In [1]:
```
#a)Importing all necessary libraries
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import random
```

In [2]:
```
#b) load the training and testing data

mnist=tf.keras.datasets.mnist
(x_train,y_train), (x_test,y_test) = mnist.load_data()

x_train=x_train / 255
x_test=x_test / 255
```

In [3]:
```
#c)Define the network architecture using keras

model=keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
     keras.layers.Dense(128,activation="relu"),
     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: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

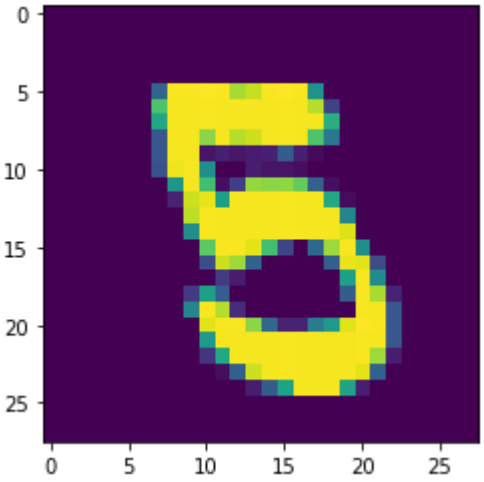
In [4]:
```
#D) train the model using SGD
model.compile(optimizer="sgd",
              loss="sparse_categorical_crossentropy",metrics=['accuracy'] )

history=model.fit(x_train, y_train, validation_data = (x_test,y_test),epochs=2)
```

```
Epoch 1/2
1875/1875 [==============================] - 13s 5ms/step - loss: 0.6603 - accuracy: 0.8
297 - val_loss: 0.3623 - val_accuracy: 0.9025
Epoch 2/2
1875/1875 [==============================] - 8s 5ms/step - loss: 0.3390 - accuracy: 0.90
54 - val_loss: 0.2938 - val_accuracy: 0.9175
```

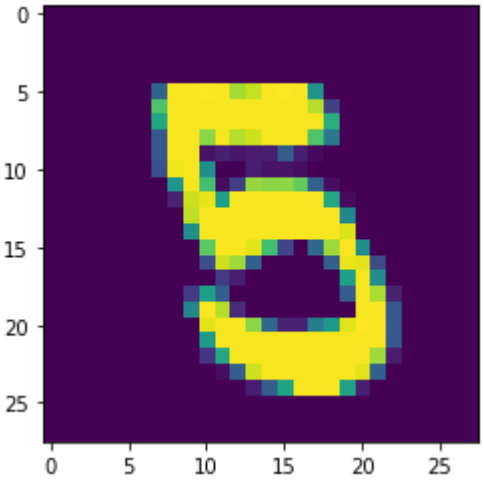
In [6]:

```
#e)Evaluate the network

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

n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()

print('Predicted value:',predicted_value[n])
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.2938 - accuracy: 0.9175
loss= 0.29380837082862854
Accuracy=  1
```




```
313/313 [==============================] - 1s 3ms/step
```

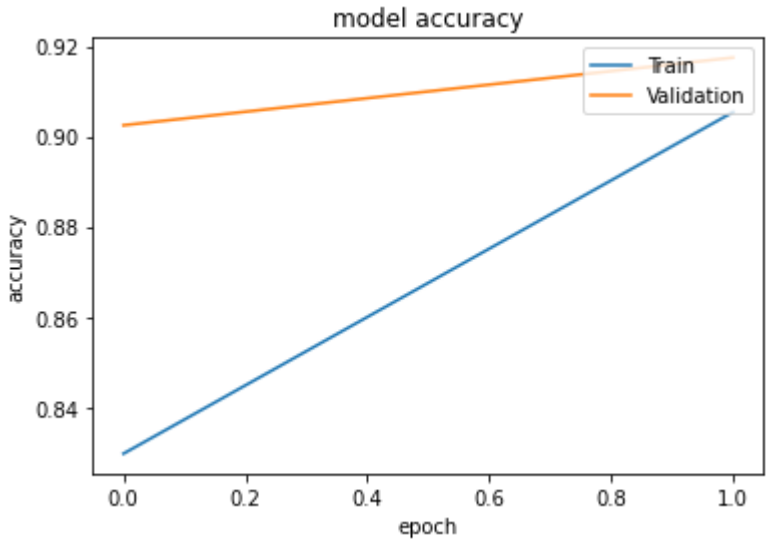



```
Predicted value: [6.0450719e-05 1.7695747e-05 5.6496286e-03 2.4301729e-01 9.3484348e-05
 2.9342890e-01 2.4848527e-04 1.3473382e-06 4.5594931e-01 1.5333917e-03]
```

In [8]:

```
#f plot the training loss and accuracy


#TRAINING ACCUARACY
plt.plot(history.history['accuracy'])
```

```
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'],loc="upper right")
plt.show()
```



In [9]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'],loc="upper left")
plt.show()
```