In [2]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
# Load dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed
column_names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'ex
                'oldpeak', 'slope', 'ca', 'thal', 'target']
df = pd.read_csv(url, names=column_names)
df
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3.0	0

303 rows × 14 columns

In [8]:
```
# Preprocessing: Replace '?' with NaN and drop missing values for simplicity
df.replace('?', np.nan, inplace=True)
df.dropna(inplace=True)
```

In [14]:
```
# Convert categorical features to numeric
df['ca'] = pd.to_numeric(df['ca'])
df['thal'] = pd.to_numeric(df['thal'])
print(df['ca'])
print(df['thal'])
```

```
0      0.0
1      3.0
2      2.0
3      0.0
4      0.0
      ...
297    0.0
298    0.0
299    2.0
300    1.0
301    1.0
Name: ca, Length: 297, dtype: float64
0      6.0
1      3.0
2      7.0
3      3.0
4      3.0
      ...
297    7.0
298    7.0
299    7.0
300    7.0
301    3.0
Name: thal, Length: 297, dtype: float64
```

In [15]:
```
# Convert target to binary classification (0: No Disease, 1: Disease)
df['target'] = df['target'].apply(lambda x: 1 if x > 0 else 0)
df['target']
```

Out[15]:
```
0      0
1      1
2      1
3      0
4      0
      ..
297    1
298    1
299    1
300    1
301    1
Name: target, Length: 297, dtype: int64
```

In [18]:
```
# Features and Target
X = df.drop('target', axis=1)
y = df['target']
print(X)
print(y)
```

```
     age  sex   cp  trestbps   chol  fbs  restecg  thalach  exang  oldpeak  \
0    63.0  1.0  1.0     145.0  233.0  1.0      2.0    150.0    0.0      2.3
1    67.0  1.0  4.0     160.0  286.0  0.0      2.0    108.0    1.0      1.5
2    67.0  1.0  4.0     120.0  229.0  0.0      2.0    129.0    1.0      2.6
3    37.0  1.0  3.0     130.0  250.0  0.0      0.0    187.0    0.0      3.5
4    41.0  0.0  2.0     130.0  204.0  0.0      2.0    172.0    0.0      1.4
..    ...  ...  ...       ...    ...  ...      ...      ...    ...      ...
297  57.0  0.0  4.0     140.0  241.0  0.0      0.0    123.0    1.0      0.2
298  45.0  1.0  1.0     110.0  264.0  0.0      0.0    132.0    0.0      1.2
299  68.0  1.0  4.0     144.0  193.0  1.0      0.0    141.0    0.0      3.4
300  57.0  1.0  4.0     130.0  131.0  0.0      0.0    115.0    1.0      1.2
301  57.0  0.0  2.0     130.0  236.0  0.0      2.0    174.0    0.0      0.0

     slope   ca  thal
0      3.0  0.0   6.0
1      2.0  3.0   3.0
2      2.0  2.0   7.0
3      3.0  0.0   3.0
4      1.0  0.0   3.0
..     ...  ...   ...
297    2.0  0.0   7.0
298    2.0  0.0   7.0
299    2.0  2.0   7.0
300    2.0  1.0   7.0
301    2.0  1.0   3.0

[297 rows x 13 columns]
0      0
1      1
2      1
3      0
4      0
      ..
297    1
298    1
299    1
300    1
301    1
Name: target, Length: 297, dtype: int64
```

In [20]:
```
# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [22]:
```
# Feature Scaling (Standardization)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [24]:
```
# Logistic Regression Model
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train_scaled, y_train)
```

Out[24]:
```
LogisticRegression
LogisticRegression(random_state=42)
```

In [27]:
```
# Predictions
y_pred = log_reg.predict(X_test_scaled)
y_pred
```

Out[27]:
```
array([0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1], dtype=int64)
```

In [29]:
```
# Performance Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

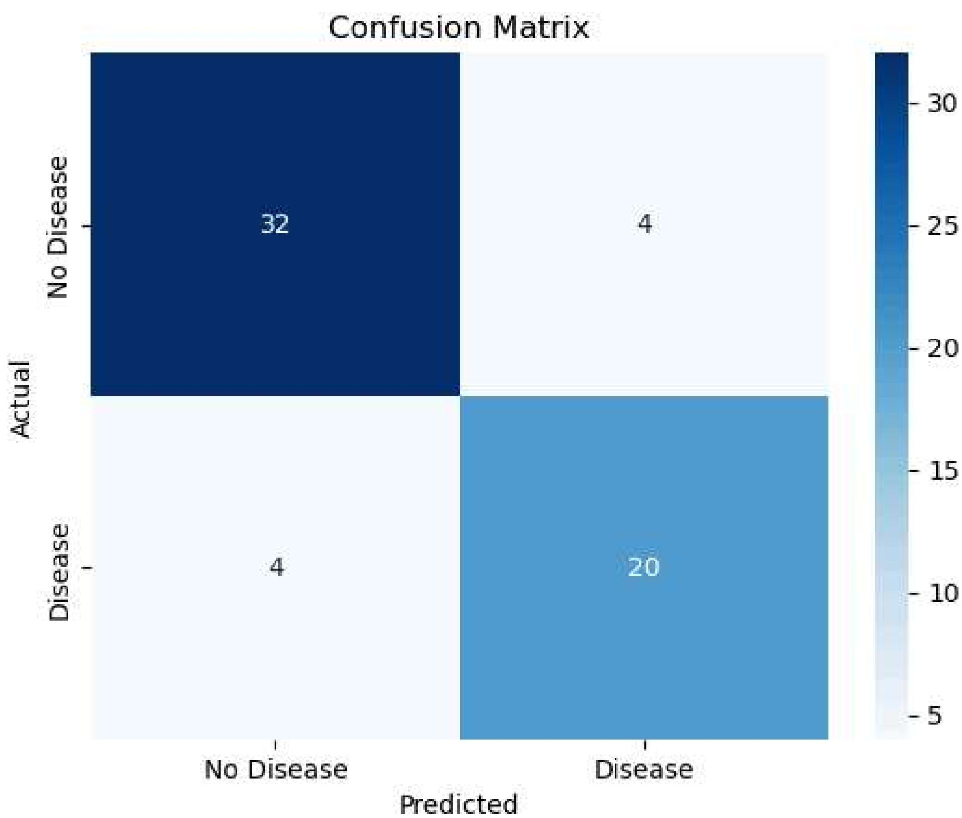
```
Accuracy: 0.8666666666666667

Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.89      0.89        36
           1       0.83      0.83      0.83        24

    accuracy                           0.87        60
   macro avg       0.86      0.86      0.86        60
weighted avg       0.87      0.87      0.87        60
```

In [33]:
```
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', '
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [34]:
```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
# Assuming the dataset is already loaded and the model is trained as explained previousl
# Pre-trained logistic regression model log_reg and scaler scaler should be available
# Function to collect user input dynamically and predict heart disease
def predict_heart_disease():
    """
    Predicts heart disease based on user input.
    """
    # Collect user input
    print("Please enter the following information:")

    age = int(input("Age: "))
    sex = int(input("Sex (1 = Male, 0 = Female): "))
    cp = int(input("Chest pain type (0-3): "))
    trestbps = int(input("Resting blood pressure (in mm Hg): "))
    chol = int(input("Serum cholesterol (in mg/dl): "))
    fbs = int(input("Fasting blood sugar > 120 mg/dl (1 = True, 0 = False): "))
    restecg = int(input("Resting electrocardiographic results (0-2): "))
    thalach = int(input("Maximum heart rate achieved: "))
    exang = int(input("Exercise induced angina (1 = Yes, 0 = No): "))
    oldpeak = float(input("ST depression induced by exercise relative to rest: "))
    slope = int(input("The slope of the peak exercise ST segment (0-2): "))
    ca = int(input("Number of major vessels (0-3): "))
    thal = int(input("Thalassemia (1 = Normal, 2 = Fixed defect, 3 = Reversible defect):
    # Organize the input data into a dictionary
    user_info = {
        'age': age,
        'sex': sex,
        'cp': cp,
        'trestbps': trestbps,
        'chol': chol,
        'fbs': fbs,
        'restecg': restecg,
        'thalach': thalach,
        'exang': exang,
        'oldpeak': oldpeak,
        'slope': slope,
        'ca': ca,
        'thal': thal
    }
    # Convert the input into a DataFrame
    user_data = pd.DataFrame([user_info])
    # Feature Scaling
    user_data_scaled = scaler.transform(user_data)
    # Prediction
    prediction = log_reg.predict(user_data_scaled)
    # Output the result
    if prediction[0] == 1:
        print("The person is at risk of heart disease.")
    else:
        print("The person is not at risk of heart disease.")
    return prediction[0]
# Example of calling the function to predict heart disease based on user input
predict_heart_disease()
```

```
Please enter the following information:
Age: 51
Sex (1 = Male, 0 = Female): 1
Chest pain type (0-3): 1
Resting blood pressure (in mm Hg): 122
Serum cholesterol (in mg/dl): 249
Fasting blood sugar > 120 mg/dl (1 = True, 0 = False): 1
Resting electrocardiographic results (0-2): 1
Maximum heart rate achieved: 120
Exercise induced angina (1 = Yes, 0 = No): 1
ST depression induced by exercise relative to rest: 1.5
The slope of the peak exercise ST segment (0-2): 2
Number of major vessels (0-3): 3
Thalassemia (1 = Normal, 2 = Fixed defect, 3 = Reversible defect): 3
The person is at risk of heart disease.
```

Out[34]:
```
1
```