In [6]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler
import numpy as np
from sklearn.model_selection import train_test_split
import os, cv2
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
```

In [7]:

```
data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/hmnist_28_28_RGB.csv')
data.head()
```

Out[7]:

	pixel0000	pixel0001	pixel0002	pixel0003	pixel0004	pixel0005	pixel0006	pixel0007	pixel0008	pixel0009	...	pixel2343	pix
0	192	153	193	195	155	192	197	154	185	202	...	173	
1	25	14	30	68	48	75	123	93	126	158	...	60	
2	192	138	153	200	145	163	201	142	160	206	...	167	
3	38	19	30	95	59	72	143	103	119	171	...	44	
4	158	113	139	194	144	174	215	162	191	225	...	209	

5 rows × 2353 columns

In [9]:

```
data['label'].unique()
```

Out[9]:

```
array([2, 4, 3, 6, 5, 1, 0])
```

In [10]:

```
y = data['label']
x = data.drop(columns = ['label'])
```

In [11]:

```
data.isnull().sum().sum()  #no null values present
```

Out[11]:

```
0
```

In [12]:

```
meta_data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')
meta_data.head()
```

Out[12]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp

	lesion_id	image_id	dx	dx_type	age	sex	localization
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [13]:

```
meta_data['dx'].unique()
```

Out[13]:

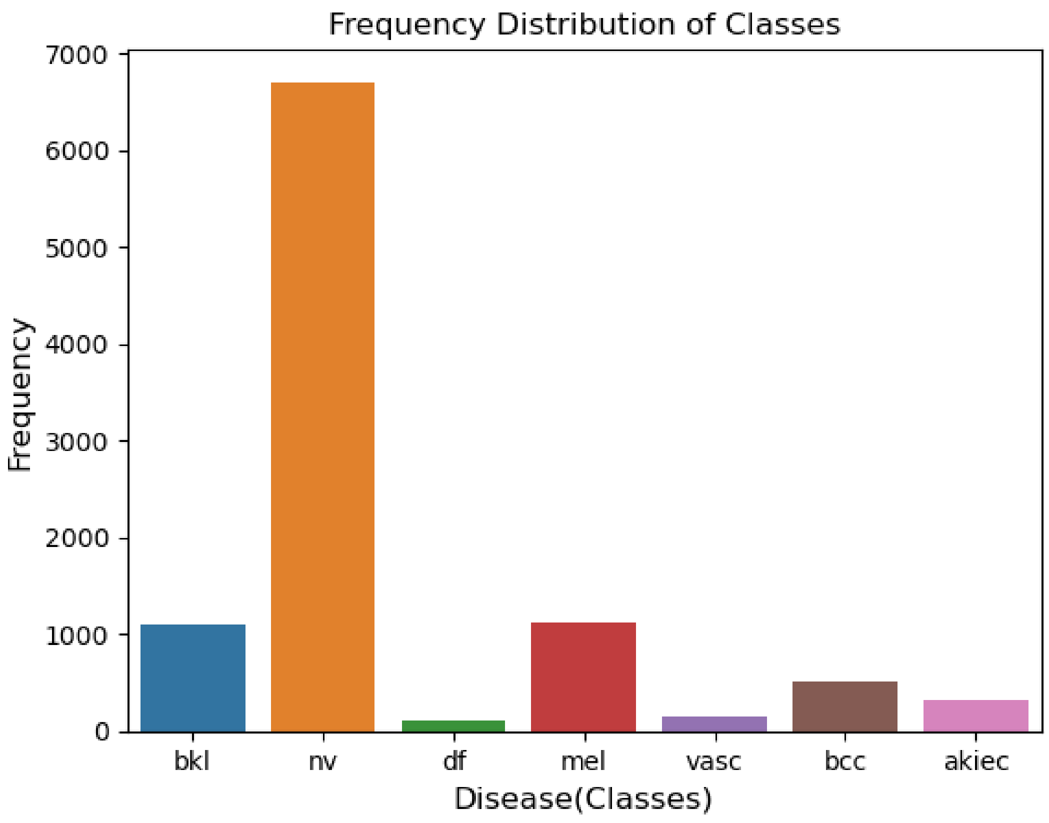
```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

In [14]:

```
sns.countplot(x = 'dx', data = meta_data)
plt.xlabel('Disease(Classes)', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes')
```

Out[14]:

```
Text(0.5, 1.0, 'Frequency Distribution of Classes')
```



In [15]:

```
print(x.shape,y.shape)
# To overcome class imbalace
oversample = RandomOverSampler()
x,y  = oversample.fit_resample(x,y)
print(x.shape,y.shape)
```

```
(10015, 2352) (10015,)
(46935, 2352) (46935,)
```

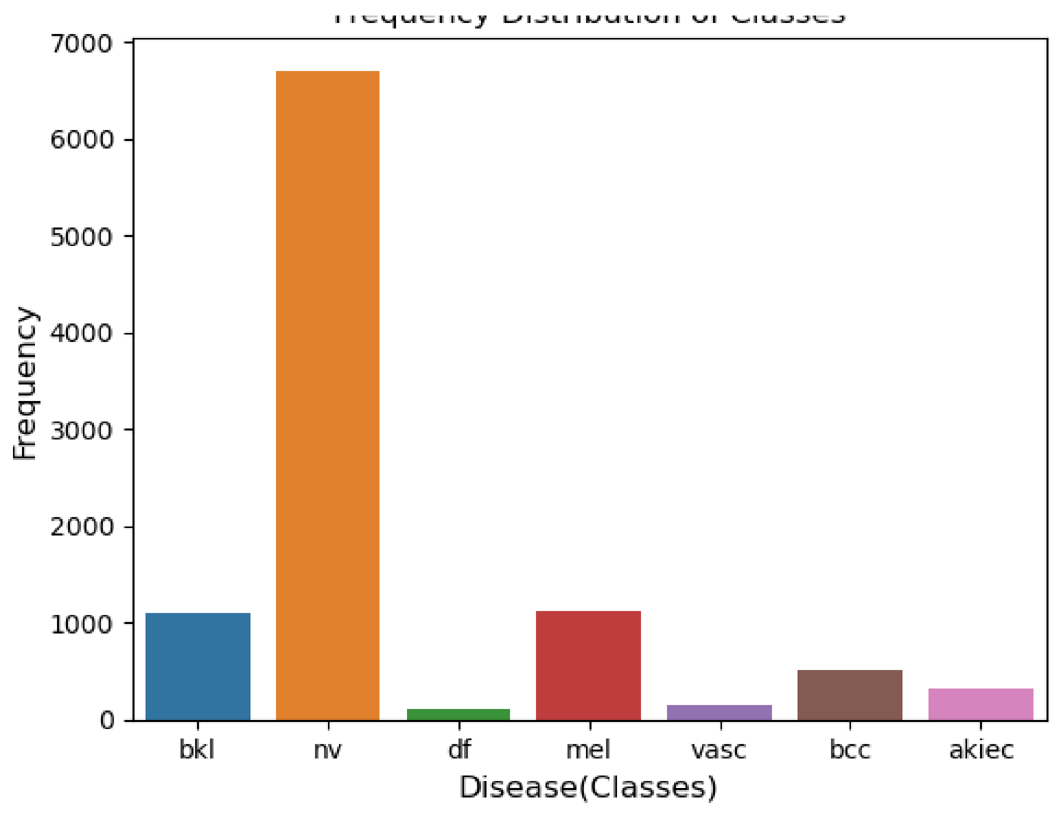
In [176]:

```
sns.countplot(x = 'dx', data = meta_data)
plt.xlabel('Disease(Classes)', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes')
```

Out[176]:

```
Text(0.5, 1.0, 'Frequency Distribution of Classes')
```

Frequency Distribution of Classes



In [ ]:

In [32]:

```
# reshaping the data so that it can be taken by convolution neural network(without distur
bing the no. of samples)
x = np.array(x).reshape(-1,28,28,3)
print('Shape of X :',x.shape)
print('Shape of y :',y.shape)
```

```
Shape of X : (46935, 28, 28, 3)
Shape of y : (46935,)
```

In [33]:

```
# Splitting Data
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.2, random_state=1)
print(X_train.shape,Y_train.shape)
print(X_test.shape , Y_test.shape)
```

```
(37548, 28, 28, 3) (37548,)
(9387, 28, 28, 3) (9387,)
```

In [18]:

```
model = Sequential()

model.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation = 'relu'
))
model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(7, activation='softmax'))
model.summary()
```

```
Model: "sequential"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 16)        448       
                                                                 
 conv2d_1 (Conv2D)           (None, 24, 24, 32)        4640      
                                                                 
 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0         
 )                                                               
                                                                 
 conv2d_2 (Conv2D)           (None, 10, 10, 32)        9248      
                                                                 
 conv2d_3 (Conv2D)           (None, 8, 8, 64)          18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 4, 4, 64)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 1024)              0         
                                                                 
 dense (Dense)               (None, 64)                65600     
                                                                 
 dense_1 (Dense)             (None, 7)                 455       
                                                                 
=================================================================
Total params: 98,887
Trainable params: 98,887
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
model.compile(loss = 'sparse_categorical_crossentropy',
             optimizer = 'adam',
              metrics = ['accuracy'])
history = model.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 128,
                    epochs = 50)
```

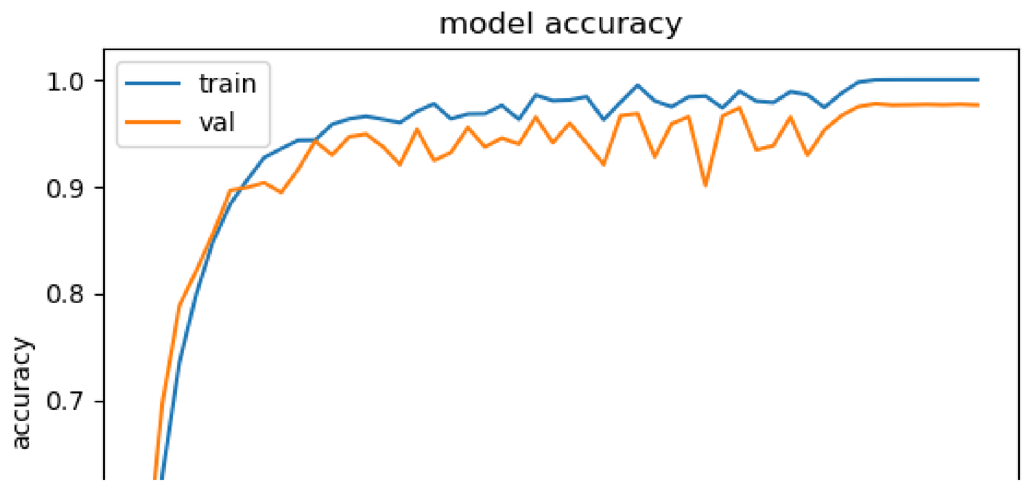
```
Epoch 1/50
235/235 [==============================] - 27s 108ms/step - loss: 2.2407 - accuracy: 0.42
00 - val_loss: 1.1698 - val_accuracy: 0.5382
Epoch 2/50
235/235 [==============================] - 25s 106ms/step - loss: 0.9816 - accuracy: 0.62
96 - val_loss: 0.8212 - val_accuracy: 0.6968
Epoch 3/50
235/235 [==============================] - 25s 105ms/step - loss: 0.7226 - accuracy: 0.73
48 - val_loss: 0.6027 - val_accuracy: 0.7881
Epoch 4/50
235/235 [==============================] - 24s 103ms/step - loss: 0.5639 - accuracy: 0.79
94 - val_loss: 0.5137 - val_accuracy: 0.8212
Epoch 5/50
235/235 [==============================] - 24s 104ms/step - loss: 0.4348 - accuracy: 0.84
91 - val_loss: 0.4219 - val_accuracy: 0.8565
Epoch 6/50
235/235 [==============================] - 25s 105ms/step - loss: 0.3399 - accuracy: 0.88
34 - val_loss: 0.3213 - val_accuracy: 0.8963
Epoch 7/50
235/235 [==============================] - 25s 105ms/step - loss: 0.2770 - accuracy: 0.90
62 - val_loss: 0.3263 - val_accuracy: 0.8991
Epoch 8/50
235/235 [==============================] - 24s 103ms/step - loss: 0.2200 - accuracy: 0.92
70 - val_loss: 0.2835 - val_accuracy: 0.9035
Epoch 9/50
235/235 [==============================] - 25s 105ms/step - loss: 0.1927 - accuracy: 0.93
53 - val_loss: 0.3118 - val_accuracy: 0.8943
Epoch 10/50
235/235 [==============================] - 25s 104ms/step - loss: 0.1705 - accuracy: 0.94
32 - val_loss: 0.2698 - val_accuracy: 0.9158
Epoch 11/50
235/235 [==============================] - 25s 105ms/step - loss: 0.1648 - accuracy: 0.94
34 - val_loss: 0.1969 - val_accuracy: 0.9425
```

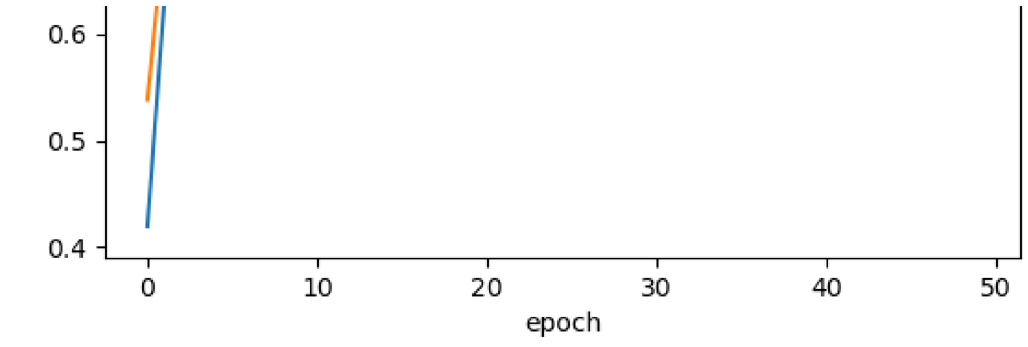
```
Epoch 12/50
235/235 [==============================] - 24s 104ms/step - loss: 0.1233 - accuracy: 0.95
81 - val_loss: 0.2373 - val_accuracy: 0.9294
Epoch 13/50
235/235 [==============================] - 26s 110ms/step - loss: 0.1073 - accuracy: 0.96
31 - val_loss: 0.1820 - val_accuracy: 0.9463
Epoch 14/50
235/235 [==============================] - 25s 105ms/step - loss: 0.1016 - accuracy: 0.96
56 - val_loss: 0.1913 - val_accuracy: 0.9490
Epoch 15/50
235/235 [==============================] - 25s 106ms/step - loss: 0.1078 - accuracy: 0.96
27 - val_loss: 0.2291 - val_accuracy: 0.9372
Epoch 16/50
235/235 [==============================] - 25s 106ms/step - loss: 0.1148 - accuracy: 0.95
97 - val_loss: 0.2536 - val_accuracy: 0.9205
Epoch 17/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0866 - accuracy: 0.97
03 - val_loss: 0.1994 - val_accuracy: 0.9535
Epoch 18/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0672 - accuracy: 0.97
74 - val_loss: 0.2455 - val_accuracy: 0.9242
Epoch 19/50
235/235 [==============================] - 25s 105ms/step - loss: 0.1038 - accuracy: 0.96
32 - val_loss: 0.2551 - val_accuracy: 0.9317
Epoch 20/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0945 - accuracy: 0.96
78 - val_loss: 0.1941 - val_accuracy: 0.9551
Epoch 21/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0927 - accuracy: 0.96
81 - val_loss: 0.2799 - val_accuracy: 0.9370
Epoch 22/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0723 - accuracy: 0.97
59 - val_loss: 0.2433 - val_accuracy: 0.9453
Epoch 23/50
235/235 [==============================] - 25s 104ms/step - loss: 0.1083 - accuracy: 0.96
27 - val_loss: 0.2846 - val_accuracy: 0.9397
Epoch 24/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0428 - accuracy: 0.98
56 - val_loss: 0.1760 - val_accuracy: 0.9648
Epoch 25/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0564 - accuracy: 0.98
03 - val_loss: 0.2706 - val_accuracy: 0.9409
Epoch 26/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0557 - accuracy: 0.98
09 - val_loss: 0.1837 - val_accuracy: 0.9591
Epoch 27/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0481 - accuracy: 0.98
39 - val_loss: 0.2371 - val_accuracy: 0.9402
Epoch 28/50
235/235 [==============================] - 25s 106ms/step - loss: 0.1195 - accuracy: 0.96
23 - val_loss: 0.2985 - val_accuracy: 0.9205
Epoch 29/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0661 - accuracy: 0.97
90 - val_loss: 0.1858 - val_accuracy: 0.9663
Epoch 30/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0163 - accuracy: 0.99
47 - val_loss: 0.1853 - val_accuracy: 0.9682
Epoch 31/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0617 - accuracy: 0.98
01 - val_loss: 0.2936 - val_accuracy: 0.9276
Epoch 32/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0806 - accuracy: 0.97
46 - val_loss: 0.2099 - val_accuracy: 0.9586
Epoch 33/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0478 - accuracy: 0.98
38 - val_loss: 0.1715 - val_accuracy: 0.9652
Epoch 34/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0452 - accuracy: 0.98
46 - val_loss: 0.4645 - val_accuracy: 0.9008
Epoch 35/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0833 - accuracy: 0.97
35 - val_loss: 0.1911 - val_accuracy: 0.9659
```

```
Epoch 36/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0314 - accuracy: 0.98
91 - val_loss: 0.1458 - val_accuracy: 0.9736
Epoch 37/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0642 - accuracy: 0.97
98 - val_loss: 0.3125 - val_accuracy: 0.9342
Epoch 38/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0683 - accuracy: 0.97
85 - val_loss: 0.2829 - val_accuracy: 0.9382
Epoch 39/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0346 - accuracy: 0.98
87 - val_loss: 0.2152 - val_accuracy: 0.9650
Epoch 40/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0460 - accuracy: 0.98
59 - val_loss: 0.3936 - val_accuracy: 0.9293
Epoch 41/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0846 - accuracy: 0.97
39 - val_loss: 0.2402 - val_accuracy: 0.9527
Epoch 42/50
235/235 [==============================] - 24s 102ms/step - loss: 0.0421 - accuracy: 0.98
71 - val_loss: 0.2066 - val_accuracy: 0.9663
Epoch 43/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0082 - accuracy: 0.99
76 - val_loss: 0.1750 - val_accuracy: 0.9747
Epoch 44/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0014 - accuracy: 0.99
98 - val_loss: 0.1577 - val_accuracy: 0.9775
Epoch 45/50
235/235 [==============================] - 25s 105ms/step - loss: 6.0696e-04 - accuracy:
1.0000 - val_loss: 0.1758 - val_accuracy: 0.9762
Epoch 46/50
235/235 [==============================] - 24s 104ms/step - loss: 2.2905e-04 - accuracy:
1.0000 - val_loss: 0.1793 - val_accuracy: 0.9764
Epoch 47/50
235/235 [==============================] - 24s 104ms/step - loss: 1.7136e-04 - accuracy:
1.0000 - val_loss: 0.1845 - val_accuracy: 0.9768
Epoch 48/50
235/235 [==============================] - 24s 104ms/step - loss: 1.3468e-04 - accuracy:
1.0000 - val_loss: 0.1942 - val_accuracy: 0.9764
Epoch 49/50
235/235 [==============================] - 24s 104ms/step - loss: 1.1190e-04 - accuracy:
1.0000 - val_loss: 0.1918 - val_accuracy: 0.9770
Epoch 50/50
235/235 [==============================] - 25s 105ms/step - loss: 9.4261e-05 - accuracy:
1.0000 - val_loss: 0.1981 - val_accuracy: 0.9763
```

In [34]:

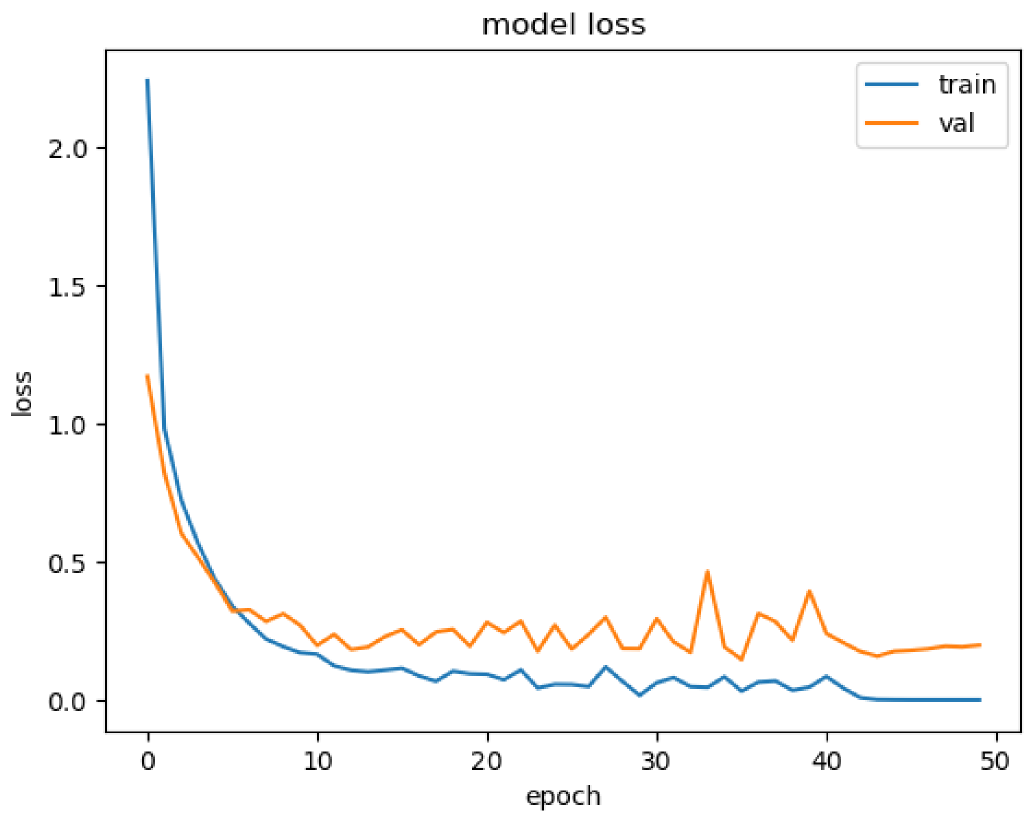
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```





In [35]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
```



In [36]:

```
results = model.evaluate(X_test , Y_test, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.17264
Test Accuracy: 97.62%
```

In [37]:

```
from sklearn.metrics import confusion_matrix , classification_report

y_true = list(Y_test)
y_pred = model.predict(X_test)
y_pred = list(map(lambda x: np.argmax(x), y_pred))
print('Y Actual Values :' , y_true[0:10])
print('Y Predicted Values :' , y_pred[0:10])
```

```
294/294 [==============================] - 3s 9ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

```
Y Predicted Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

In [38]:

```
classes = {2:'bkl', 4:'nv', 3:'df', 6:'mel', 5:'vasc', 1:'bcc', 0:'akiec'}

classes_labels=[]
for key in classes.keys():
    classes_labels.append(key)
print(classes_labels)
```

```
[2, 4, 3, 6, 5, 1, 0]
```

In [39]:

```
classes = {4: ('nv', ' melanocytic nevi'),
           6: ('mel', 'melanoma'),
           2 :('bkl', 'benign keratosis-like lesions'),
           1:('bcc' , ' basal cell carcinoma'),
           5: ('vasc', ' pyogenic granulomas and hemorrhage'),
           0: ('akiec', 'Actinic keratoses and intraepithelial carcinomae'),
           3: ('df', 'dermatofibroma')}
```

In [40]:

```
cm = confusion_matrix(y_true,y_pred,labels=classes_labels)
print(confusion_matrix(y_true,y_pred,labels=classes_labels))
```

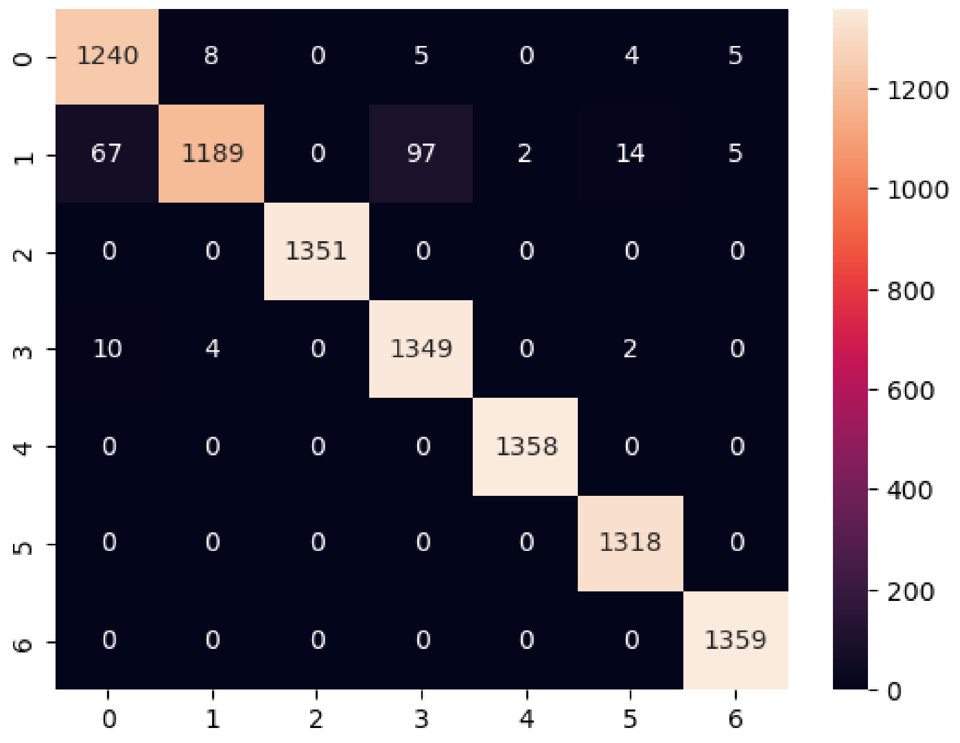
```
[[1240    8    0    5    0    4    5]
 [  67 1189    0   97    2   14    5]
 [   0    0 1351    0    0    0    0]
 [  10    4    0 1349    0    2    0]
 [   0    0    0    0 1358    0    0]
 [   0    0    0    0    0 1318    0]
 [   0    0    0    0    0    0 1359]]
```

In [41]:

```
sns.heatmap(cm, annot = True, fmt='')
```

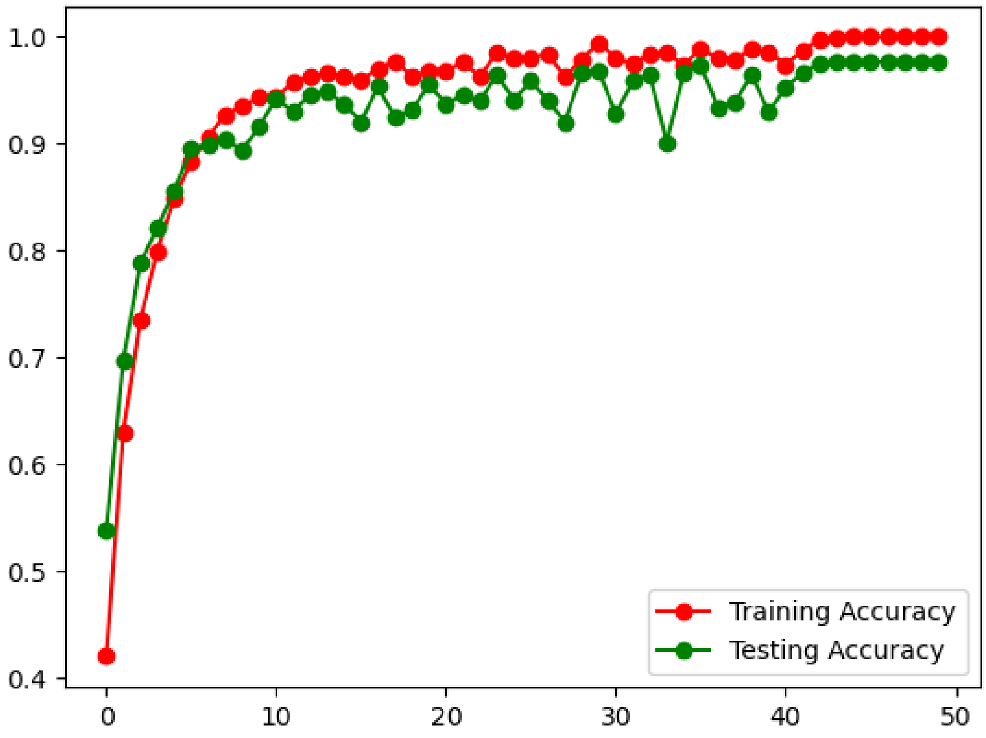
Out[41]:

```
<AxesSubplot:>
```



In [42]:

```
#training acc vs testing acc graph
plt.plot(history.history["accuracy"] , 'ro-' , label = "Training Accuracy")
plt.plot(history.history["val_accuracy"] , 'go-' , label = "Testing Accuracy")
plt.legend()
plt.show()
```



In [43]:

```
#predicting
y_pred  = model.predict(X_test).round()
```

```
294/294 [==============================] - 3s 9ms/step
```

In [44]:

```
target_names = [f"{classes[i]}" for i in range(7)]
print(len(Y_test) ,"  ",len(y_pred))
y_pred = list(map(lambda x: np.argmax(x), y_pred))
print(classification_report(Y_test , y_pred,target_names=target_names))
```

```
9387    9387
                                                               precision    recall  f1-
score   support

('akiec', 'Actinic keratoses and intraepithelial carcinomae')       0.99      1.00      1
.00      1359
                                    ('bcc', ' basal cell carcinoma')       0.99      1.00
0.99      1318
                            ('bkl', 'benign keratosis-like lesions')       0.94      0.98
0.96      1262
                                            ('df', 'dermatofibroma')       1.00      1.00
1.00      1351
                                      ('nv', ' melanocytic nevi')       0.99      0.86
0.92      1374
                     ('vasc', ' pyogenic granulomas and hemorrhage')       1.00      1.00
1.00      1358
                                                  ('mel', 'melanoma')       0.93      0.99
0.96      1365

                                                           accuracy
0.98      9387
                                                          macro avg       0.98      0.98
0.98      9387
                                                       weighted avg       0.98      0.98
0.98      9387
```

In [ ]:

```

```

NEW CNN Model 4-Layers

In [45]:

```
model_CNN = Sequential()
    model_CNN.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation
= 'relu', padding = 'same'))
    model_CNN.add(MaxPool2D(pool_size = (2,2)))

    model_CNN.add(Conv2D(32, kernel_size = (3,3), activation = 'relu', padding = 'same')
)
    model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))

    model_CNN.add(Conv2D(64, kernel_size = (3,3), activation = 'relu', padding = 'same')
)
    model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))
    model_CNN.add(Conv2D(128, kernel_size = (3,3), activation = 'relu', padding = 'same'
))
    model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))

    model_CNN.add(Flatten())
    model_CNN.add(Dense(64, activation = 'relu'))
    model_CNN.add(Dense(32, activation='relu'))
    model_CNN.add(Dense(7, activation='softmax'))

    optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)

    model_CNN.compile(loss = 'sparse_categorical_crossentropy',
                 optimizer = optimizer,
                  metrics = ['accuracy'])
    print(model_CNN.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 28, 28, 16)        448       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 14, 14, 16)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 14, 14, 32)        4640      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 32)         0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 7, 7, 64)          18496     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 4, 4, 64)         0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 4, 4, 128)         73856     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 2, 2, 128)        0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 512)               0         
                                                                 
 dense_2 (Dense)             (None, 64)                32832     
                                                                 
 dense_3 (Dense)             (None, 32)                2080      
                                                                 
 dense_4 (Dense)             (None, 7)                 231       
                                                                 
_________________________________________________________________
```

```
=================================================================
Total params: 132,583
Trainable params: 132,583
Non-trainable params: 0
_________________________________________________________________
None
```

In [46]:

```
history = model_CNN.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 128,
                    epochs = 50)
```

```
Epoch 1/50
235/235 [==============================] - 20s 77ms/step - loss: 2.2062 - accuracy: 0.393
1 - val_loss: 1.1830 - val_accuracy: 0.5507
Epoch 2/50
235/235 [==============================] - 17s 74ms/step - loss: 1.0033 - accuracy: 0.619
7 - val_loss: 0.8135 - val_accuracy: 0.7075
Epoch 3/50
235/235 [==============================] - 18s 76ms/step - loss: 0.7482 - accuracy: 0.719
5 - val_loss: 0.6418 - val_accuracy: 0.7638
Epoch 4/50
235/235 [==============================] - 17s 74ms/step - loss: 0.5512 - accuracy: 0.795
5 - val_loss: 0.4952 - val_accuracy: 0.8189
Epoch 5/50
235/235 [==============================] - 18s 76ms/step - loss: 0.4306 - accuracy: 0.841
5 - val_loss: 0.3839 - val_accuracy: 0.8631
Epoch 6/50
235/235 [==============================] - 18s 75ms/step - loss: 0.3420 - accuracy: 0.879
1 - val_loss: 0.3941 - val_accuracy: 0.8589
Epoch 7/50
235/235 [==============================] - 18s 76ms/step - loss: 0.2749 - accuracy: 0.904
4 - val_loss: 0.2474 - val_accuracy: 0.9152
Epoch 8/50
235/235 [==============================] - 18s 77ms/step - loss: 0.2116 - accuracy: 0.926
2 - val_loss: 0.2837 - val_accuracy: 0.8973
Epoch 9/50
235/235 [==============================] - 17s 74ms/step - loss: 0.1874 - accuracy: 0.936
3 - val_loss: 0.2229 - val_accuracy: 0.9225
Epoch 10/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1893 - accuracy: 0.934
7 - val_loss: 0.2190 - val_accuracy: 0.9314
Epoch 11/50
235/235 [==============================] - 18s 75ms/step - loss: 0.1351 - accuracy: 0.954
1 - val_loss: 0.1745 - val_accuracy: 0.9477
Epoch 12/50
235/235 [==============================] - 18s 77ms/step - loss: 0.1050 - accuracy: 0.963
8 - val_loss: 0.1714 - val_accuracy: 0.9470
Epoch 13/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1151 - accuracy: 0.960
0 - val_loss: 0.2353 - val_accuracy: 0.9270
Epoch 14/50
235/235 [==============================] - 18s 77ms/step - loss: 0.1065 - accuracy: 0.962
8 - val_loss: 0.1641 - val_accuracy: 0.9513
Epoch 15/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0958 - accuracy: 0.967
0 - val_loss: 0.2381 - val_accuracy: 0.9328
Epoch 16/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1240 - accuracy: 0.955
8 - val_loss: 0.1614 - val_accuracy: 0.9551
Epoch 17/50
235/235 [==============================] - 18s 75ms/step - loss: 0.1144 - accuracy: 0.960
5 - val_loss: 0.1763 - val_accuracy: 0.9482
Epoch 18/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0562 - accuracy: 0.981
4 - val_loss: 0.1244 - val_accuracy: 0.9636
Epoch 19/50
235/235 [==============================] - 18s 78ms/step - loss: 0.1079 - accuracy: 0.962
8 - val_loss: 0.1861 - val_accuracy: 0.9473
```

```
Epoch 20/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0743 - accuracy: 0.972
8 - val_loss: 0.1462 - val_accuracy: 0.9597
Epoch 21/50
235/235 [==============================] - 19s 80ms/step - loss: 0.0597 - accuracy: 0.978
8 - val_loss: 0.2509 - val_accuracy: 0.9249
Epoch 22/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0640 - accuracy: 0.978
1 - val_loss: 0.2125 - val_accuracy: 0.9419
Epoch 23/50
235/235 [==============================] - 18s 78ms/step - loss: 0.1135 - accuracy: 0.962
0 - val_loss: 0.1689 - val_accuracy: 0.9566
Epoch 24/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0488 - accuracy: 0.983
5 - val_loss: 0.1719 - val_accuracy: 0.9638
Epoch 25/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0399 - accuracy: 0.985
6 - val_loss: 0.2633 - val_accuracy: 0.9268
Epoch 26/50
235/235 [==============================] - 18s 78ms/step - loss: 0.1040 - accuracy: 0.966
0 - val_loss: 0.2146 - val_accuracy: 0.9505
Epoch 27/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0756 - accuracy: 0.974
6 - val_loss: 0.1280 - val_accuracy: 0.9694
Epoch 28/50
235/235 [==============================] - 19s 80ms/step - loss: 0.0166 - accuracy: 0.994
0 - val_loss: 0.1653 - val_accuracy: 0.9691
Epoch 29/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0732 - accuracy: 0.974
8 - val_loss: 0.1609 - val_accuracy: 0.9643
Epoch 30/50
235/235 [==============================] - 18s 78ms/step - loss: 0.0808 - accuracy: 0.973
6 - val_loss: 0.1951 - val_accuracy: 0.9461
Epoch 31/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0445 - accuracy: 0.984
8 - val_loss: 0.1676 - val_accuracy: 0.9650
Epoch 32/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0542 - accuracy: 0.982
2 - val_loss: 0.2234 - val_accuracy: 0.9368
Epoch 33/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0762 - accuracy: 0.973
9 - val_loss: 0.1967 - val_accuracy: 0.9587
Epoch 34/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0347 - accuracy: 0.988
3 - val_loss: 0.1550 - val_accuracy: 0.9646
Epoch 35/50
235/235 [==============================] - 19s 80ms/step - loss: 0.0214 - accuracy: 0.993
0 - val_loss: 0.1940 - val_accuracy: 0.9615
Epoch 36/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0650 - accuracy: 0.977
9 - val_loss: 0.2945 - val_accuracy: 0.9330
Epoch 37/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0881 - accuracy: 0.971
6 - val_loss: 0.2232 - val_accuracy: 0.9582
Epoch 38/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0211 - accuracy: 0.993
6 - val_loss: 0.1789 - val_accuracy: 0.9609
Epoch 39/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0355 - accuracy: 0.988
1 - val_loss: 0.1644 - val_accuracy: 0.9676
Epoch 40/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0271 - accuracy: 0.990
7 - val_loss: 0.2005 - val_accuracy: 0.9622
Epoch 41/50
235/235 [==============================] - 18s 74ms/step - loss: 0.0232 - accuracy: 0.992
5 - val_loss: 0.2582 - val_accuracy: 0.9523
Epoch 42/50
235/235 [==============================] - 18s 75ms/step - loss: 0.1141 - accuracy: 0.964
7 - val_loss: 0.1601 - val_accuracy: 0.9615
Epoch 43/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0273 - accuracy: 0.990
8 - val_loss: 0.1969 - val_accuracy: 0.9628
Epoch 44/50
Epoch 44/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0158 - accuracy: 0.995
1 - val_loss: 0.1818 - val_accuracy: 0.9714
Epoch 45/50
235/235 [==============================] - 17s 73ms/step - loss: 0.0133 - accuracy: 0.995
3 - val_loss: 0.1677 - val_accuracy: 0.9684
Epoch 46/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0994 - accuracy: 0.969
1 - val_loss: 0.1632 - val_accuracy: 0.9654
Epoch 47/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0232 - accuracy: 0.992
0 - val_loss: 0.1358 - val_accuracy: 0.9760
Epoch 48/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0120 - accuracy: 0.996
0 - val_loss: 0.1974 - val_accuracy: 0.9625
Epoch 49/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0500 - accuracy: 0.983
8 - val_loss: 0.2534 - val_accuracy: 0.9525
Epoch 50/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0667 - accuracy: 0.979
1 - val_loss: 0.1998 - val_accuracy: 0.9625
```

In [47]:

```python
print(model_CNN.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 28, 28, 16)        448       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 14, 14, 16)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 14, 14, 32)        4640      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 32)         0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 7, 7, 64)          18496     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 4, 4, 64)         0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 4, 4, 128)         73856     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 2, 2, 128)        0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 512)               0         
                                                                 
 dense_2 (Dense)             (None, 64)                32832     
                                                                 
 dense_3 (Dense)             (None, 32)                2080      
                                                                 
 dense_4 (Dense)             (None, 7)                 231       
                                                                 
=================================================================
Total params: 132,583
Trainable params: 132,583
Non-trainable params: 0
_________________________________________________________________
None
```

In [48]:

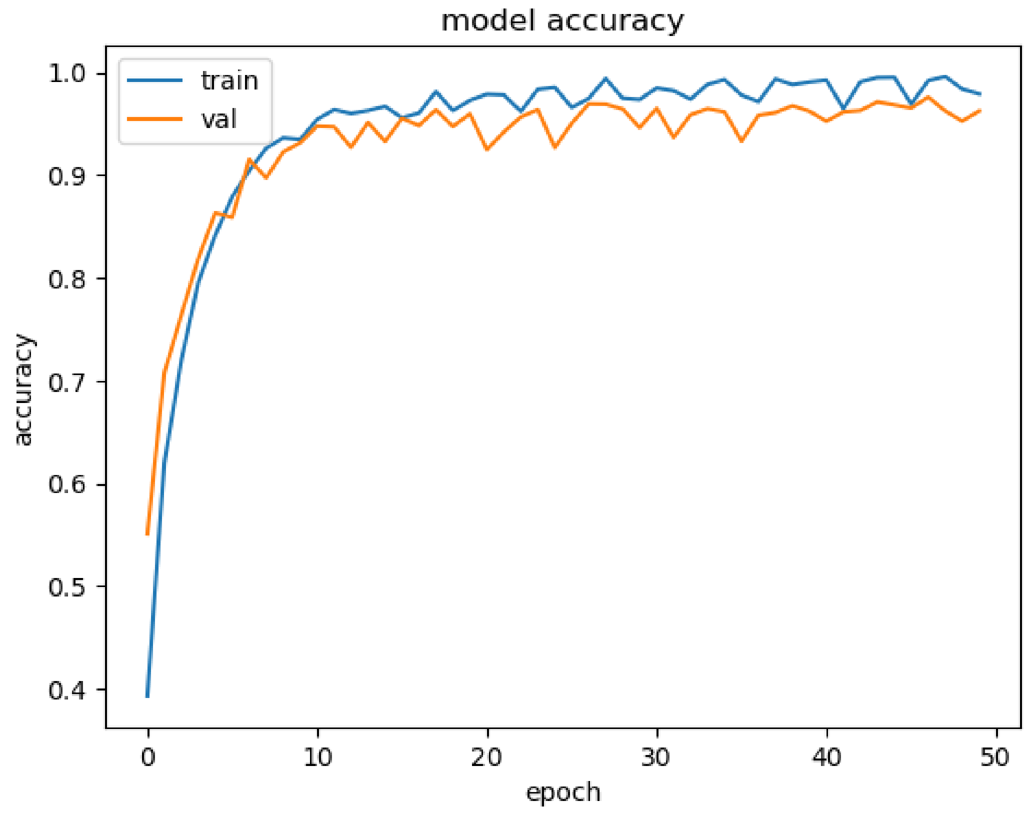
```python
results = model_CNN.evaluate(X_test , Y_test, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.19332
Test Accuracy: 96.48%
```

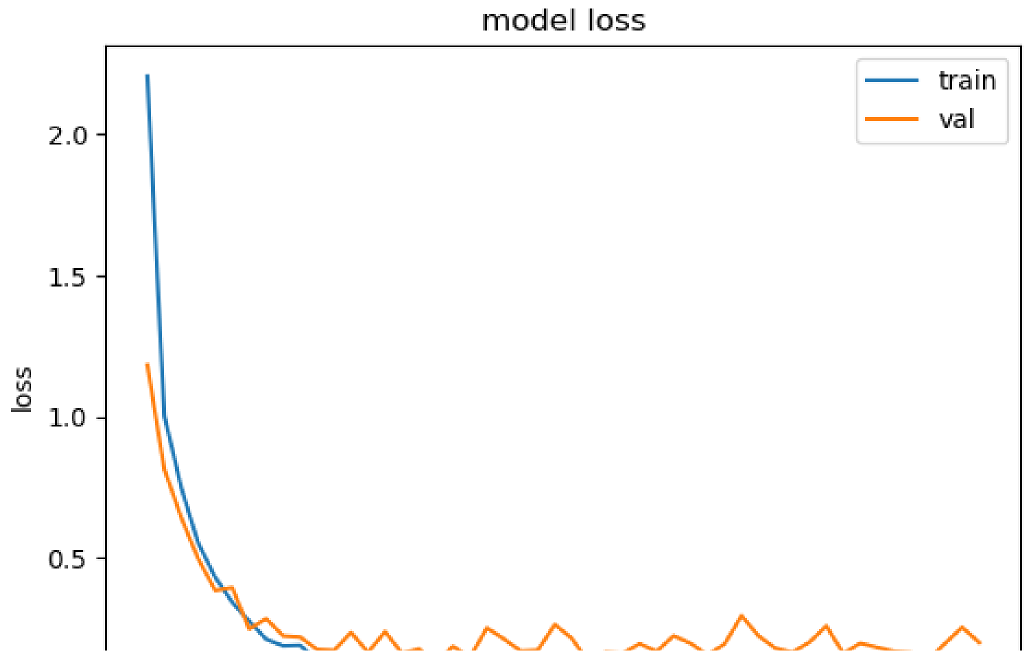
In [49]:

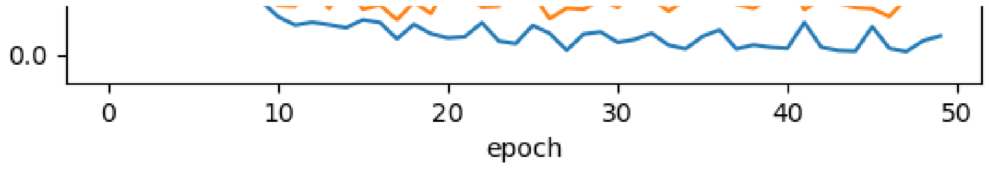
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [50]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
```





In [51]:

```
from sklearn.metrics import confusion_matrix , classification_report

y_true_CNN = list(Y_test)
y_pred_CNN = model_CNN.predict(X_test)
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print('Y Actual Values :' , y_true_CNN[0:10])
print('Y Predicted Values :' , y_pred[0:10])
```

```
294/294 [==============================] - 3s 9ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
Y Predicted Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

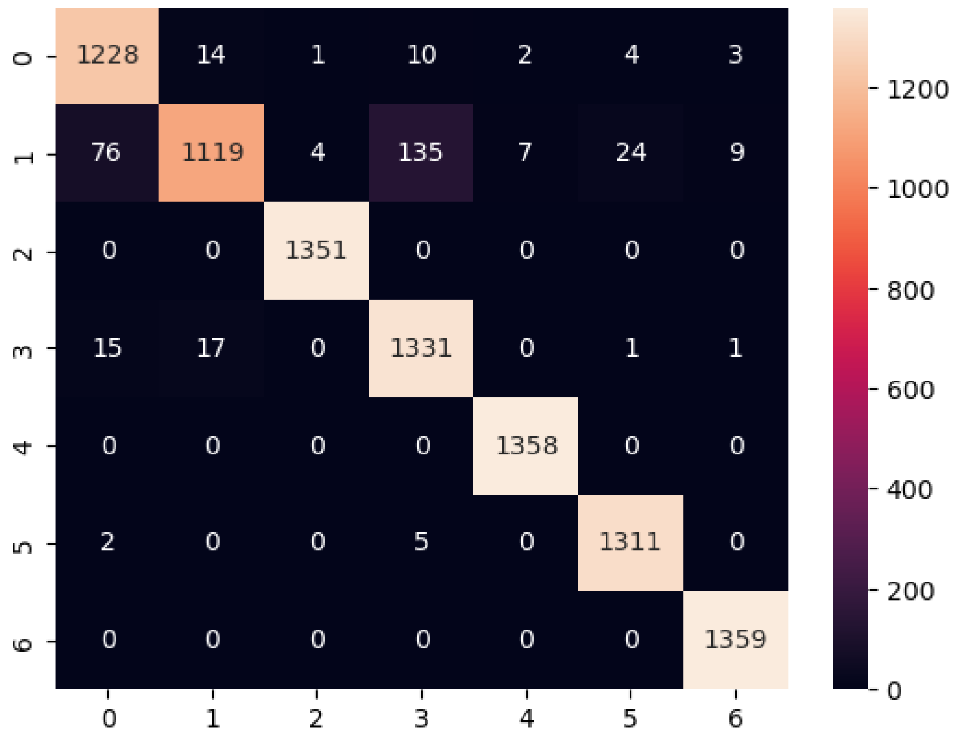
In [52]:

```
cm_CNN = confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels)
print(confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels))
sns.heatmap(cm_CNN, annot = True, fmt='')
```

```
[[1228   14    1   10    2    4    3]
 [  76 1119    4  135    7   24    9]
 [   0    0 1351    0    0    0    0]
 [  15   17    0 1331    0    1    1]
 [   0    0    0    0 1358    0    0]
 [   2    0    0    5    0 1311    0]
 [   0    0    0    0    0    0 1359]]
```

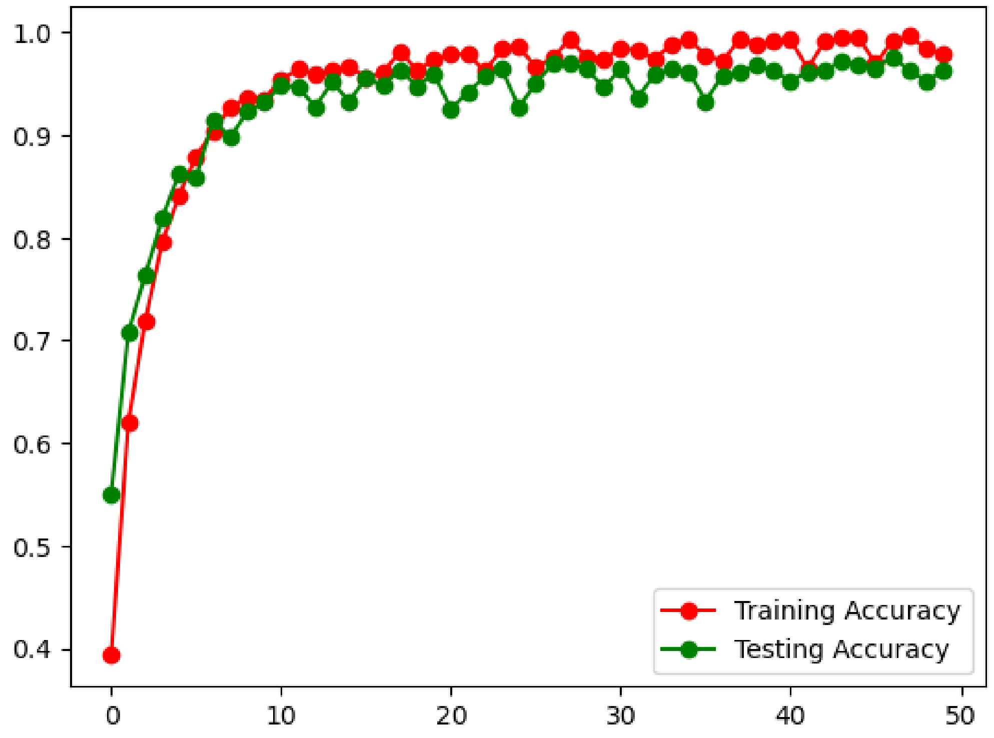
Out[52]:

```
<AxesSubplot:>
```



In [53]:

```
#training acc vs testing acc graph
plt.plot(history.history["accuracy"] , 'ro-' , label = "Training Accuracy")
plt.plot(history.history["val_accuracy"] , 'go-' , label = "Testing Accuracy")
plt.legend()
plt.show()
```



In [54]:

```
#predicting
y_pred_CNN  = model.predict(X_test).round()
target_names = [f"{classes[i]}" for i in range(7)]
print(len(Y_test) ,"   ",len(y_pred_CNN))
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print(classification_report(Y_test , y_pred_CNN,target_names=target_names))
```

```
294/294 [==============================] - 3s 9ms/step
9387    9387
                                                               precision    recall  f1-
score   support

('akiec', 'Actinic keratoses and intraepithelial carcinomae')       0.99      1.00      1
.00      1359
                              ('bcc', ' basal cell carcinoma')       0.99      1.00
0.99      1318
                      ('bkl', 'benign keratosis-like lesions')       0.94      0.98
0.96      1262
                                    ('df', 'dermatofibroma')       1.00      1.00
1.00      1351
                                 ('nv', ' melanocytic nevi')       0.99      0.86
0.92      1374
                ('vasc', ' pyogenic granulomas and hemorrhage')       1.00      1.00
1.00      1358
                                         ('mel', 'melanoma')       0.93      0.99
0.96      1365

                                                    accuracy
0.98      9387
                                                   macro avg       0.98      0.98
0.98      9387
                                                weighted avg       0.98      0.98
0.98      9387
```

NEW CNN Model 4-Layers with Early Stopping & Reduce Learning Rate

In [65]:

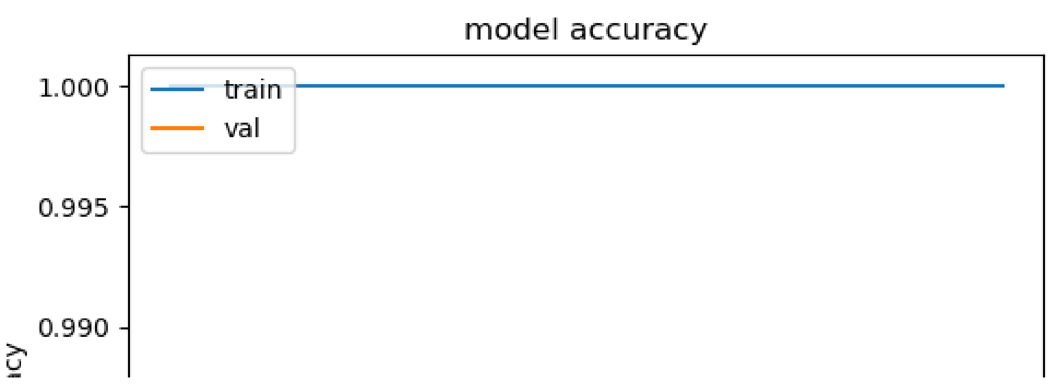
```
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, mode='auto')
```

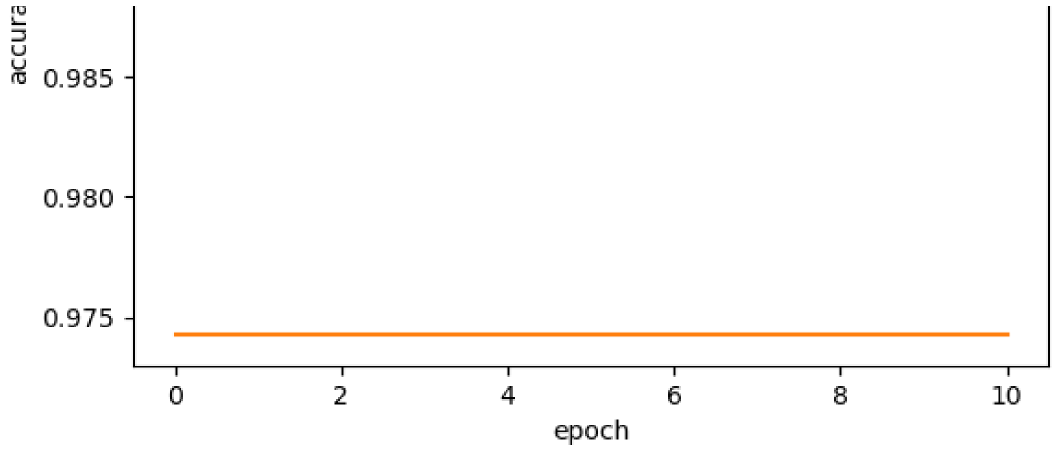
```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=3, verbose=1, mod
e='auto')
history_1 = model_CNN.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 64,
                    epochs = 50,
                    callbacks = [reduce_lr, early_stop])
```

```
Epoch 1/50
470/470 [==============================] - 20s 43ms/step - loss: 5.8255e-04 - accuracy: 1
.0000 - val_loss: 0.1584 - val_accuracy: 0.9743 - lr: 1.0000e-07
Epoch 2/50
470/470 [==============================] - 19s 41ms/step - loss: 5.8166e-04 - accuracy: 1
.0000 - val_loss: 0.1584 - val_accuracy: 0.9743 - lr: 1.0000e-07
Epoch 3/50
470/470 [==============================] - 19s 41ms/step - loss: 5.8055e-04 - accuracy: 1
.0000 - val_loss: 0.1584 - val_accuracy: 0.9743 - lr: 1.0000e-07
Epoch 4/50
470/470 [==============================] - ETA: 0s - loss: 5.7919e-04 - accuracy: 1.0000
Epoch 4: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
470/470 [==============================] - 19s 41ms/step - loss: 5.7919e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-07
Epoch 5/50
470/470 [==============================] - 20s 42ms/step - loss: 5.7791e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-08
Epoch 6/50
470/470 [==============================] - 20s 43ms/step - loss: 5.7783e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-08
Epoch 7/50
469/470 [============================>.] - ETA: 0s - loss: 5.7776e-04 - accuracy: 1.0000
Epoch 7: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
470/470 [==============================] - 20s 42ms/step - loss: 5.7773e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-08
Epoch 8/50
470/470 [==============================] - 20s 42ms/step - loss: 5.7763e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-09
Epoch 9/50
470/470 [==============================] - 20s 42ms/step - loss: 5.7763e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-09
Epoch 10/50
469/470 [============================>.] - ETA: 0s - loss: 5.7674e-04 - accuracy: 1.0000
Epoch 10: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-10.
470/470 [==============================] - 20s 42ms/step - loss: 5.7763e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-09
Epoch 11/50
470/470 [==============================] - 20s 42ms/step - loss: 5.7763e-04 - accuracy: 1
.0000 - val_loss: 0.1585 - val_accuracy: 0.9743 - lr: 1.0000e-10
Epoch 11: early stopping
```

In [66]:

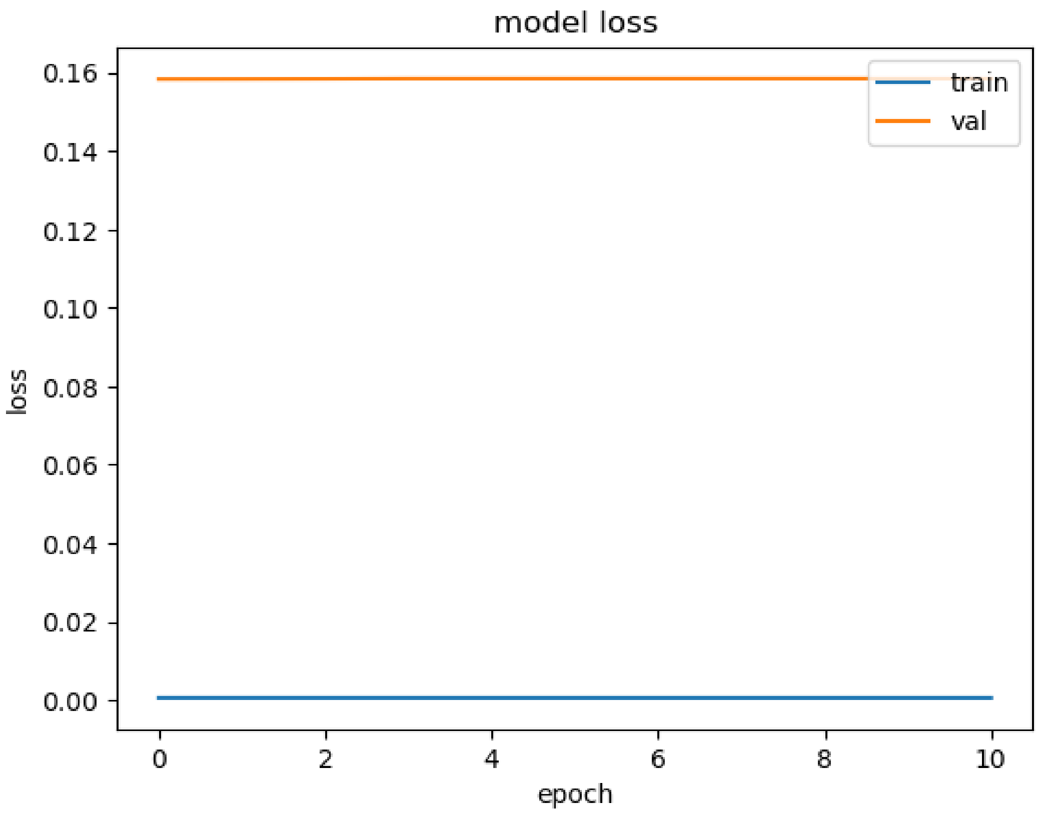
```
plt.plot(history_1.history['accuracy'])
plt.plot(history_1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```





In [67]:

```
plt.plot(history_1.history['loss'])
plt.plot(history_1.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
```



In [68]:

```
from sklearn.metrics import confusion_matrix , classification_report

y_true_CNN = list(Y_test)
y_pred_CNN = model_CNN.predict(X_test)
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print('Y Actual Values :' , y_true_CNN[0:10])
print('Y Predicted Values :' , y_pred[0:10])
```

```
294/294 [==============================] - 3s 9ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
Y Predicted Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

In [69]:

```
results_1 = model_CNN.evaluate(X_test , Y_test, verbose=0)
```

```
print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.19332
Test Accuracy: 96.48%
```

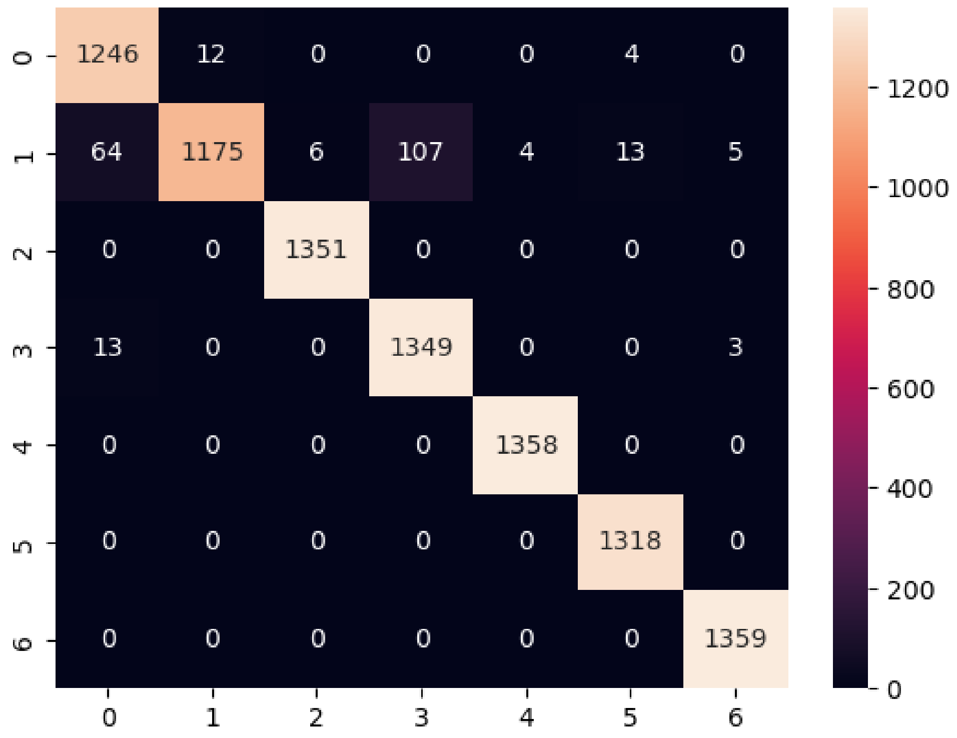
In [60]:

```
cm_CNN_lr = confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels)
print(confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels))
sns.heatmap(cm_CNN_lr, annot = True, fmt='')
```

```
[[1246   12    0    0    0    4    0]
 [  64 1175    6  107    4   13    5]
 [   0    0 1351    0    0    0    0]
 [  13    0    0 1349    0    0    3]
 [   0    0    0    0 1358    0    0]
 [   0    0    0    0    0 1318    0]
 [   0    0    0    0    0    0 1359]]
```

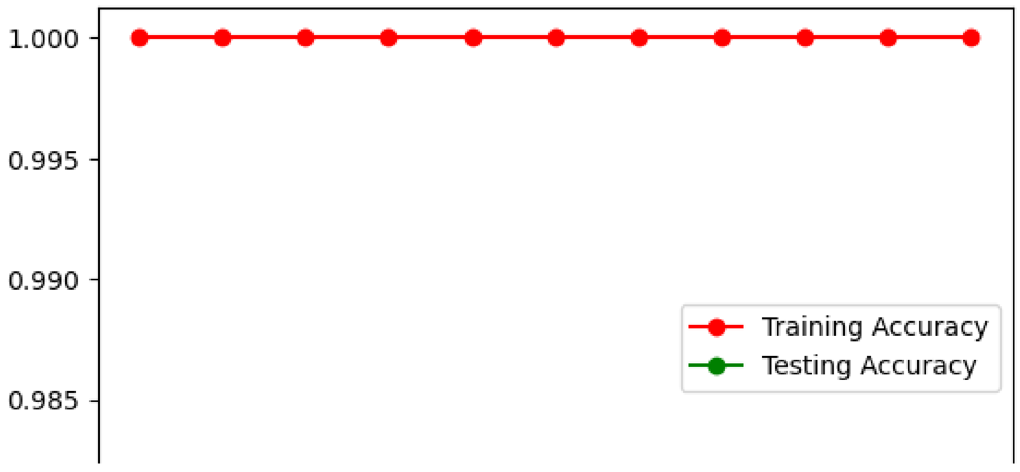
Out[60]:

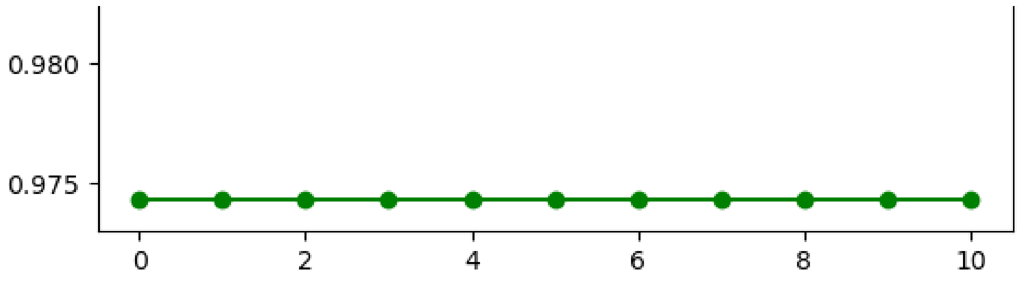
```
<AxesSubplot:>
```



In [70]:

```
#training acc vs testing acc graph
plt.plot(history_1.history["accuracy"] , 'ro-' , label = "Training Accuracy")
plt.plot(history_1.history["val_accuracy"] , 'go-' , label = "Testing Accuracy")
plt.legend()
plt.show()
```





In [71]:

```
y_pred_L  = model_CNN.predict(X_test).round()
```

```
294/294 [==============================] - 3s 9ms/step
```

In [72]:

```
#predicting
y_pred_CNN_L  = model.predict(X_test).round()
target_names = [f"{classes[i]}" for i in range(7)]
print(len(Y_test) ,"  ",len(y_pred_CNN_L))
y_pred_CNN_L = list(map(lambda x: np.argmax(x), y_pred_CNN_L))
print(classification_report(Y_test , y_pred_CNN_L,target_names=target_names))
```

```
294/294 [==============================] - 3s 10ms/step
9387    9387
                                                              precision    recall  f1-
score   support

('akiec', 'Actinic keratoses and intraepithelial carcinomae')       0.99      1.00      1
.00      1359
                              ('bcc', ' basal cell carcinoma')       0.99      1.00
0.99      1318
                      ('bkl', 'benign keratosis-like lesions')       0.94      0.98
0.96      1262
                                ('df', 'dermatofibroma')       1.00      1.00
1.00      1351
                             ('nv', ' melanocytic nevi')       0.99      0.86
0.92      1374
                ('vasc', ' pyogenic granulomas and hemorrhage')       1.00      1.00
1.00      1358
                                     ('mel', 'melanoma')       0.93      0.99
0.96      1365

                                                  accuracy
0.98      9387
                                                 macro avg       0.98      0.98
0.98      9387
                                              weighted avg       0.98      0.98
0.98      9387
```

In [73]:

```
# Layers definitions
from keras import backend as K
for l in range(len(model_CNN.layers)):
    print(l, model_CNN.layers[l])
```

```
0 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3761f80b50>
1 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f37620e40d0>
2 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3761f33150>
3 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f376df1d190>
4 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f3761fbd690>
5 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f376088d510>
6 <keras.layers.convolutional.conv2d.Conv2D object at 0x7f376db8b710>
7 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f37620b4290>
8 <keras.layers.reshaping.flatten.Flatten object at 0x7f37608c3410>
9 <keras.layers.core.dense.Dense object at 0x7f37608c3f90>
10 <keras.layers.core.dense.Dense object at 0x7f376db8bc10>
11 <keras.layers.core.dense.Dense object at 0x7f3760888090>
```

[illegible]

In [ ]:

```
```