In [23]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [24]:
```
data = pd.read_csv("/kaggle/input/dataset/emotions.csv")
```

In [25]:
```
data
```

Out[25]:

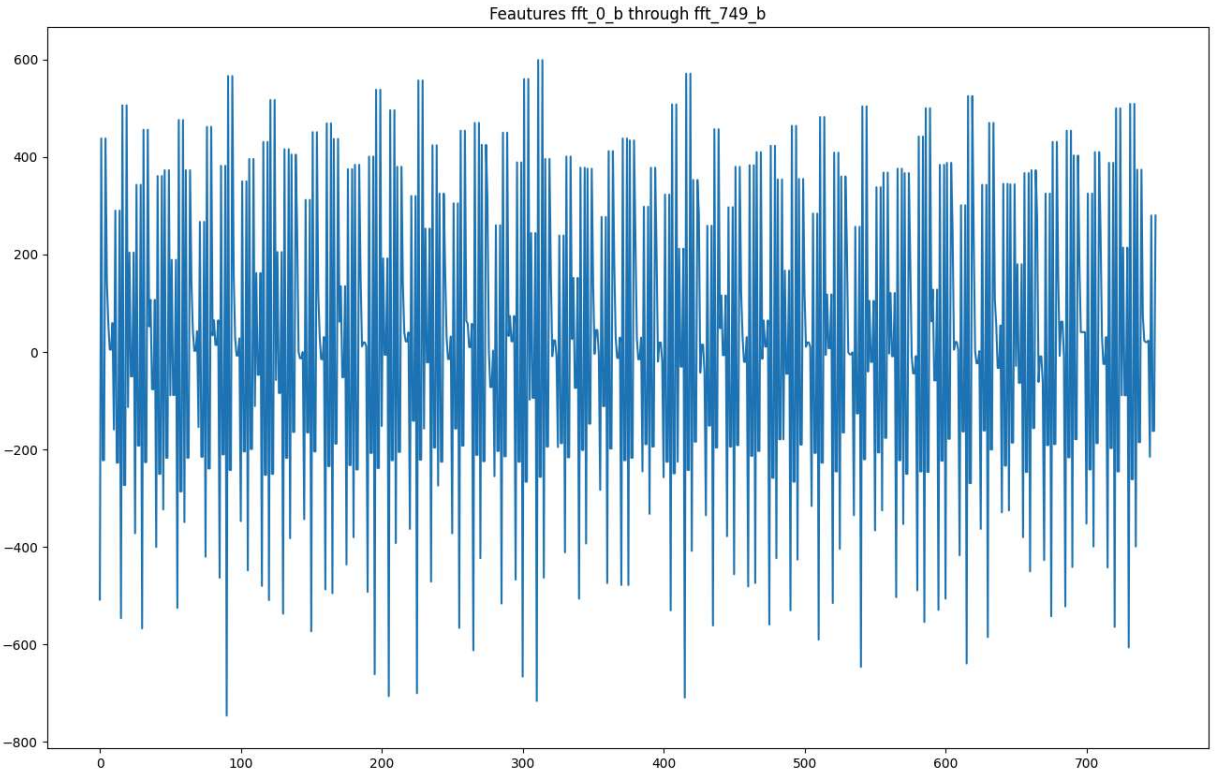
	# mean_0_a	mean_1_a	mean_2_a	mean_3_a	mean_4_a	mean_d_0_a	mean_d_1_a	me
0	4.620	30.3	-356.0	15.60	26.3	1.070	0.411	
1	28.800	33.1	32.0	25.80	22.8	6.550	1.680	
2	8.900	29.4	-416.0	16.70	23.7	79.900	3.360	
3	14.900	31.6	-143.0	19.80	24.3	-0.584	-0.284	
4	28.300	31.3	45.2	27.30	24.5	34.800	-5.790	
...	...	...	...	...	...	...	...	
2127	32.400	32.2	32.2	30.80	23.4	1.640	-2.030	
2128	16.300	31.3	-284.0	14.30	23.9	4.200	1.090	
2129	-0.547	28.3	-259.0	15.80	26.7	9.080	6.900	
2130	16.800	19.9	-288.0	8.34	26.0	2.460	1.580	
2131	27.000	32.0	31.8	25.00	28.9	4.990	1.950	

2132 rows × 2549 columns

In [26]:
```
sample = data.loc[0, 'fft_0_b':'fft_749_b']
sample
```

Out[26]:
```
fft_0_b     -508.0
fft_1_b      438.0
fft_2_b     -222.0
fft_3_b     -222.0
fft_4_b      438.0
             ...
fft_745_b   -215.0
fft_746_b    280.0
fft_747_b   -162.0
fft_748_b   -162.0
fft_749_b    280.0
Name: 0, Length: 750, dtype: object
```

In [27]:
```
plt.figure(figsize=(16, 10))
plt.plot(range(len(sample)), sample)
plt.title('Feautures fft_0_b through fft_749_b')
plt.show()
```

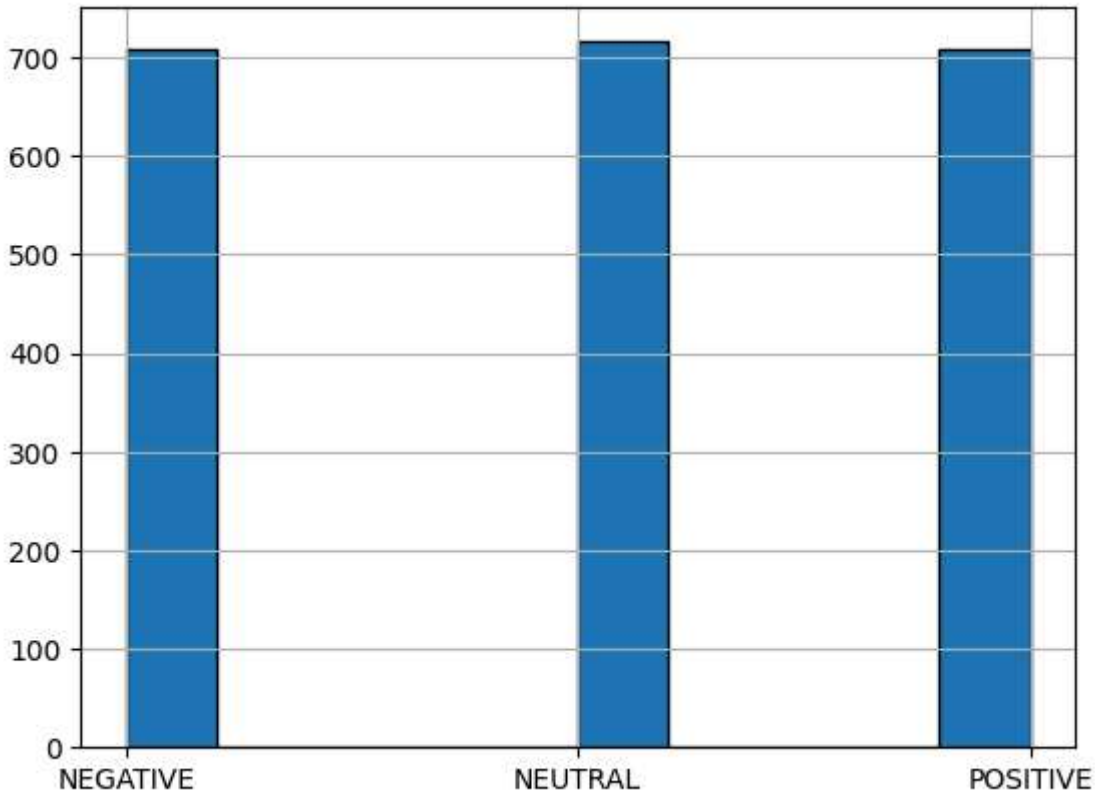


In [28]:
```
data['label'].value_counts()
```

Out[28]:
```
label
NEUTRAL     716
NEGATIVE    708
POSITIVE    708
Name: count, dtype: int64
```

In [29]:
```
data['label'].hist(edgecolor='black')
```

Out[29]:
```
<Axes: >
```



In [30]:
```
label_mapping = {'NEGATIVE': 0, 'NEUTRAL': 1, 'POSITIVE': 2}
```

In [31]:
```
original_data = data.copy()
data["label"] = data["label"].replace(label_mapping)

y = data["label"]
x = data.drop("label", axis=1)

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_st
```

```
/tmp/ipykernel_34/1063426523.py:2: FutureWarning: Downcasting behavior in `replace`
is deprecated and will be removed in a future version. To retain the old behavior, e
xplicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  data["label"] = data["label"].replace(label_mapping)
```

In [32]:
```
x_train
```

Out[32]:

	# mean_0_a	mean_1_a	mean_2_a	mean_3_a	mean_4_a	mean_d_0_a	mean_d_1_a	me
480	-2.450	24.6	2.16	-10.400	18.9	-2.650	-0.101	
1394	6.340	16.9	22.40	0.164	23.1	-1.590	-0.530	
1204	12.900	30.5	-269.00	13.100	23.8	1.120	-1.890	
1654	-12.400	26.6	-870.00	8.590	24.8	7.880	-1.530	
1010	15.400	26.6	-137.00	14.900	29.4	-0.609	0.198	
...	...	...	...	...	...	...	...	
1638	13.900	17.5	-327.00	3.810	27.1	-6.400	-5.460	
1095	31.300	31.0	30.70	28.900	27.5	2.800	0.253	
1130	0.959	24.5	7.31	-12.600	21.2	-4.170	0.307	
1294	26.700	29.3	22.70	26.000	25.9	8.500	1.680	
860	13.300	31.8	-147.00	11.300	29.1	-0.723	2.160	

1705 rows × 2548 columns

In [33]:
```
y_train
```

Out[33]:
```
480     2
1394    2
1204    2
1654    0
1010    0
       ..
1638    0
1095    1
1130    2
1294    1
860     0
Name: label, Length: 1705, dtype: int64
```

In [34]:
```
# Building the RNN

from keras.models import Sequential
from keras.layers import GRU
from keras.layers import Dropout
from keras.layers import Dense

model = Sequential()

model.add(tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=2), input_shape=(

model.add(GRU(units=256,return_sequences=True))
```

```
model.add(tf.keras.layers.Flatten())

model.add(Dense(units=3, activation = "softmax"))

model.summary()
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/core/lambda_layer.py:66: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 2548, 1)	0
gru_1 (GRU)	(None, 2548, 256)	198,912
flatten_1 (Flatten)	(None, 652288)	0
dense_1 (Dense)	(None, 3)	1,956,867

Total params: 2,155,779 (8.22 MB)

Trainable params: 2,155,779 (8.22 MB)

Non-trainable params: 0 (0.00 B)

In [35]:
```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

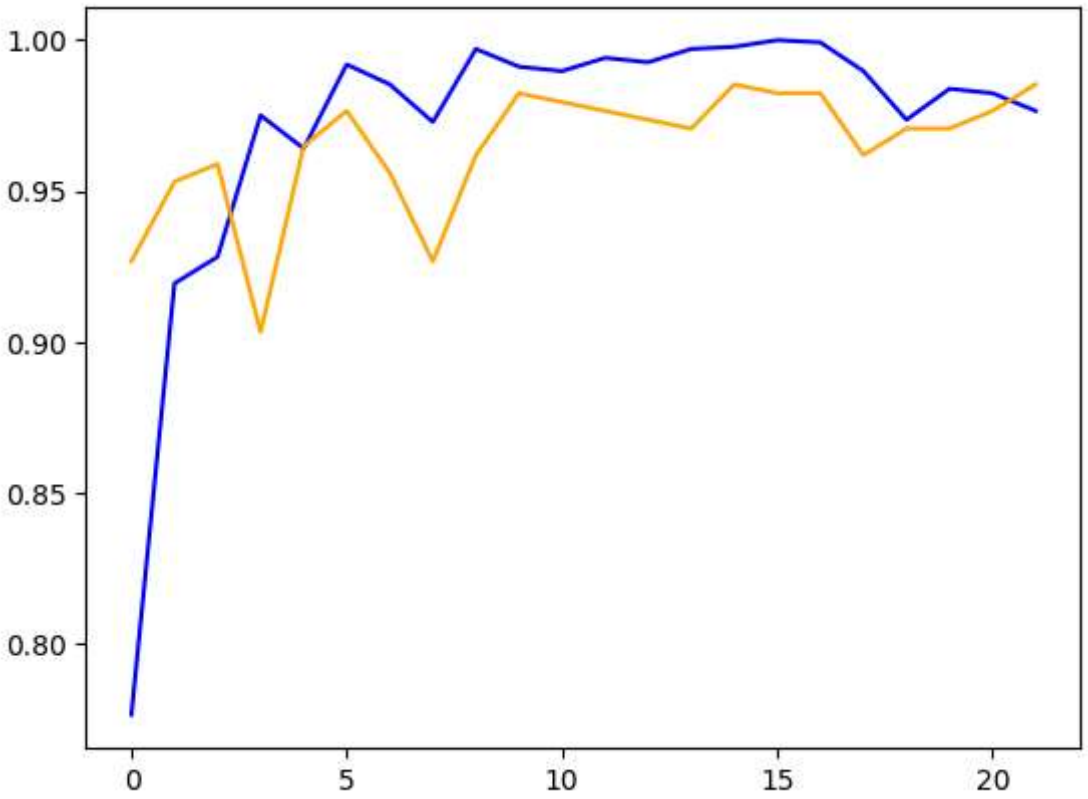
In [36]:
```
history = model.fit(
    x_train,
    y_train,
    validation_split=0.2,
    batch_size=32,
    epochs=50,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_accuracy',
            patience=7,
            restore_best_weights=True
        )
    ]
)
```

```
Epoch 1/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 115ms/step - accuracy: 0.6527 - loss: 52.4166 - val_ac
curacy: 0.9267 - val_loss: 5.6744
Epoch 2/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9284 - loss: 3.6426 - val_acc
uracy: 0.9531 - val_loss: 1.6354
Epoch 3/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9342 - loss: 4.5815 - val_acc
uracy: 0.9589 - val_loss: 1.8130
Epoch 4/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 107ms/step - accuracy: 0.9746 - loss: 0.7407 - val_acc
uracy: 0.9032 - val_loss: 4.0039
Epoch 5/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 108ms/step - accuracy: 0.9501 - loss: 2.2150 - val_acc
uracy: 0.9648 - val_loss: 1.3982
Epoch 6/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 108ms/step - accuracy: 0.9945 - loss: 0.0838 - val_acc
uracy: 0.9765 - val_loss: 0.9080
Epoch 7/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 108ms/step - accuracy: 0.9891 - loss: 0.4023 - val_acc
uracy: 0.9560 - val_loss: 3.1080
Epoch 8/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 108ms/step - accuracy: 0.9751 - loss: 0.9818 - val_acc
uracy: 0.9267 - val_loss: 5.9890
Epoch 9/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 107ms/step - accuracy: 0.9922 - loss: 0.0790 - val_acc
uracy: 0.9619 - val_loss: 1.4369
Epoch 10/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9947 - loss: 0.1679 - val_acc
uracy: 0.9824 - val_loss: 1.1030
Epoch 11/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9880 - loss: 0.5290 - val_acc
uracy: 0.9795 - val_loss: 0.6348
Epoch 12/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9937 - loss: 0.3274 - val_acc
uracy: 0.9765 - val_loss: 1.6696
Epoch 13/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9941 - loss: 0.1160 - val_acc
uracy: 0.9736 - val_loss: 1.9612
Epoch 14/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 105ms/step - accuracy: 0.9967 - loss: 0.1768 - val_acc
uracy: 0.9707 - val_loss: 1.3454
Epoch 15/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 105ms/step - accuracy: 0.9953 - loss: 0.1594 - val_acc
uracy: 0.9853 - val_loss: 0.7976
Epoch 16/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 4s 104ms/step - accuracy: 1.0000 - loss: 2.8729e-04 - val
_accuracy: 0.9824 - val_loss: 0.9588
Epoch 17/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 105ms/step - accuracy: 0.9999 - loss: 1.2541e-04 - val
_accuracy: 0.9824 - val_loss: 1.0271
Epoch 18/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 105ms/step - accuracy: 0.9943 - loss: 0.4002 - val_acc
uracy: 0.9619 - val_loss: 3.0729
Epoch 19/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9749 - loss: 2.0552 - val_acc
```

```
uracy: 0.9707 - val_loss: 3.3682
Epoch 20/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 105ms/step - accuracy: 0.9875 - loss: 0.7821 - val_acc
uracy: 0.9707 - val_loss: 1.7795
Epoch 21/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9853 - loss: 0.7548 - val_acc
uracy: 0.9765 - val_loss: 1.3305
Epoch 22/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 106ms/step - accuracy: 0.9705 - loss: 2.3424 - val_acc
uracy: 0.9853 - val_loss: 1.4657
```

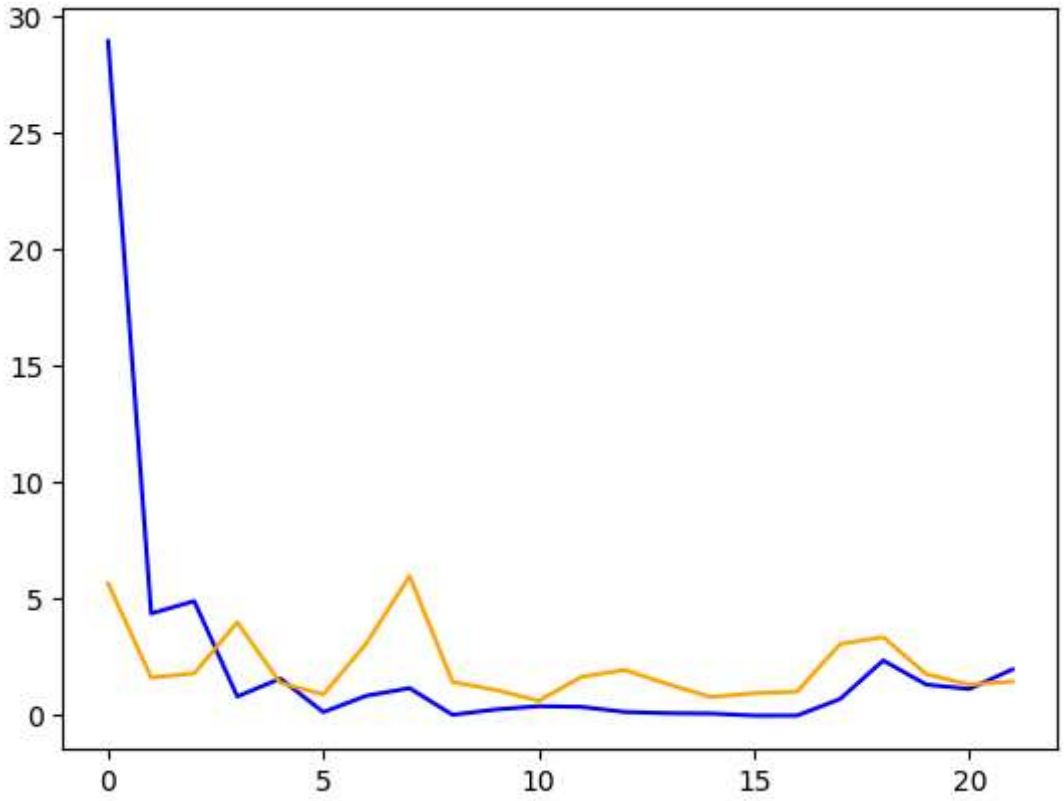
In [37]:
```
plt.plot(history.history["accuracy"], color = "blue")
plt.plot(history.history["val_accuracy"],  color = "orange")
```

Out[37]:
```
[<matplotlib.lines.Line2D at 0x7ba1e46328c0>]
```



In [38]:
```
plt.plot(history.history["loss"], color = "blue")
plt.plot(history.history["val_loss"],  color = "orange")
```

Out[38]:
```
[<matplotlib.lines.Line2D at 0x7ba1e4547850>]
```



In [39]:
```
model_acc = model.evaluate(x_test, y_test, verbose=0)[1]
print("Test Accuracy: {:.2f}%".format(model_acc * 100))
```

```
Test Accuracy: 98.13%
```

In [40]:
```
y_pred = model.predict(x_test)
y_pred = np.argmax(y_pred, axis = 1)
y_pred
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step
```

Out[40]:
```
array([1, 1, 0, 0, 2, 1, 0, 2, 2, 2, 0, 0, 0, 2, 1, 1, 0, 0, 2, 2, 1, 1,
       0, 0, 1, 2, 0, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 2, 0, 1, 2, 1,
       2, 1, 1, 2, 2, 0, 0, 0, 1, 0, 2, 0, 0, 1, 1, 1, 0, 1, 0, 1, 2, 0,
       0, 0, 1, 2, 2, 1, 2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 2, 0, 0, 0, 2, 1, 0, 1, 0, 1, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2,
       0, 2, 2, 1, 1, 0, 0, 2, 0, 2, 2, 2, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0,
       2, 1, 0, 0, 2, 2, 0, 2, 1, 0, 0, 2, 2, 1, 0, 1, 2, 1, 2, 1, 2, 0,
       2, 1, 2, 2, 1, 1, 2, 0, 1, 2, 2, 0, 2, 1, 1, 1, 0, 1, 2, 1, 0, 0,
       2, 1, 2, 1, 1, 1, 2, 2, 0, 1, 0, 1, 0, 1, 1, 2, 2, 0, 0, 1, 0, 1,
       2, 0, 1, 2, 0, 1, 2, 0, 0, 2, 2, 1, 2, 1, 1, 0, 0, 1, 1, 0, 2, 0,
       0, 2, 0, 1, 1, 2, 0, 1, 2, 1, 2, 1, 2, 2, 0, 0, 0, 0, 2, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 2, 2, 0, 1, 0, 0, 0, 0, 0, 2, 2, 2, 1, 1, 0, 1,
       2, 2, 2, 0, 0, 1, 2, 0, 0, 1, 1, 2, 0, 1, 1, 0, 1, 2, 0, 1, 2, 2,
       2, 0, 0, 2, 1, 0, 2, 1, 1, 2, 2, 2, 2, 1, 0, 2, 0, 1, 0, 1, 2, 2,
       0, 1, 2, 1, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 1, 1,
       1, 0, 1, 2, 2, 0, 2, 1, 0, 2, 0, 1, 2, 2, 0, 0, 2, 0, 0, 1, 2, 2,
       2, 0, 1, 0, 0, 1, 2, 0, 2, 1, 1, 2, 2, 1, 0, 2, 2, 2, 2, 2, 1, 0,
       1, 2, 1, 1, 0, 0, 1, 0, 1, 2, 1, 2, 1, 2, 0, 0, 1, 2, 2, 0, 1, 0,
       0, 2, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 2, 0, 2, 1, 2, 1, 1, 1, 0,
       2, 1, 0, 2, 1, 1, 1, 0, 2])
```

In [41]:
```
y_test = np.array(y_test)
y_test
```

Out[41]:
```
array([1, 1, 0, 0, 2, 1, 0, 2, 2, 0, 0, 0, 0, 2, 1, 1, 0, 0, 2, 2, 1, 1,
       0, 0, 1, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 2, 0, 1, 2, 1,
       2, 1, 1, 2, 2, 0, 0, 0, 1, 0, 2, 0, 0, 1, 1, 1, 0, 1, 0, 1, 2, 0,
       0, 0, 1, 2, 2, 1, 2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 2, 0, 0, 0, 2, 1, 0, 1, 0, 1, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2,
       0, 2, 2, 1, 1, 0, 0, 2, 0, 2, 2, 2, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0,
       2, 1, 0, 0, 2, 2, 0, 2, 1, 0, 0, 2, 0, 1, 0, 1, 2, 1, 2, 1, 2, 0,
       2, 1, 2, 2, 1, 1, 2, 0, 1, 2, 2, 2, 2, 1, 1, 1, 0, 1, 2, 1, 0, 0,
       2, 1, 2, 1, 1, 1, 2, 2, 0, 1, 0, 1, 0, 1, 1, 2, 2, 0, 0, 1, 0, 1,
       1, 0, 1, 2, 0, 1, 2, 0, 0, 2, 2, 1, 2, 1, 1, 0, 0, 1, 1, 0, 2, 0,
       0, 2, 0, 1, 1, 2, 0, 1, 2, 1, 2, 1, 2, 2, 0, 0, 0, 0, 2, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 2, 2, 0, 1, 0, 0, 0, 0, 0, 2, 2, 2, 1, 1, 0, 1,
       2, 2, 2, 0, 0, 1, 2, 0, 0, 1, 1, 2, 0, 1, 1, 0, 1, 2, 0, 1, 0, 2,
       2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 2, 2, 2, 1, 0, 2, 0, 1, 0, 1, 2, 2,
       0, 1, 2, 1, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 1, 2,
       1, 0, 1, 2, 2, 0, 2, 1, 0, 2, 0, 1, 2, 2, 0, 0, 2, 0, 0, 1, 2, 2,
       2, 0, 1, 0, 0, 1, 2, 0, 2, 1, 1, 2, 2, 1, 0, 2, 2, 2, 2, 2, 1, 0,
       1, 2, 1, 1, 0, 0, 1, 0, 1, 2, 1, 2, 1, 2, 0, 0, 1, 2, 2, 0, 1, 0,
       0, 2, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 2, 0, 2, 1, 2, 1, 1, 1, 0,
       2, 1, 0, 2, 1, 1, 1, 0, 2])
```

In [42]:
```
print(np.concatenate((y_test.reshape(len(y_test),1),y_pred.reshape(len(y_pred),1)),
```

```
[[1 1]
 [1 1]
 [0 0]
 [0 0]
 [2 2]
 [1 1]
 [0 0]
 [2 2]
 [2 2]
 [0 2]
 [0 0]
 [0 0]
 [0 0]
 [2 2]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [2 2]
 [2 2]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [2 2]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 2]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [2 2]
 [0 0]
 [1 1]
 [2 2]
 [1 1]
 [2 2]
 [1 1]
 [1 1]
 [2 2]
 [2 2]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [2 2]
 [0 0]
```

```
[0 0]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[0 0]
[1 1]
[2 2]
[0 0]
[0 0]
[0 0]
[1 1]
[2 2]
[2 2]
[1 1]
[2 2]
[2 2]
[0 0]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[2 2]
[0 0]
[0 0]
[0 0]
[2 2]
[1 1]
[0 0]
[1 1]
[0 0]
[1 1]
[0 0]
[2 2]
[2 2]
[1 1]
[2 2]
[1 1]
[2 2]
[0 0]
[0 0]
[2 2]
[2 2]
[0 0]
[2 2]
```

```
[2 2]
[1 1]
[1 1]
[0 0]
[0 0]
[2 2]
[0 0]
[2 2]
[2 2]
[2 2]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[2 2]
[0 0]
[0 0]
[0 0]
[2 2]
[1 1]
[0 0]
[0 0]
[2 2]
[2 2]
[0 0]
[2 2]
[1 1]
[0 0]
[0 0]
[2 2]
[0 2]
[1 1]
[0 0]
[1 1]
[2 2]
[1 1]
[2 2]
[1 1]
[2 2]
[0 0]
[2 2]
[1 1]
[2 2]
[2 2]
[1 1]
[1 1]
[2 2]
[0 0]
[1 1]
[2 2]
[2 2]
[2 0]
[2 2]
[1 1]
```

```
[1 1]
[1 1]
[0 0]
[1 1]
[2 2]
[1 1]
[0 0]
[0 0]
[2 2]
[1 1]
[2 2]
[1 1]
[1 1]
[1 1]
[2 2]
[2 2]
[0 0]
[1 1]
[0 0]
[1 1]
[0 0]
[1 1]
[1 1]
[2 2]
[2 2]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[1 2]
[0 0]
[1 1]
[2 2]
[0 0]
[1 1]
[2 2]
[0 0]
[0 0]
[2 2]
[2 2]
[1 1]
[2 2]
[1 1]
[1 1]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[2 2]
[0 0]
[0 0]
[2 2]
[0 0]
[1 1]
```

```
[1 1]
[2 2]
[0 0]
[1 1]
[2 2]
[1 1]
[2 2]
[1 1]
[2 2]
[2 2]
[0 0]
[0 0]
[0 0]
[0 0]
[2 2]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[1 1]
[0 0]
[1 1]
[2 2]
[2 2]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[2 2]
[2 2]
[2 2]
[1 1]
[1 1]
[0 0]
[1 1]
[2 2]
[2 2]
[2 2]
[0 0]
[0 0]
[1 1]
[2 2]
[0 0]
[0 0]
[1 1]
[1 1]
[2 2]
[0 0]
[1 1]
[1 1]
[0 0]
```

```
[1 1]
[2 2]
[0 0]
[1 1]
[0 2]
[2 2]
[2 2]
[0 0]
[0 0]
[2 2]
[1 1]
[0 0]
[2 2]
[1 1]
[1 1]
[0 2]
[2 2]
[2 2]
[2 2]
[1 1]
[0 0]
[2 2]
[0 0]
[1 1]
[0 0]
[1 1]
[2 2]
[2 2]
[0 0]
[1 1]
[2 2]
[1 1]
[2 2]
[0 0]
[2 2]
[2 2]
[2 2]
[1 1]
[2 2]
[2 2]
[2 2]
[2 2]
[2 2]
[0 0]
[2 2]
[2 2]
[0 0]
[2 2]
[2 2]
[1 1]
[2 1]
[1 1]
[0 0]
[1 1]
[2 2]
[2 2]
[0 0]
```

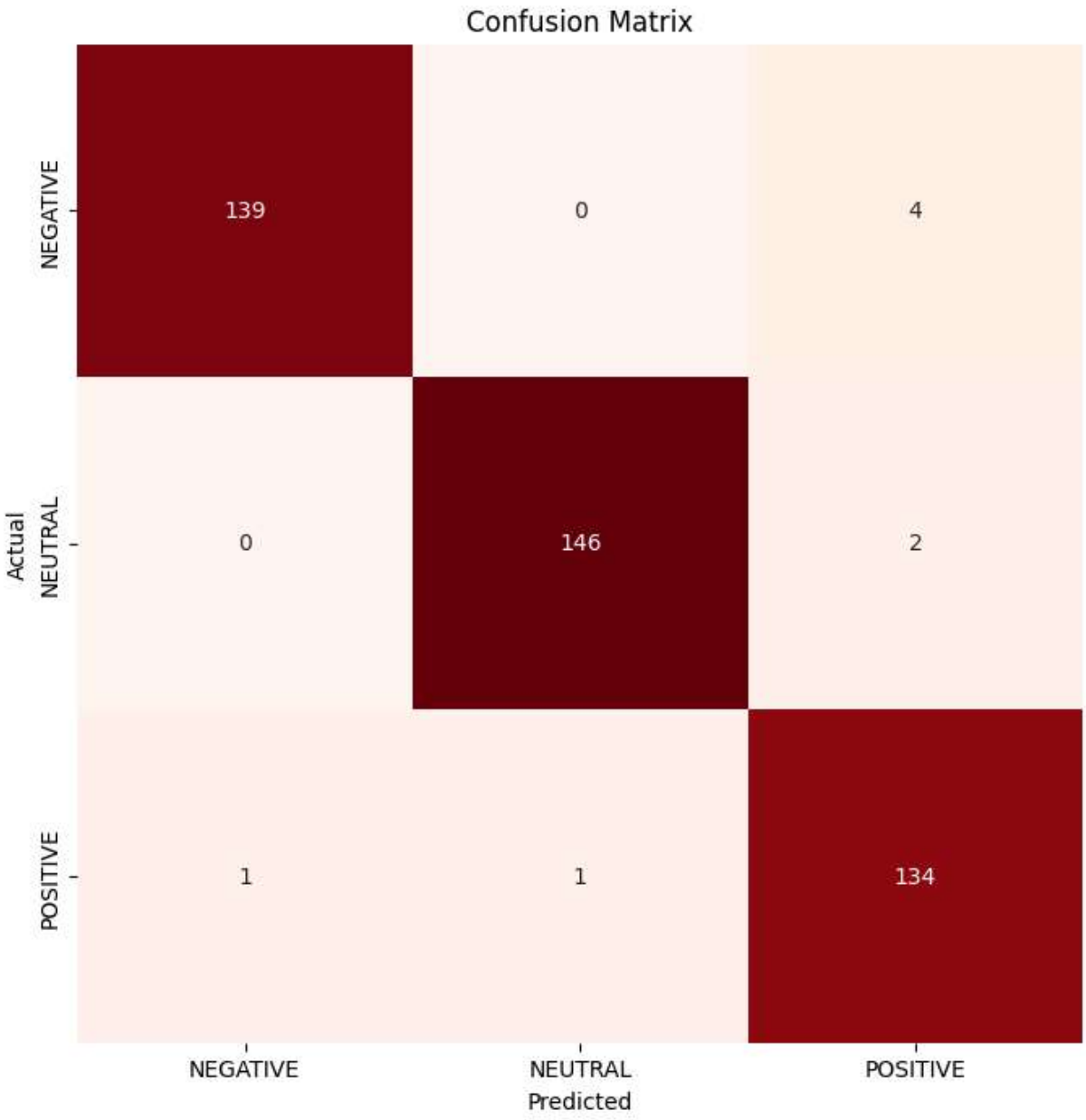
```
[2 2]
[1 1]
[0 0]
[2 2]
[0 0]
[1 1]
[2 2]
[2 2]
[0 0]
[0 0]
[2 2]
[0 0]
[0 0]
[1 1]
[2 2]
[2 2]
[2 2]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[2 2]
[0 0]
[2 2]
[1 1]
[1 1]
[2 2]
[2 2]
[1 1]
[0 0]
[2 2]
[2 2]
[2 2]
[2 2]
[2 2]
[1 1]
[0 0]
[1 1]
[2 2]
[1 1]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[2 2]
[1 1]
[2 2]
[1 1]
[2 2]
[0 0]
[0 0]
[1 1]
[2 2]
```

```
[2 2]
[0 0]
[1 1]
[0 0]
[0 0]
[2 2]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[2 2]
[0 0]
[2 2]
[1 1]
[2 2]
[1 1]
[1 1]
[1 1]
[0 0]
[2 2]
[1 1]
[0 0]
[2 2]
[1 1]
[1 1]
[1 1]
[0 0]
[2 2]]
```

In [43]:

```
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 8))
sns.heatmap(cm, annot=True, vmin=0, fmt='g', cbar=False, cmap='Reds')
plt.xticks(np.arange(3) + 0.5, label_mapping.keys())
plt.yticks(np.arange(3) + 0.5, label_mapping.keys())
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [44]:
```
clr = classification_report(y_test, y_pred, target_names=label_mapping.keys())
print("Classification Report:\n----------------------\n", clr)
```

```
Classification Report:
----------------------
              precision    recall  f1-score   support

    NEGATIVE       0.99      0.97      0.98       143
     NEUTRAL       0.99      0.99      0.99       148
    POSITIVE       0.96      0.99      0.97       136

    accuracy                           0.98       427
   macro avg       0.98      0.98      0.98       427
weighted avg       0.98      0.98      0.98       427
```