In [2]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
from tensorflow import keras
```

In [3]:
```
mnist=tf.keras.datasets.mnist
(xtrain,ytrain),(xtest,ytest)=mnist.load_data()
```

In [4]:
```
xtrain.shape
```

Out[4]: (60000, 28, 28)

In [5]:
```
xtest.shape
```

Out[5]: (10000, 28, 28)

In [6]:
```
xtrain=xtrain/255
xtest=xtest/255
```

In [7]:
```
xtrain.min(),xtrain.max()
```

Out[7]: (np.float64(0.0), np.float64(1.0))

In [8]:
```
xtrain=xtrain.reshape(xtrain.shape+(1,))
xtest=xtest.reshape(xtest.shape+(1,))
```

In [9]:
```
model=keras.Sequential([
    keras.layers.Conv2D(32,(3,3),activation="relu",input_shape=(28,28,1)),
    keras.layers.MaxPooling2D((2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128,activation="relu"),
    keras.layers.Dense(64,activation="relu"),
    keras.layers.Dense(10,activation="softmax")
])
model.summary()
model.compile(optimizer="sgd",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
```

```
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer u
sing an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 128)	692,352
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 10)	650

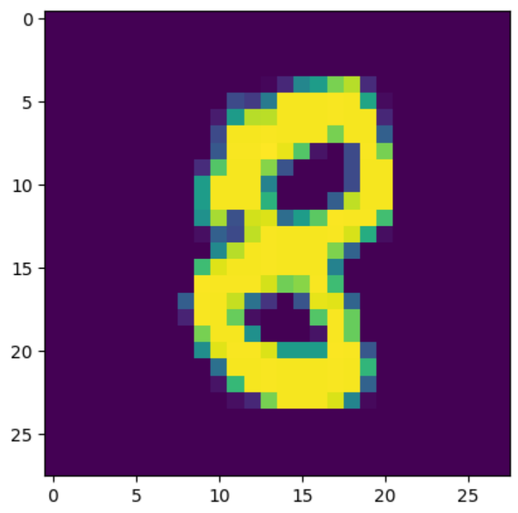
Total params: 701,578 (2.68 MB)
Trainable params: 701,578 (2.68 MB)
Non-trainable params: 0 (0.00 B)

In [10]:
```
history=model.fit(xtrain,ytrain,epochs=10,batch_size=32)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 48s 24ms/step - accuracy: 0.8570 - loss: 0.4952
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 25ms/step - accuracy: 0.9404 - loss: 0.1964
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 80s 24ms/step - accuracy: 0.9554 - loss: 0.1468
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 45s 24ms/step - accuracy: 0.9638 - loss: 0.1187
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 25ms/step - accuracy: 0.9702 - loss: 0.0991
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 24ms/step - accuracy: 0.9739 - loss: 0.0864
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 24ms/step - accuracy: 0.9769 - loss: 0.0763
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 25ms/step - accuracy: 0.9794 - loss: 0.0682
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 24ms/step - accuracy: 0.9813 - loss: 0.0609
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 24ms/step - accuracy: 0.9832 - loss: 0.0556
```

In [11]:
```
p=model.predict((xtest))
r=random.randint(0,9999)
plt.imshow(xtest[r])
print(p[r])
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step
[2.5835161e-05 5.4108500e-07 3.4306453e-05 3.6297435e-05 1.1489771e-08
 6.9135217e-06 4.6808843e-04 4.9496851e-10 9.9940515e-01 2.2894061e-05]
```



In [12]:
```
model.evaluate(xtest,ytest)
```

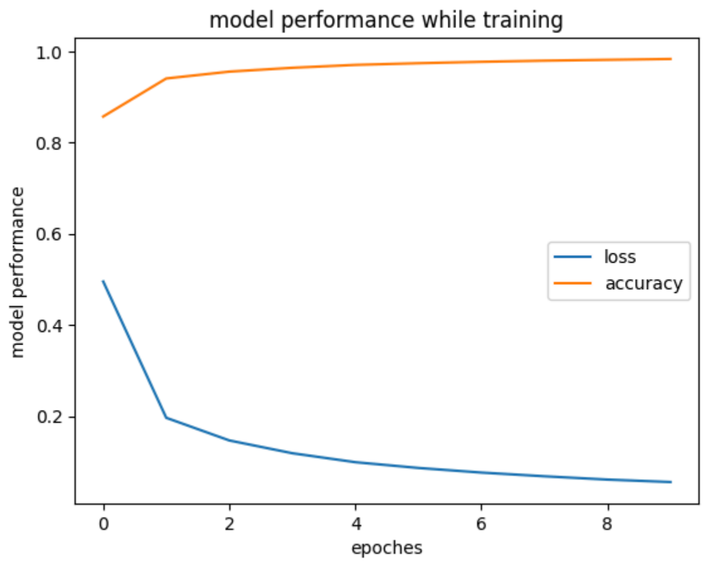
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - accuracy: 0.9788 - loss: 0.0680
```

Out[12]: [0.06800468266010284, 0.9787999987602234]

In [14]:
```
plt.plot(history.history["loss"])
plt.plot(history.history["accuracy"])

plt.xlabel("epoches")
plt.ylabel("model performance")
plt.title("model performance while training")
plt.legend(["loss","accuracy"],loc='best')
```

Out[14]: <matplotlib.legend.Legend at 0x2203509f770>


model performance while training
model performance
epoches
loss
accuracy
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8

In [ ]: