INTERN:-Sejal Rahul Sanas

*Intern ID:- CT4MGAE

*Domain:- Machine Learning

*Duration:-January 20, 2025, to May 20, 2025

*Company:- CODETECH IT SOLUTIONS

*Mentor:- Neela Santhosh Kumar

TASK TWO: SENTIMENT ANALYSIS WITH NLP

PERFORM SENTIMENT ANALYSIS ON A DATASET OF CUSTOMER REVIEWS USING TF-IDF VECTORIZATION AND LOGISTIC REGRESSION.

DELIVERABLE: A JUPYTER NOTEBOOK SHOWCASING PREPROCESSING, MODELING,AND SENTIMENT EVALUATION

In [1]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Step 1: Load the dataset
# Creating a simple dataset with customer reviews and sentiment labels
# 1 = Positive sentiment, 0 = Negative sentiment
data = pd.DataFrame({
    "review": [
        "This product is amazing!",
        "Terrible service, would not recommend.",
        "I love it, works perfectly.",
        "Not worth the money. Completely dissatisfied.",
        "Very good quality and fast delivery.",
        "Awful experience. The item arrived broken.",
        "Exceptional! Exceeded my expectations.",
        "Mediocre at best. Expected better.",
        "Fantastic! Will buy again.",
        "Poor customer service. Never buying here again."
    ],
    "sentiment": [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # Sentiment labels
})
```

In [3]:
```
# Display dataset preview
print("Dataset preview:")
print(data.head())
```

```
Dataset preview:
                                        review  sentiment
0                       This product is amazing!          1
1        Terrible service, would not recommend.          0
2                    I love it, works perfectly.          1
3  Not worth the money. Completely dissatisfied.          0
4          Very good quality and fast delivery.          1
```

In [4]:
```
# Step 2: Split the dataset into features (X) and labels (y)
X = data['review']
y = data['sentiment']
```

In [5]:
```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [6]:
```
# Step 4: Vectorize the text data using TF-IDF
# Transform the text data into numerical format for modeling
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [7]:
```
# Step 5: Initialize the Logistic Regression model
model = LogisticRegression(random_state=42)
```

In [8]:
```
# Step 6: Train the model on the training data
model.fit(X_train_tfidf, y_train)
```

Out[8]:
```
▾ LogisticRegression
LogisticRegression(random_state=42)
```

In [9]:
```
# Step 7: Make predictions on the test set
y_pred = model.predict(X_test_tfidf)
```

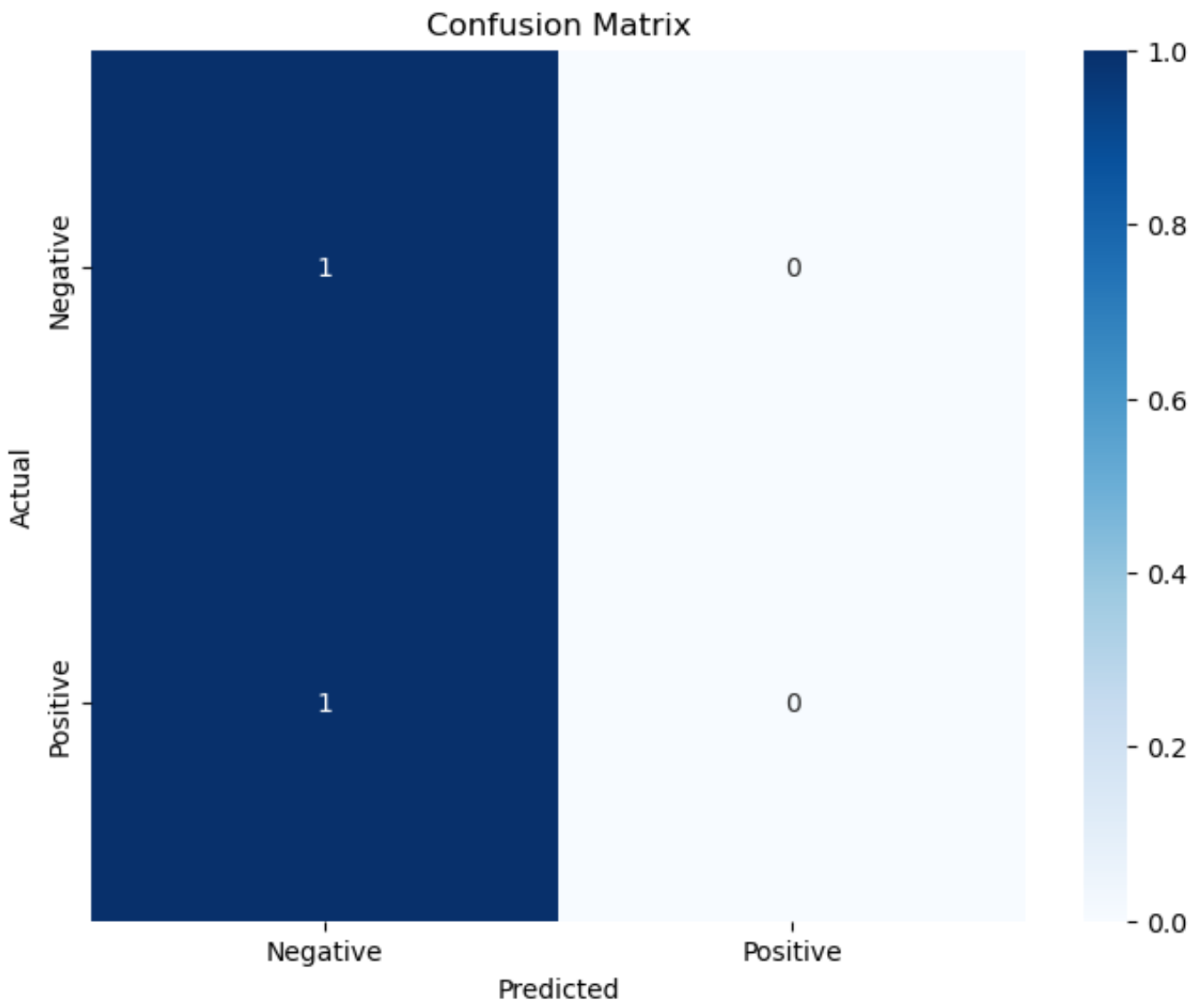
In [10]:
```
# Step 8: Evaluate the model
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

```
Model Accuracy: 0.50
```

In [11]:
```
# Print classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```

Classification Report:

              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2


C:\Users\Shejal Sanas\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Shejal Sanas\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Shejal Sanas\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

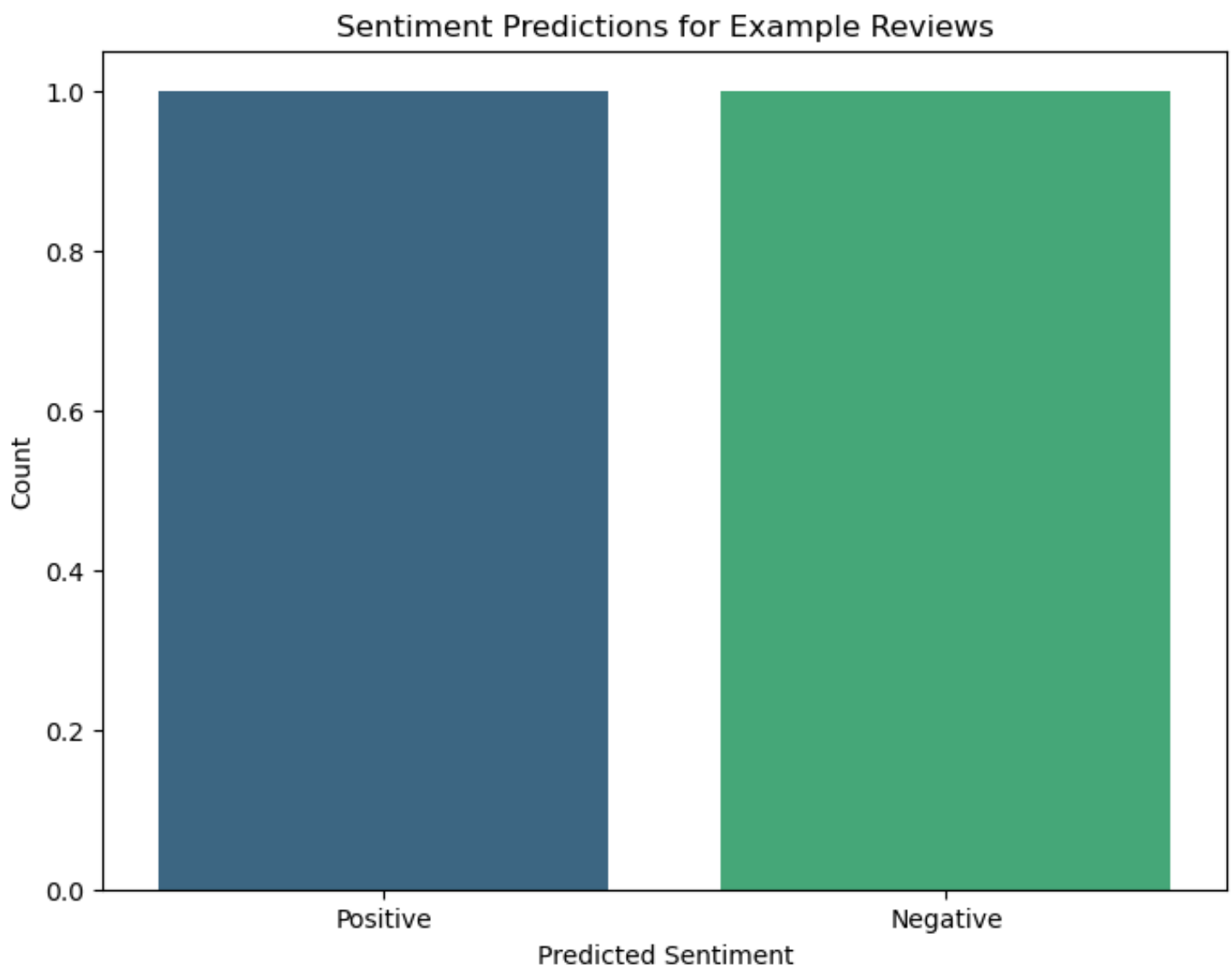
In [12]:
```
# Step 9: Plot the Confusion Matrix
# Visualize the confusion matrix to better understand predictions
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], ytic
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [13]:
```
# Step 10: Example predictions on new reviews
# Input some example reviews to test the model
example_reviews = [
    "The product quality is excellent!",
    "Horrible experience, will not return.",
    "Decent product but expected better.",
    "Absolutely love it! Fast delivery too."
]
# Transform the example reviews using the TF-IDF vectorizer
example_tfidf = vectorizer.transform(example_reviews)

# Predict the sentiment for each example review
example_preds = model.predict(example_tfidf)

# Display predictions
for review, sentiment in zip(example_reviews, example_preds):
    sentiment_label = "Positive" if sentiment == 1 else "Negative"
    print(f"Review: '{review}' | Sentiment: {sentiment_label}")
```

```
Review: 'The product quality is excellent!' | Sentiment: Positive
Review: 'Horrible experience, will not return.' | Sentiment: Negative
Review: 'Decent product but expected better.' | Sentiment: Negative
Review: 'Absolutely love it! Fast delivery too.' | Sentiment: Positive
```

In [14]:
```
# Step 11: Visualize Example Predictions
# Create a bar plot to visualize the sentiment predictions
sentiments = ["Positive" if s == 1 else "Negative" for s in example_preds]
plt.figure(figsize=(8, 6))
sns.barplot(x=sentiments, y=[1]*len(sentiments), palette="viridis", dodge=False)
plt.title("Sentiment Predictions for Example Reviews")
plt.xlabel("Predicted Sentiment")
plt.ylabel("Count")
plt.show()
```



In [15]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:
```

# Step 1: Load the dataset
# Creating a simple dataset with customer reviews and sentiment labels
# 1 = Positive sentiment, 0 = Negative sentiment
data = pd.DataFrame({
    "review": [
        "This product is amazing!",
        "Terrible service, would not recommend.",
        "I love it, works perfectly.",
        "Not worth the money. Completely dissatisfied.",
        "Very good quality and fast delivery.",
        "Awful experience. The item arrived broken.",
        "Exceptional! Exceeded my expectations.",
        "Mediocre at best. Expected better.",
        "Fantastic! Will buy again.",
        "Poor customer service. Never buying here again."
    ],
    "sentiment": [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # Sentiment labels
})
```

In [17]:
```
# Display dataset preview
print("Dataset preview:")
print(data.head(0))
```

```
Dataset preview:
Empty DataFrame
Columns: [review, sentiment]
Index: []
```

In [18]:
```
print(data.head(5))
print(data.tail(5))
```

```
                                          review  sentiment
0                       This product is amazing!          1
1          Terrible service, would not recommend.         0
2                     I love it, works perfectly.          1
3  Not worth the money. Completely dissatisfied.          0
4            Very good quality and fast delivery.         1
                                            review  sentiment
5       Awful experience. The item arrived broken.          0
6           Exceptional! Exceeded my expectations.          1
7               Mediocre at best. Expected better.          0
8                       Fantastic! Will buy again.          1
9  Poor customer service. Never buying here again.          0
```

In [19]:
```
# Step 2: Split the dataset into features (X) and labels (y)
X = data['review']
y = data['sentiment']
```

In [20]:
```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Vectorize the text data using TF-IDF
# Transform the text data into numerical format for modeling
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [21]:
```
# Step 5: Initialize the Logistic Regression model
model = LogisticRegression(random_state=42)
```

In [22]:
```
# Step 6: Train the model on the training data
model.fit(X_train_tfidf, y_train)
```

Out[22]:
```
LogisticRegression
LogisticRegression(random_state=42)
```

In [23]:
```

# Step 7: Make predictions on the test set
y_pred = model.predict(X_test_tfidf)
```

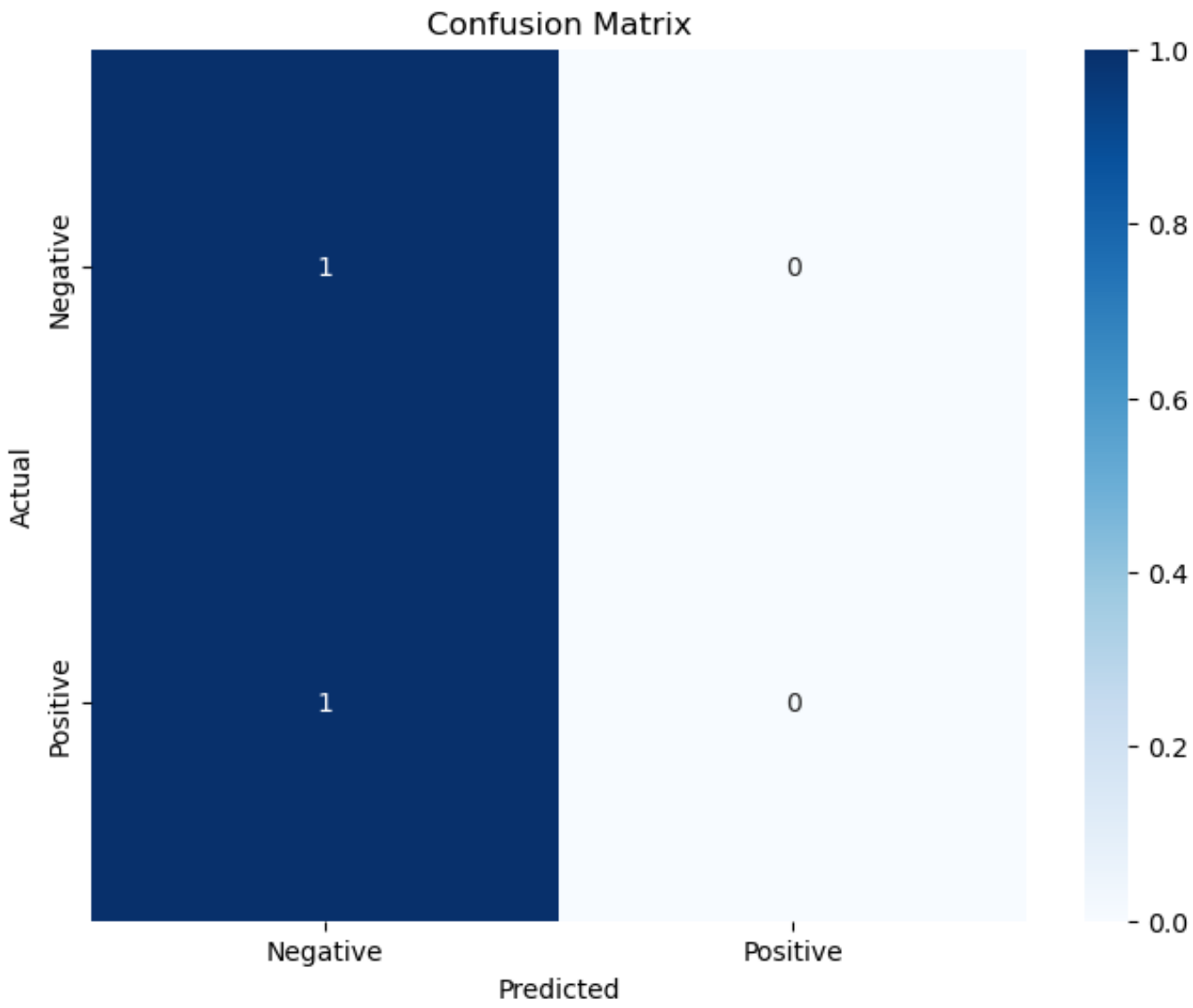
In [24]:
```
# Step 8: Evaluate the model
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

```
Model Accuracy: 0.50
```

In [25]:
```
# Print classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2


C:\Users\Shejal Sanas\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Shejal Sanas\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

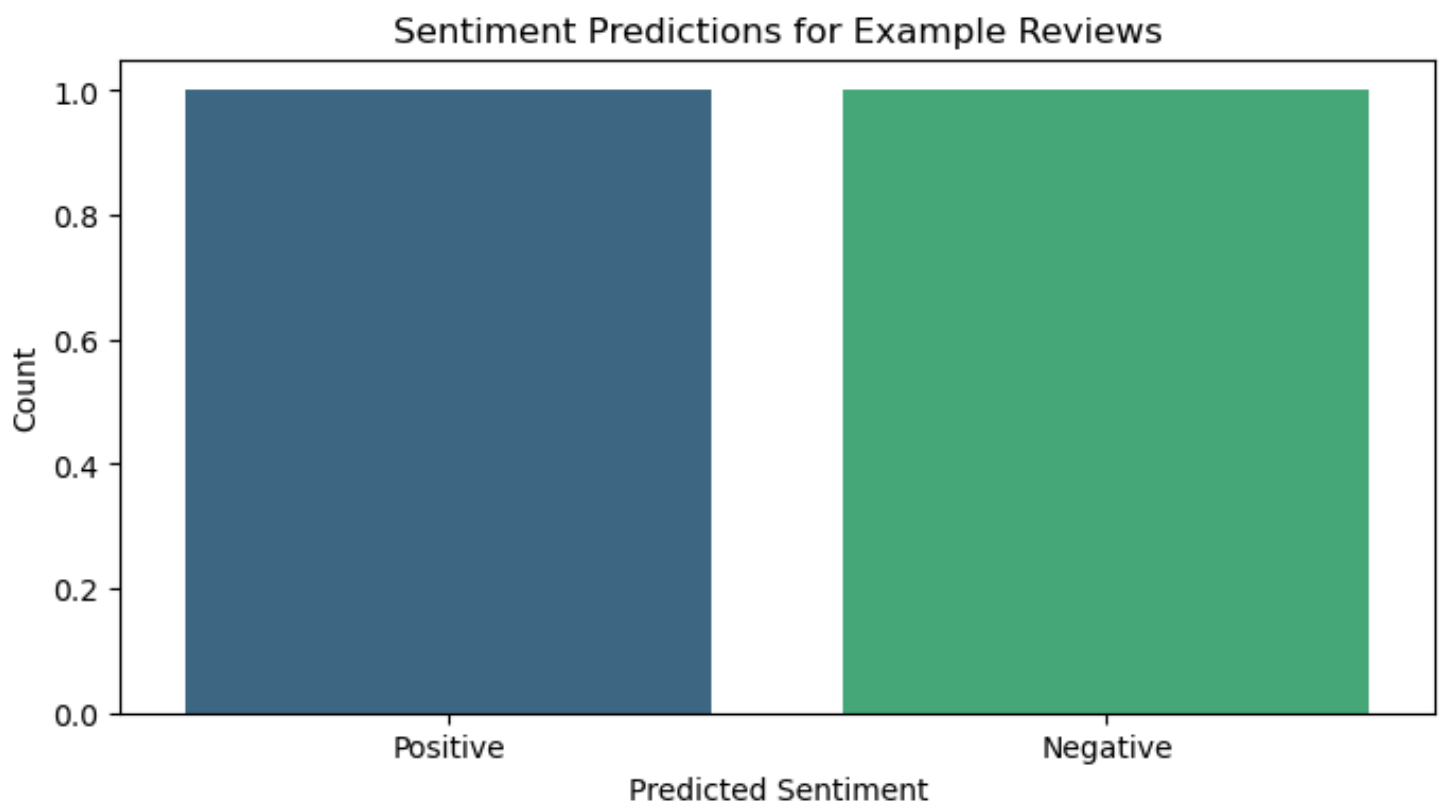
In [26]:
```
# Step 9: Plot the Confusion Matrix
# Visualize the confusion matrix to better understand predictions
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], ytic
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [27]:
```python
# Step 10: Example predictions on new reviews
# Input some example reviews to test the model
example_reviews = [
    "The product quality is excellent!",
    "Horrible experience, will not return.",
    "Decent product but expected better.",
    "Absolutely love it! Fast delivery too."
]

# Transform the example reviews using the TF-IDF vectorizer
example_tfidf = vectorizer.transform(example_reviews)

# Predict the sentiment for each example review
example_preds = model.predict(example_tfidf)

# Display predictions
for review, sentiment in zip(example_reviews, example_preds):
    sentiment_label = "Positive" if sentiment == 1 else "Negative"
    print(f"Review: '{review}' | Sentiment: {sentiment_label}")
```

```
Review: 'The product quality is excellent!' | Sentiment: Positive
Review: 'Horrible experience, will not return.' | Sentiment: Negative
Review: 'Decent product but expected better.' | Sentiment: Negative
Review: 'Absolutely love it! Fast delivery too.' | Sentiment: Positive
```

In [28]:
```python
# Step 11: Visualize Example Predictions
# Create a bar plot to visualize the sentiment predictions
sentiments = ["Positive" if s == 1 else "Negative" for s in example_preds]
plt.figure(figsize=(8, 4))
sns.barplot(x=sentiments, y=[1]*len(sentiments), palette="viridis", dodge=False)
plt.title("Sentiment Predictions for Example Reviews")
plt.xlabel("Predicted Sentiment")
plt.ylabel("Count")
plt.show()
```



#!/usr/bin/env python

#coding: utf-8