In [5]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [11]:
```
#3. Look for datasets tagged with “Binary Classification”, or browse through d

# Got the binary classification dataset "Cardiovascular_Disease_Dataset" from

#4.Select a dataset that is labeled for binary classification. This means that
# Yes the selected dataset has the target variable as binary classification, w

#5. Make sure the dataset has a sufficient number of instances and features to
# As checked the dataset has required rows and sufficient number of features d

#6. Display the first few and last few rows of the dataset to get a sense of w
data = pd.read_csv('Cardiovascular_Disease_Dataset.csv')

print("Top 5 rows", data.head(5))

print("Bottom 5 rows", data.tail(5))
```

```
Top 5 rows    patientid  age  gender  chestpain  restingBP  serumcholestrol
\
0     103368   53       1          2        171               0
1     119250   40       1          0         94             229
2     119372   49       1          2        133             142
3     132514   43       1          0        138             295
4     146211   31       1          1        199               0

   fastingbloodsugar  restingrelectro  maxheartrate  exerciseangia  oldpeak
\
0                  0                1           147              0      5.3
1                  0                1           115              0      3.7
2                  0                0           202              1      5.0
3                  1                1           153              0      3.2
4                  0                2           136              0      5.3

   slope  noofmajorvessels  target
0      3                 3       1
1      1                 1       0
2      1                 0       0
3      2                 2       1
4      3                 2       1
Bottom 5 rows      patientid  age  gender  chestpain  restingBP  serumcholes
trol  \
995    9949544   48       1          2        139             349
996    9953423   47       1          3        143             258
997    9965859   69       1          0        156             434
998    9988507   45       1          1        186             417
999    9990855   25       1          0        158             270

     fastingbloodsugar  restingrelectro  maxheartrate  exerciseangia  oldpea
k  \
995                  0                2           183              1      5.
6
996                  1                1            98              1      5.
7
997                  1                0           196              0      1.
4
998                  0                1           117              1      5.
9
999                  0                0           143              1      4.
7

     slope  noofmajorvessels  target
995      2                 2       1
996      1                 0       0
997      3                 1       1
998      3                 2       1
999      0                 0       0
```

In [13]:
```
#7. Determine the number of instances and features in the dataset
num_instances, num_features = data.shape

print("Number of instances:", num_instances)
print("Number of features:", num_features)
```

```
Number of instances: 1000
Number of features: 14
```

In [12]:
```
column_names = data.columns
column_names
```

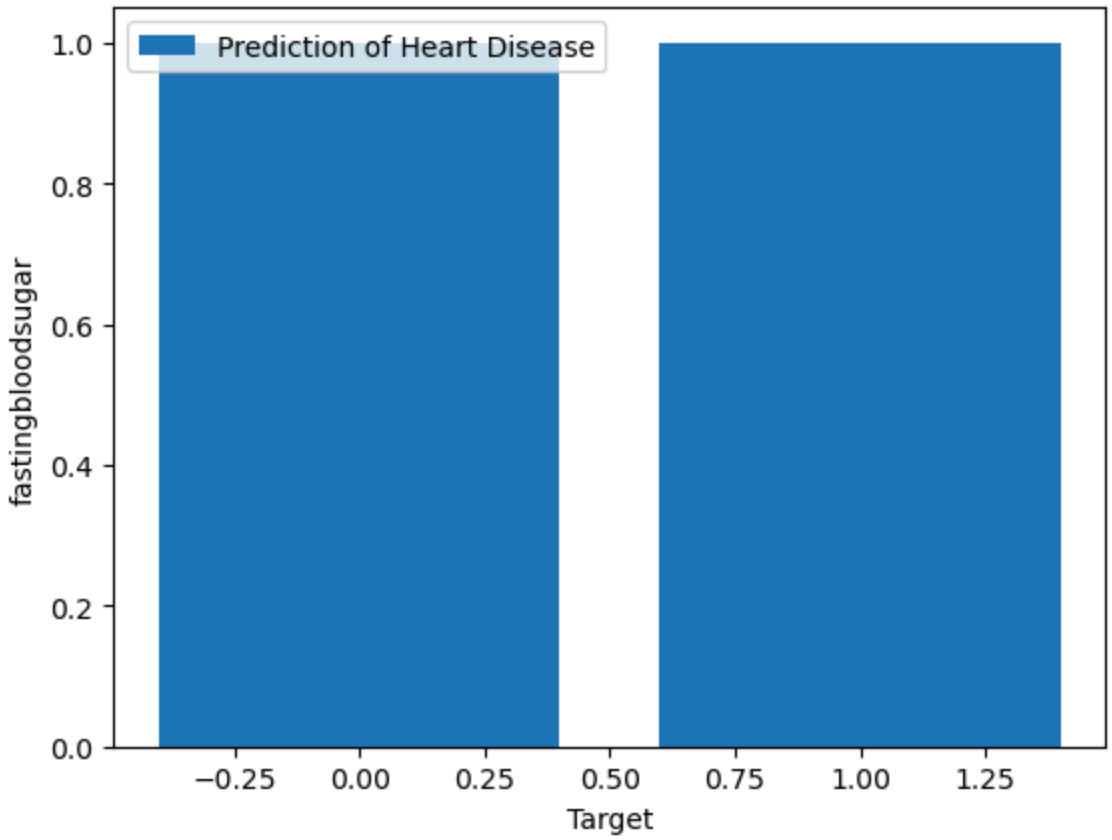
Out[12]:
```
Index(['patientid', 'age', 'gender', 'chestpain', 'restingBP',
       'serumcholestrol', 'fastingbloodsugar', 'restingrelectro',
       'maxheartrate', 'exerciseangia', 'oldpeak', 'slope', 'noofmajorvessel
s',
       'target'],
      dtype='object')
```

In [14]:
```
#8. Examine the features (columns) of the dataset, and determine their data ty
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   patientid          1000 non-null   int64  
 1   age                1000 non-null   int64  
 2   gender             1000 non-null   int64  
 3   chestpain          1000 non-null   int64  
 4   restingBP          1000 non-null   int64  
 5   serumcholestrol    1000 non-null   int64  
 6   fastingbloodsugar  1000 non-null   int64  
 7   restingrelectro    1000 non-null   int64  
 8   maxheartrate       1000 non-null   int64  
 9   exerciseangia      1000 non-null   int64  
 10  oldpeak            1000 non-null   float64
 11  slope              1000 non-null   int64  
 12  noofmajorvessels   1000 non-null   int64  
 13  target             1000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 109.5 KB
```

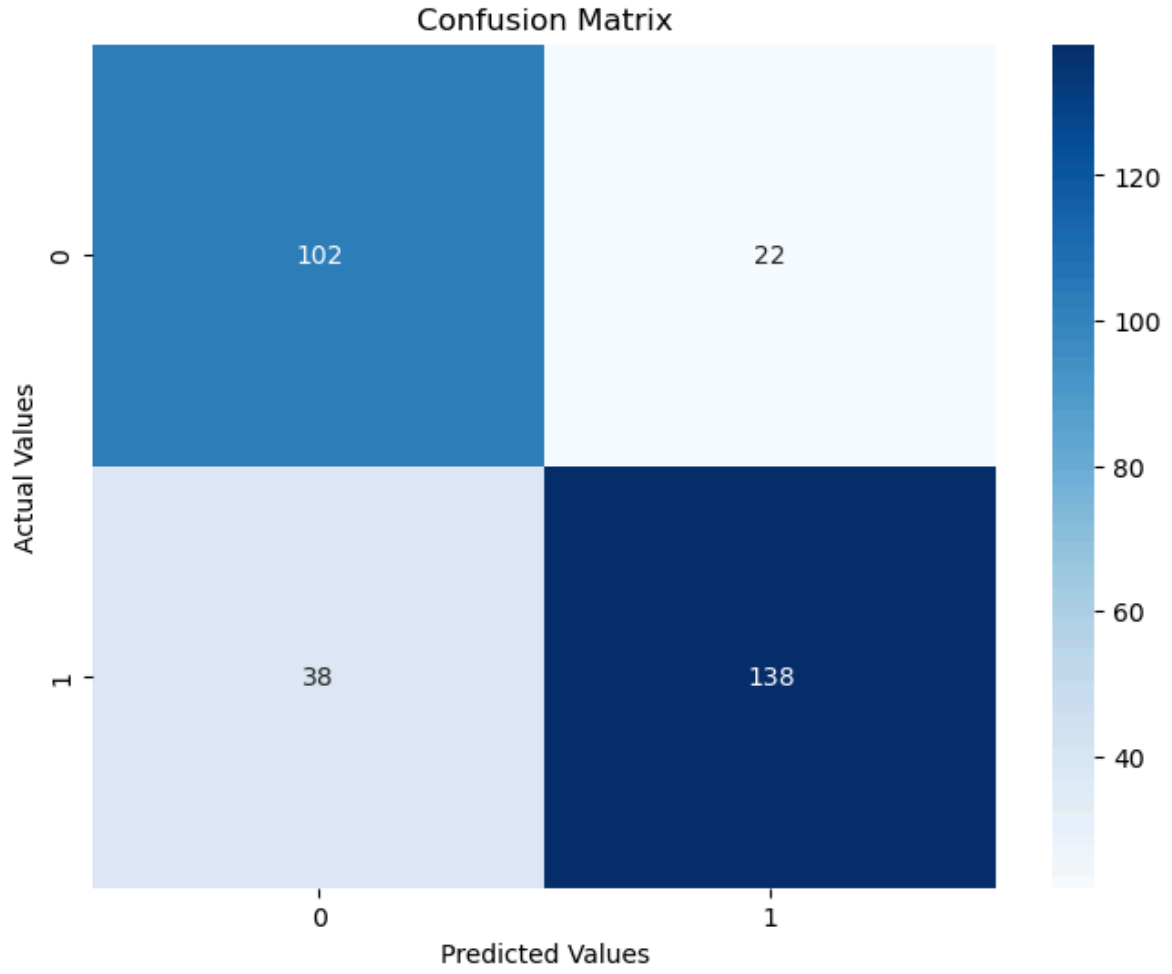
In [25]:
```
#9. Plot the dependant variable distribution. (use bar plot)
x = data["target"]
y = data["fastingbloodsugar"]
plt.bar(x,y,label="Prediction of Heart Disease")
plt.legend()
plt.xlabel("Target")
plt.ylabel("Fastingbloodsugar")
```

Out[25]: Text(0, 0.5, 'fastingbloodsugar')



In [37]:
```
#11. Split your dataset and explain your approach.
# I am choosing the below four independent variables for my prediction that if
from sklearn.metrics import confusion_matrix, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

#12. Design and fit a Logistic regression model
X = data[['fastingbloodsugar', 'maxheartrate', 'exerciseangia', 'chestpain']]
y = data['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randc

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred)
print("recall score is: ", recall_score(y_test, y_pred))

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap='Blues')
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()
plt.tight_layout()
plt.show()
```

```
recall score is:  0.7840909090909091
```



<Figure size 640x480 with 0 Axes>

In [ ]:
```
#13.Compute the evaluation metrics and discuss the ones most critical in your
# from the recall score points, we could see that there were individuals with
#14. Compute the confusion matrix and analyze it.
#After analysing the confusion matrix and could see that.
#False Positives (FP): The model incorrectly predicted 102 instances as positi
#False Negatives (FN): The model incorrectly predicted 22 instances as negativ
#True Negatives (TN): The model correctly predicted 38 instances as negative (
#True Positives (TP): The model correctly predicted 138 instances as positive
```

In [34]:
```
fastingbloodsugar = int(input("Enter 0 if fasting blood sugar or 1 if it is nc
maxheartrate = int(input("Enter the maximum heart rate: "))
exerciseangia = int(input("Enter 0 if exerciseangia or 1 if it is not: "))
input_data = [[fastingbloodsugar, maxheartrate, exerciseangia]]
outcome = model.predict(input_data)
print(f"Predicted Heart Disease:{outcome}")
```

```
Enter 0 if fasting blood sugar or 1 if it is not: 0
Enter the maximum heart rate: 55
Enter 0 if exerciseangia or 1 if it is not: 0
Predicted Heart Disease:[0]
```