In [4]:
```
import tensorflow as tf
from keras.models import Sequential
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
import random
```

```
C:\Users\Admin\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumP
y version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected versio
n 1.26.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [5]:
```
(x_train,y_train),(x_test,y_test)=mnist.load_data()
x_train=x_train/255
x_test=x_test/255
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
st.npz
11490434/11490434 [==============================] - 3s 0us/step
```

In [6]:
```
import keras
model=keras.Sequential()
model.add(keras.layers.Flatten(input_shape=(28,28)))
model.add(keras.layers.Dense(128,activation='relu'))
model.add(keras.layers.Dense(10,activation='softmax'))
model.summary()
```

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

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

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

In [7]:
```
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=["Accur
```

In [8]:
```
H=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.6536 - Accuracy:
0.8343 - val_loss: 0.3645 - val_Accuracy: 0.8991
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3381 - Accuracy:
0.9054 - val_loss: 0.2942 - val_Accuracy: 0.9172
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2880 - Accuracy:
0.9197 - val_loss: 0.2595 - val_Accuracy: 0.9269
Epoch 4/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2572 - Accuracy:
0.9283 - val_loss: 0.2370 - val_Accuracy: 0.9321
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2340 - Accuracy:
0.9345 - val_loss: 0.2192 - val_Accuracy: 0.9375
```

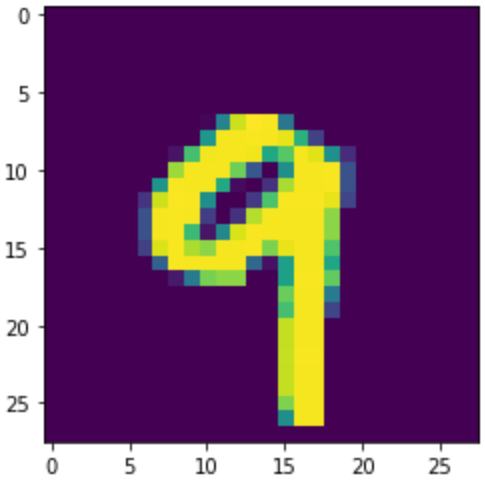
In [9]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.2192 - Accuracy: 0.9
375
```

In [10]:
```
print("Loss=%.3f"%test_loss)
print("Accuracy=%.3f"%test_acc)
```

```
Loss=0.219
Accuracy=0.938
```

In [11]:
```
n=random.randint(0,999)
plt.imshow(x_test[n])
plt.show()
```



In [12]:
```
prediction=model.predict(x_test)
print("The handwritten number in the image is %d"%np.argmax(prediction[n]))
```

```
313/313 [==============================] - 1s 2ms/step
The handwritten number in the image is 9
```