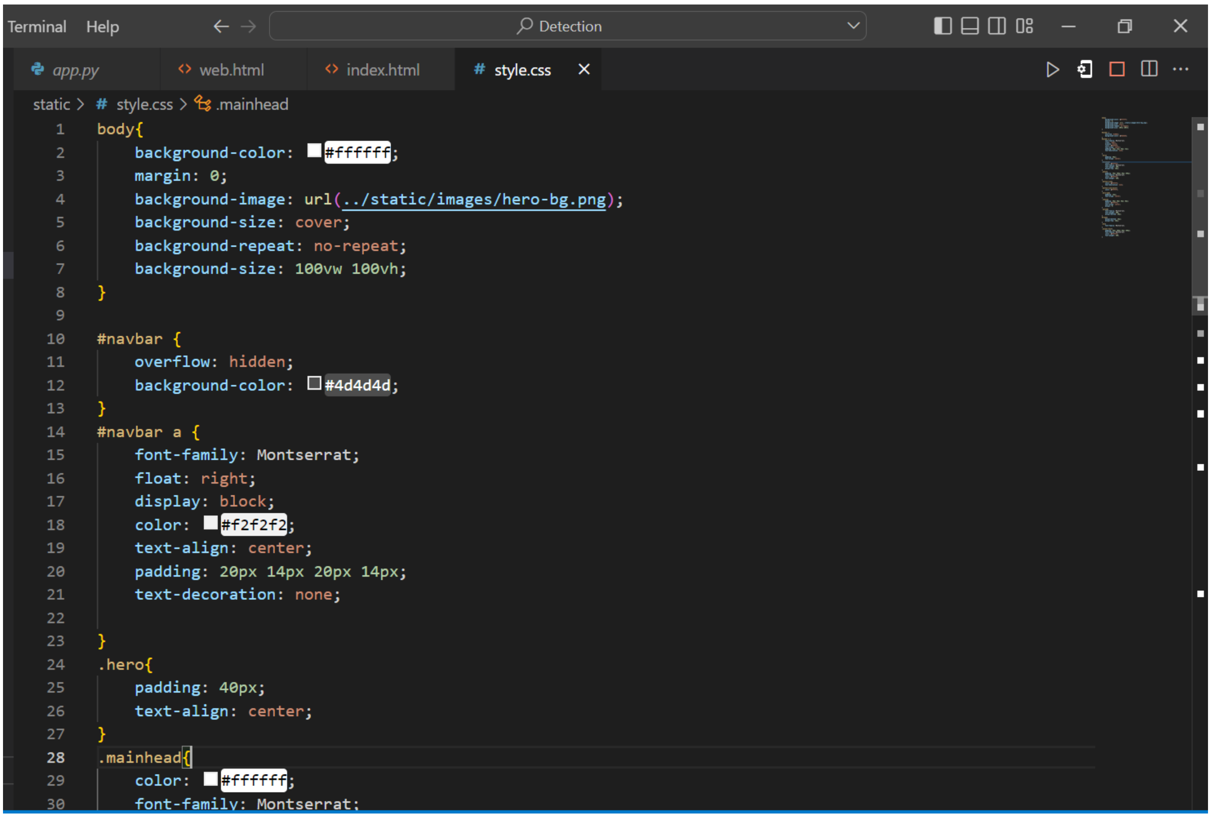


```html



    // To display the uploaded image
    function preview() {
        frame.src=URL.createObjectURL(event.target.files[0]);
    }

    // To display the answer image
    function disp() {
        if ("{{ dispimg }}" == "True") {
            let frame = document.getElementById("frame");
            var urlname = JSON.parse('{{ prediction |default("")| tojson }}');
            frame.src = "../static/images/" + urlname + ".png";
        }

    }


</script>

<body onload="disp()">

    <!-- Navigation bar -->
    <div id="navbar">
        <a href="{{url_for('redirect_to')}}" target="_blank">Github</a>
        <a href="{{url_for('web')}}">Recognize</a>
        <a href="{{url_for('index')}}">Home</a>
    </div>

```

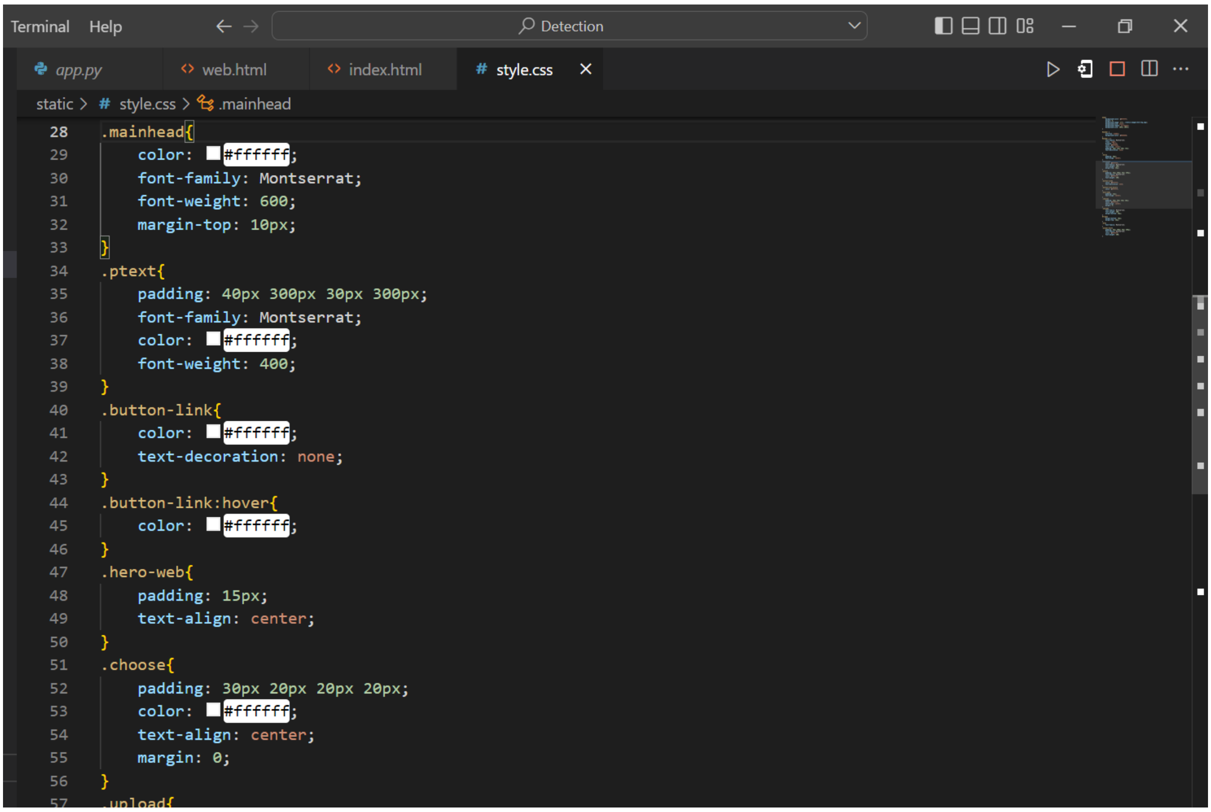


```html
    <!-- Main body -->
    <div class="hero-web">
        <div class="choose">
            <form action="/web" method="POST" enctype="multipart/form-data">
                <h3 class="upload">Handwritten Digit Recognition</h3>
                <input name="imgfile" id="image" type="file" accept="image/png, image/jpeg"
                onchange="preview()"  style="display:none;"/><br>
                <label for="image" class="btn btn-secondary mybtn">Upload image</label>
                <br><br>

                <!-- Frame to display input image -->
                <img id="frame" src="../static/images/default.jpeg" width="100px" height="100px" />
                <br><br>

                <!-- Predict button -->
                <input type="submit" class="btn btn-secondary" value="Predict"></input>
            </form>
        </div>

        <div class="answer">
            {% if prediction %}
                <h4 class="answer-text">Predicted digit : {{prediction}}</h4>
            {% endif %}
        </div>

    </div>

</body>
</html>
```

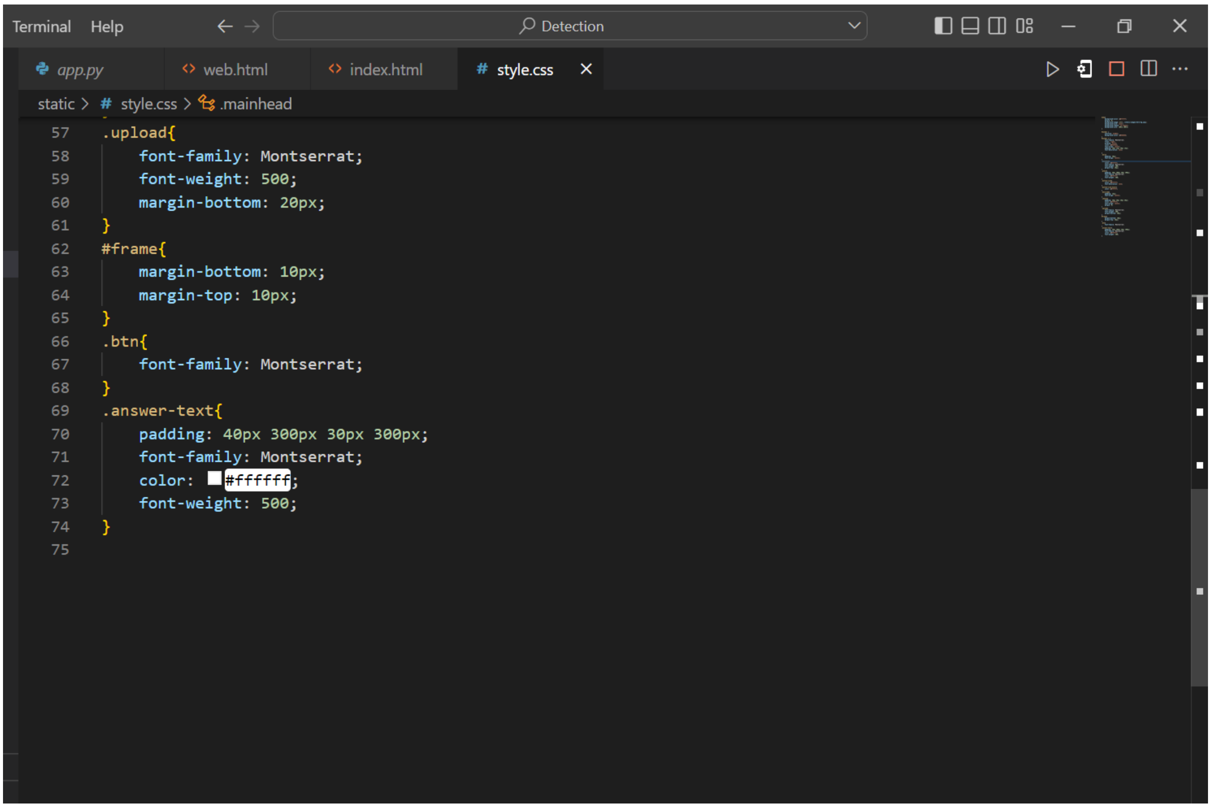
STATIC - CSS FILES



```css
body{
    background-color: #ffffff;
    margin: 0;
    background-image: url(../static/images/hero-bg.png);
    background-size: cover;
    background-repeat: no-repeat;
    background-size: 100vw 100vh;
}

#navbar {
    overflow: hidden;
    background-color: #4d4d4d;
}
#navbar a {
    font-family: Montserrat;
    float: right;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 20px 14px 20px 14px;
    text-decoration: none;

}
.hero{
    padding: 40px;
    text-align: center;
}
.mainhead{
    color: #ffffff;
    font-family: Montserrat;
```



```css
.mainhead{
    color: #ffffff;
    font-family: Montserrat;
    font-weight: 600;
    margin-top: 10px;
}
.ptext{
    padding: 40px 300px 30px 300px;
    font-family: Montserrat;
    color: #ffffff;
    font-weight: 400;
}
.button-link{
    color: #ffffff;
    text-decoration: none;
}
.button-link:hover{
    color: #ffffff;
}
.hero-web{
    padding: 15px;
    text-align: center;
}
.choose{
    padding: 30px 20px 20px 20px;
    color: #ffffff;
    text-align: center;
    margin: 0;
}
.upload{
```



```css
.upload{
    font-family: Montserrat;
    font-weight: 500;
    margin-bottom: 20px;
}
#frame{
    margin-bottom: 10px;
    margin-top: 10px;
}
.btn{
    font-family: Montserrat;
}
.answer-text{
    padding: 40px 300px 30px 300px;
    font-family: Montserrat;
    color: #ffffff;
    font-weight: 500;
}
```

GITHUB LINK

DEMO LINK