In [1]:

```
#implementing Feedforward neural networks with Keras and TensorFlow
#a. Import the necessary packages
#b. Load the training and testing data (MNIST/CIFAR10)
#c. Define the network architecture using Keras
#d. Train the model using SGD
#e. Evaluate the network
#f. Plot the training loss and accuracy"""
```

In [2]:

```
import numpy as np
import random
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras.datasets import mnist
import warnings
#warnings.filterwarnings("Ignore",category='UserWarning')
```

```
WARNING:tensorflow:From C:\Users\kalya\anaconda3\envs\Lib\site-packages\keras
\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprec
ated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:

```
#Load the  dataset (MNIST)
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

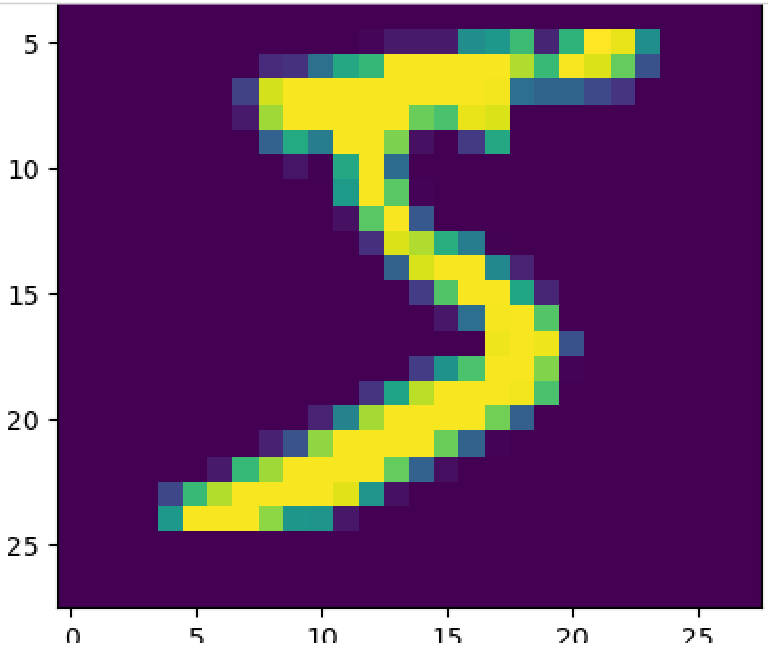
In [4]:

```
#shape of training dataset 60000 images having 28*28 size
print(f"Shape of X_train {x_train.shape}")
print(f"Shape of y_train {y_train.shape}")
#shape of testing dataset 10000 images having 28*28 size
print(f"Shape of x_test  {x_test.shape}")
print(f"Shape of y_test  {y_test.shape}")
```

```
Shape of X_train (60000, 28, 28)
Shape of y_train (60000,)
Shape of x_test  (10000, 28, 28)
Shape of y_test  (10000,)
```

In [5]:
```python
plt.imshow(x_train[0])
#plt.matshow(x_train[0])

#printing corresponding label
print(y_train[0])
```



In [6]:
```python
x_train[0]
```

```
         0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

In [7]:
```python
x_train = x_train/255
x_test = x_test/255
```

In [8]:

```
x_train[0]
```

Out[8]:

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
```

Define Network Architecture Using Keras

In [9]:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),             #flatten the input
    keras.layers.Dense(128,activation='relu',name='L1'),     #hidden layer
    keras.layers.Dense(10,activation='softmax',name='L2')   #output layer
])
```

```
WARNING:tensorflow:From C:\Users\kalya\anaconda3\envs\Lib\site-packages\keras
\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use t
f.compat.v1.get_default_graph instead.
```

In [10]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 L1 (Dense)                  (None, 128)               100480

 L2 (Dense)                  (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train the Model Using SGD

In [11]:

```
model.compile(optimizer="sgd",
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\kalya\anaconda3\envs\Lib\site-packages\keras
\src\optimizers\__init__.py:309: The name tf.train.Optimizer is deprecated. Pl
ease use tf.compat.v1.train.Optimizer instead.
```

In [12]:

```
# Training the model
history = model.fit(x_train, y_train,
                          epochs=10,
              validation_data=(x_test, y_test),
              shuffle=True)
```

```
Epoch 1/10
WARNING:tensorflow:From C:\Users\kalya\anaconda3\envs\Lib\site-packages\keras
\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecate
d. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\kalya\anaconda3\envs\Lib\site-packages\keras
\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_fun
ctions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functi
ons instead.

1875/1875 [==============================] - 4s 2ms/step - loss: 0.6530 - accu
racy: 0.8363 - val_loss: 0.3516 - val_accuracy: 0.9055
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3347 - accu
racy: 0.9067 - val_loss: 0.2883 - val_accuracy: 0.9204
Epoch 3/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2853 - accu
racy: 0.9191 - val_loss: 0.2546 - val_accuracy: 0.9299
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2547 - accu
racy: 0.9284 - val_loss: 0.2317 - val_accuracy: 0.9349
Epoch 5/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2317 - accu
racy: 0.9353 - val_loss: 0.2148 - val_accuracy: 0.9392
Epoch 6/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2132 - accu
racy: 0.9406 - val_loss: 0.1978 - val_accuracy: 0.9449
Epoch 7/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.1979 - accu
racy: 0.9445 - val_loss: 0.1857 - val_accuracy: 0.9461
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1848 - accu
racy: 0.9480 - val_loss: 0.1761 - val_accuracy: 0.9492
Epoch 9/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1737 - accu
racy: 0.9509 - val_loss: 0.1667 - val_accuracy: 0.9511
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1638 - accu
racy: 0.9543 - val_loss: 0.1594 - val_accuracy: 0.9535
```

Evaluate the Network

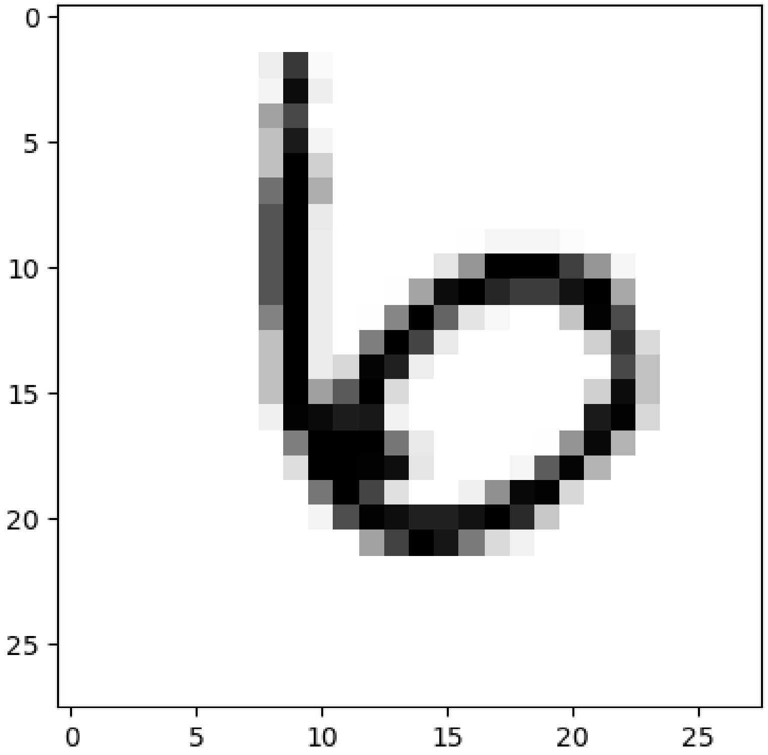
In [13]:

```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

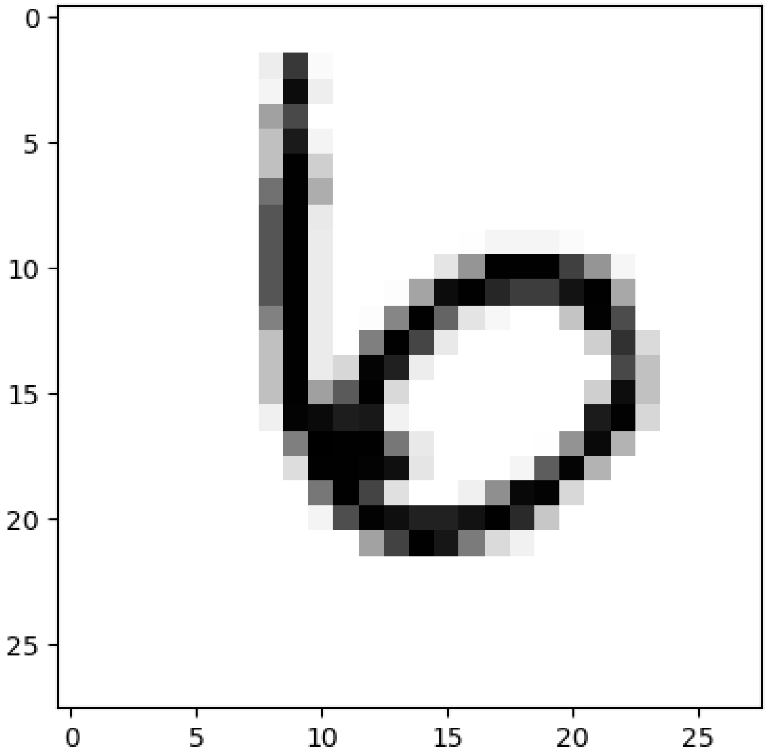
```
313/313 [==============================] - 0s 1ms/step - loss: 0.1594 - accura
cy: 0.9535
loss=0.159
Accuracy=0.953
```

In [14]:

```
n=random.randint(0,9999)
plt.imshow(x_test[n],cmap='Greys')
plt.show()
predicted_value=model.predict(x_test)
plt.imshow(x_test[n],cmap='Greys')
plt.show()
print("the number is= %d" %np.argmax(predicted_value[n]))
```



```
313/313 [==============================] - 0s 1ms/step
```



```
the number is= 6
```

In [15]:

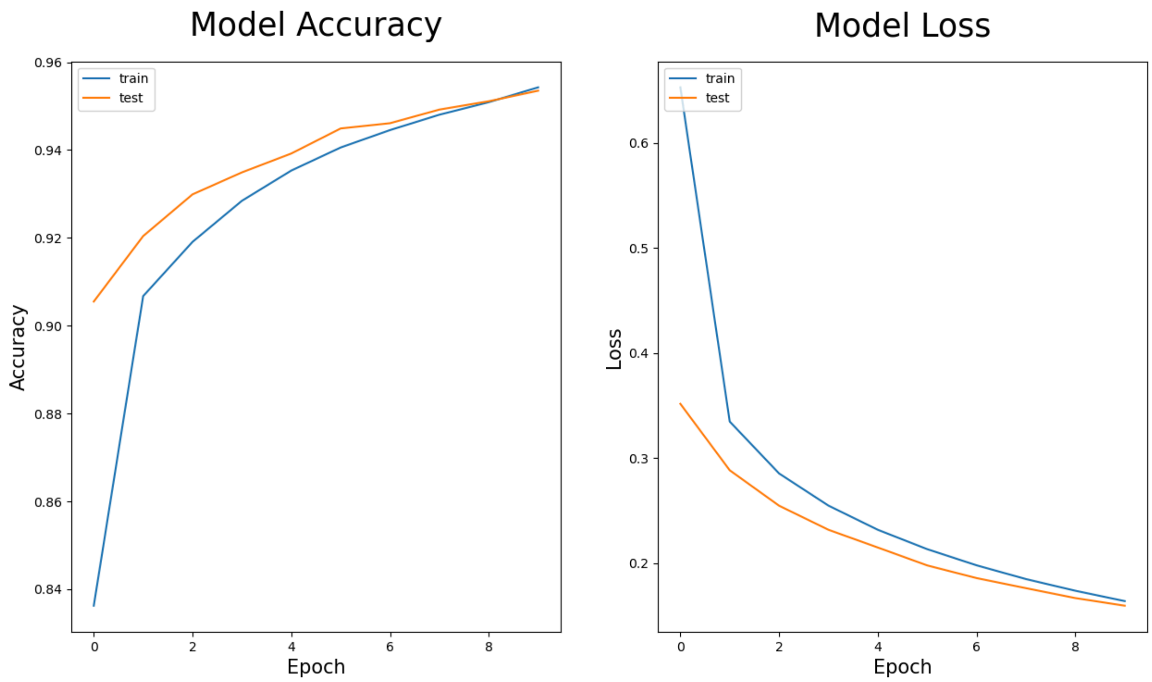
```
# Plot the Training Loss and Accuracy"
```

In [16]:

```
plt.figure(figsize=[15,8])

# summarize history for accuracy
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy', size=25, pad=20)
plt.ylabel('Accuracy', size=15)
plt.xlabel('Epoch', size=15)
plt.legend(['train', 'test'], loc='upper left')


# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss', size=25, pad=20)
plt.ylabel('Loss', size=15)
plt.xlabel('Epoch', size=15)
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [ ]: