In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    ConfusionMatrixDisplay
)
import matplotlib.pyplot as plt
```

In [3]:

```
df = pd.read_csv(r"C:\Users\joshb\Downloads\imdb_reviews_sample.csv")

# Convert text labels ("pos"/"neg") to numeric labels (1/0)
df["label"] = df["sentiment"].map({"pos": 1, "neg": 0})
```

In [5]:

```
#split the data
X_train, X_test, y_train, y_test = train_test_split(
    df["review"], df["label"], test_size=0.3, random_state=42
)
```

In [7]:

```
#Convert reviews into TF-IDF feature vectors

vectorizer = TfidfVectorizer(stop_words="english", max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [9]:

```
log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train_tfidf, y_train)
y_pred_log = log_reg.predict(X_test_tfidf)

print(" Logistic Regression Accuracy:", round(accuracy_score(y_test, y_pred_log), 3))
print(classification_report(y_test, y_pred_log))
```

```
 Logistic Regression Accuracy: 0.828
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       305
           1       0.81      0.85      0.83       295

    accuracy                           0.83       600
   macro avg       0.83      0.83      0.83       600
weighted avg       0.83      0.83      0.83       600
```

In [11]:

```
#Train Naïve Bayes model

nb = MultinomialNB()
nb.fit(X_train_tfidf, y_train)
y_pred_nb = nb.predict(X_test_tfidf)

print("\n Naïve Bayes Accuracy:", round(accuracy_score(y_test, y_pred_nb), 3))
print(classification_report(y_test, y_pred_nb))
```

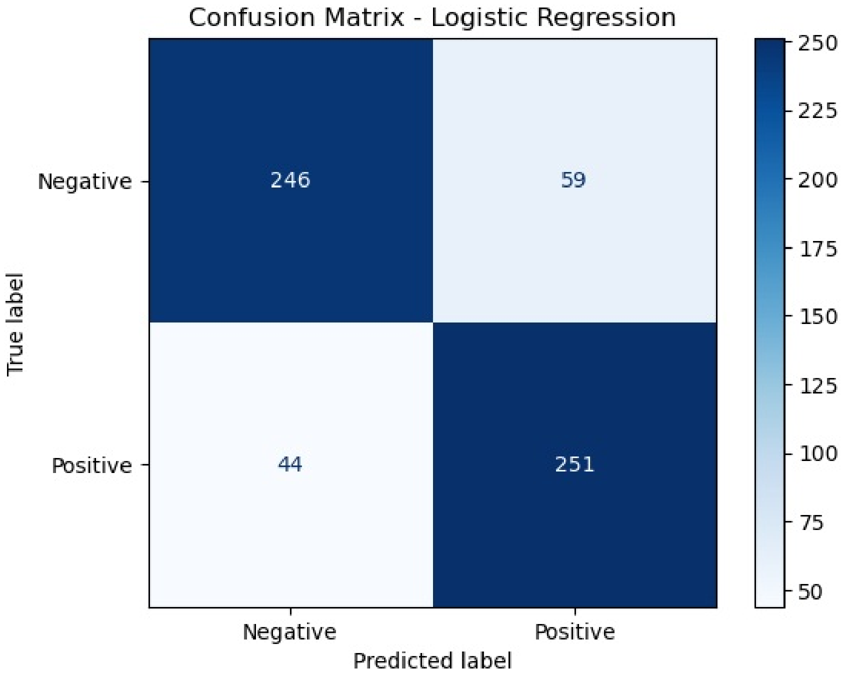
```
 Naïve Bayes Accuracy: 0.835
              precision    recall  f1-score   support

           0       0.82      0.87      0.84       305
           1       0.85      0.80      0.83       295

    accuracy                           0.83       600
   macro avg       0.84      0.83      0.83       600
weighted avg       0.84      0.83      0.83       600
```

In [13]:

```
# Show Confusion Matrix (Logistic Regression)
cm = confusion_matrix(y_test, y_pred_log)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Negative", "Positive"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```


Confusion Matrix - Logistic Regression
Negative
Positive
True label
246
59
44
251
Negative
Positive
Predicted label
250
225
200
175
150
125
100
75
50

```
In [ ]:
```