In [1]:

```python
#STEP 1: Import Required Libraries
import pandas as pd
import re
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.naive_bayes import MultinomialNB, GaussianNB

from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
    roc_curve,
    auc
)

import matplotlib.pyplot as plt
```

In [2]:

```python
#STEP 2: Load the Dataset
df = pd.read_csv(r"D:\Downloads\archive\spam.csv", encoding='latin-1')
```

In [3]:

```python
#STEP 3: Keep Only Required Columns
df = df[['v1', 'v2']]
df.columns = ['label', 'message']
```

In [4]:

```python
#STEP 4: Convert Labels (spam / ham → numbers)
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
df.head()
```

Out[4]:

	label	message
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

In [5]:

```python
#STEP 5: Text Preprocessing
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z ]', ' ', text)
    return text

df['message'] = df['message'].apply(clean_text)
```

In [6]:

```python
#STEP 6: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    df['message'], df['label'], test_size=0.2, random_state=42
)
```

In [18]:

```python
#STEP 7: Feature Extraction (TF-IDF)
tfidf = TfidfVectorizer(stop_words='english')
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

In [8]:

```python
#STEP 8: Multinomial Naive Bayes
mnb = MultinomialNB()
mnb.fit(X_train_tfidf, y_train)

mnb_pred = mnb.predict(X_test_tfidf)
```

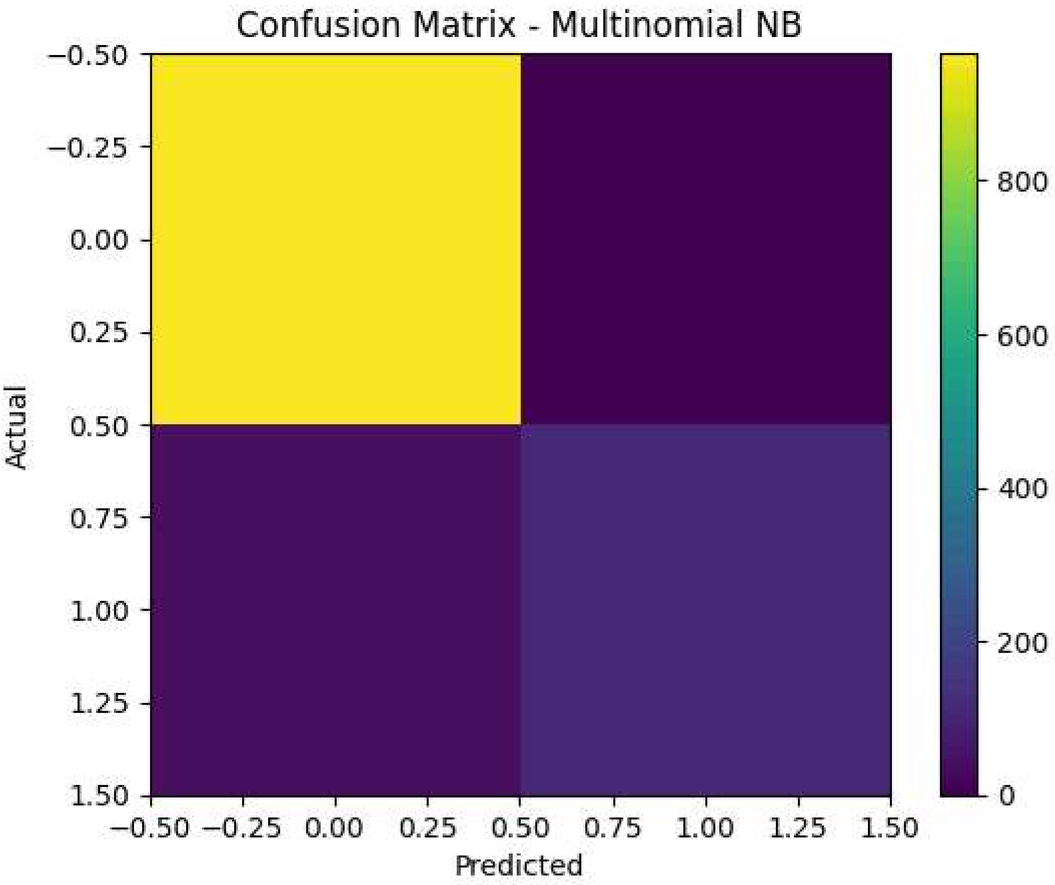
In [9]:

```python
#STEP 9: Performance Metrics (Multinomial NB)
print("Multinomial Naive Bayes")
print("Accuracy :", accuracy_score(y_test, mnb_pred))
print("Precision:", precision_score(y_test, mnb_pred))
print("Recall   :", recall_score(y_test, mnb_pred))
print("F1 Score :", f1_score(y_test, mnb_pred))
```

```
Multinomial Naive Bayes
Accuracy : 0.9659192825112107
Precision: 1.0
Recall   : 0.7466666666666667
F1 Score : 0.8549618320610687
```

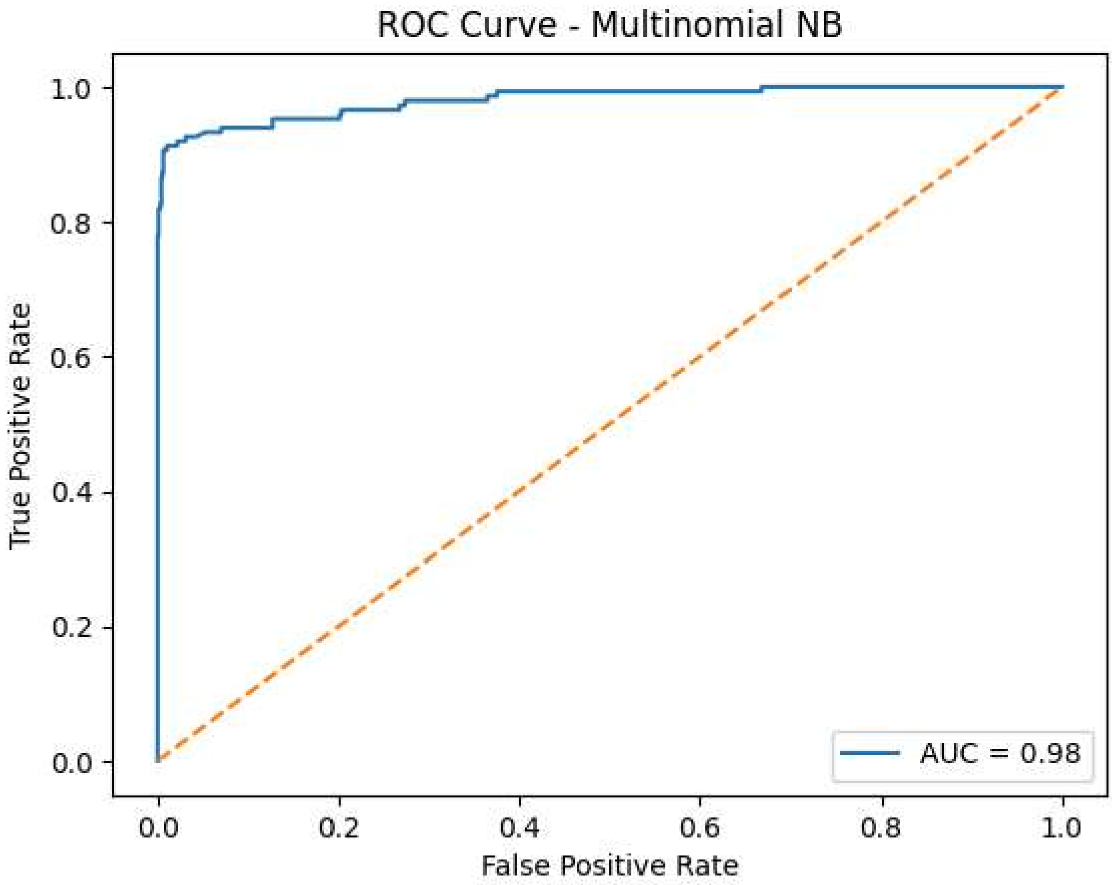
In [10]:

```python
#STEP 10: Confusion Matrix (Multinomial NB)
cm_mnb = confusion_matrix(y_test, mnb_pred)

plt.figure()
plt.imshow(cm_mnb)
plt.title("Confusion Matrix - Multinomial NB")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.colorbar()
plt.show()
```



In [11]:

```
#STEP 11: ROC Curve (Multinomial NB)
mnb_probs = mnb.predict_proba(X_test_tfidf)[:,1]
fpr, tpr, _ = roc_curve(y_test, mnb_probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label="AUC = %.2f" % roc_auc)
plt.plot([0,1],[0,1],'--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Multinomial NB")
plt.legend()
plt.show()
```



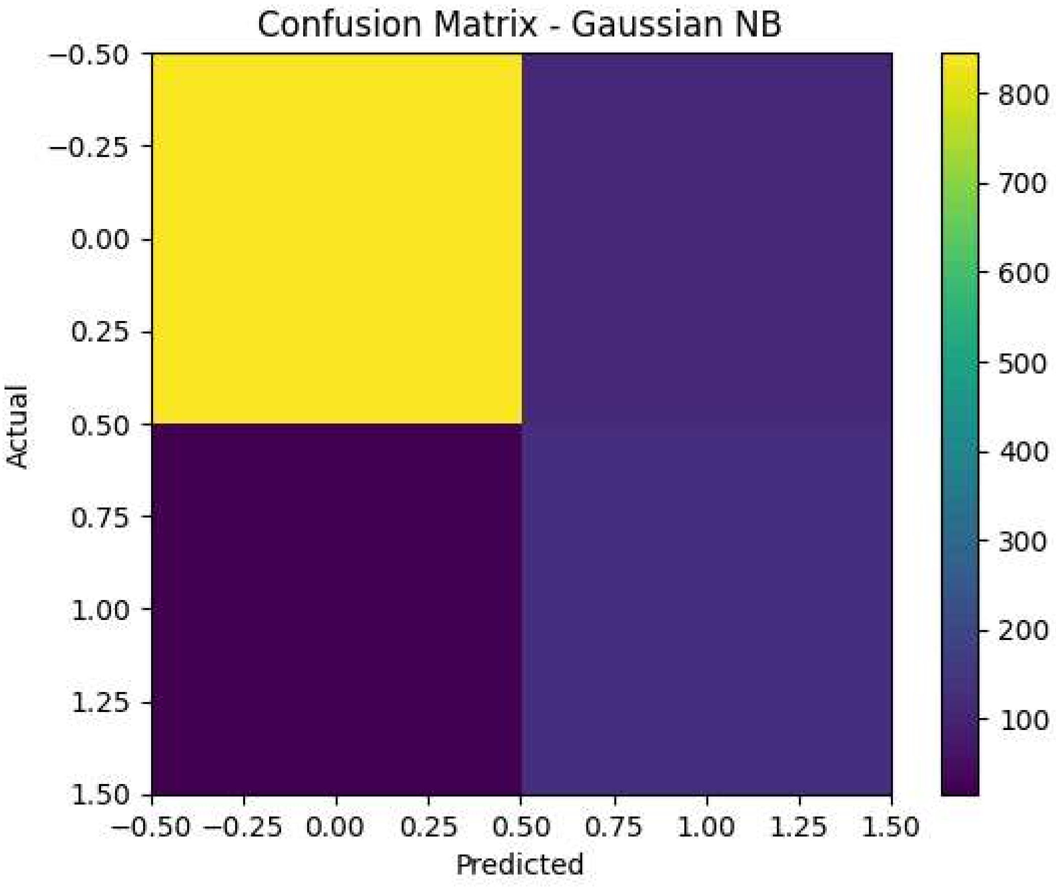
```
In [12]: #STEP 12: Bag of Words (CountVectorizer)
         bow = CountVectorizer(stop_words='english')
         X_train_bow = bow.fit_transform(X_train).toarray()
         X_test_bow = bow.transform(X_test).toarray()
```

```
In [13]: #STEP 13: Gaussian Naive Bayes
         gnb = GaussianNB()
         gnb.fit(X_train_bow, y_train)

         gnb_pred = gnb.predict(X_test_bow)
```

```
In [14]: #STEP 14: Performance Metrics (Gaussian NB)
         print("Gaussian Naive Bayes")
         print("Accuracy :", accuracy_score(y_test, gnb_pred))
         print("Precision:", precision_score(y_test, gnb_pred))
         print("Recall   :", recall_score(y_test, gnb_pred))
         print("F1 Score :", f1_score(y_test, gnb_pred))
```

```
Gaussian Naive Bayes
Accuracy : 0.8780269058295964
Precision: 0.5275590551181102
Recall   : 0.8933333333333333
F1 Score : 0.6633663366336634
```

```
In [15]: #STEP 13: Confusion Matrix (Gaussian NB)
         cm_gnb = confusion_matrix(y_test, gnb_pred)

         plt.figure()
plt.imshow(cm_gnb)
plt.title("Confusion Matrix - Gaussian NB")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.colorbar()
plt.show()
```



In [19]:
```
#STEP 14: Define Spam Keywords
spam_keywords = [
    "free", "win", "winner", "prize", "lottery",
    "offer", "urgent", "money", "loan",
    "credit", "click", "buy now", "cash"
]

def keyword_spam_check(text):
    for word in spam_keywords:
        if word in text:
            return 1
    return 0
```

In [21]:
```
#STEP 15:Real-World Testing
new_messages = [
    "Congratulations you won a free prize",
    "Hi, are we meeting tomorrow?",
    "Urgent loan approval waiting"
]

for msg in new_messages:
    cleaned = clean_text(msg)

    # Keyword check
    if keyword_spam_check(cleaned):
        print(msg, "-> Spam")
    else:
        vec = tfidf.transform([cleaned])
        pred = mnb.predict(vec)
        print(msg, "->", "Spam" if pred[0] == 1 else "Not Spam")
```

```
Congratulations you won a free prize -> Spam
Hi, are we meeting tomorrow? -> Not Spam
Urgent loan approval waiting -> Spam
```

In [ ]: