In [1]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
```

In [2]:
```
# Load and prepare the dataset
data = pd.read_csv('heart.csv')
data.head()
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [3]:
```
X = data.iloc[:,:13].values
y = data["target"].values
```

In [4]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

In [5]:
```
# Scale the data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [6]:
```
# Build the ANN model
classifier = Sequential()
classifier.add(Dense(activation="relu", input_dim=13, units=8, kernel_in
classifier.add(Dense(activation="relu", units=14, kernel_initializer="un
classifier.add(Dense(activation="sigmoid", units=1, kernel_initializer="
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics
```

In [7]:
```
# Fit the model
classifier.fit(X_train, y_train, batch_size=8, epochs=100)
```

```
Epoch 1/100
27/27 [==============================] - 0s 485us/step - loss: 0.6927 -
accuracy: 0.5236
Epoch 2/100
27/27 [==============================] - 0s 384us/step - loss: 0.6901 -
accuracy: 0.5849
Epoch 3/100
27/27 [==============================] - 0s 344us/step - loss: 0.6792 -
accuracy: 0.8019
Epoch 4/100
27/27 [==============================] - 0s 326us/step - loss: 0.6514 -
accuracy: 0.8585
Epoch 5/100
27/27 [==============================] - 0s 300us/step - loss: 0.5988 -
accuracy: 0.8585
Epoch 6/100
27/27 [==============================] - 0s 326us/step - loss: 0.5281 -
accuracy: 0.8632
Epoch 7/100
27/27 [==============================] - 0s 305us/step - loss: 0.4630 -
accuracy: 0.8679
Epoch 8/100
27/27 [==============================] - 0s 326us/step - loss: 0.4108 -
accuracy: 0.8679
Epoch 9/100
27/27 [==============================] - 0s 305us/step - loss: 0.3800 -
accuracy: 0.8726
Epoch 10/100
27/27 [==============================] - 0s 312us/step - loss: 0.3587 -
accuracy: 0.8726
Epoch 11/100
27/27 [==============================] - 0s 338us/step - loss: 0.3480 -
accuracy: 0.8726
Epoch 12/100
27/27 [==============================] - 0s 323us/step - loss: 0.3390 -
accuracy: 0.8679
Epoch 13/100
27/27 [==============================] - 0s 334us/step - loss: 0.3342 -
accuracy: 0.8726
Epoch 14/100
27/27 [==============================] - 0s 305us/step - loss: 0.3300 -
accuracy: 0.8726
Epoch 15/100
27/27 [==============================] - 0s 331us/step - loss: 0.3272 -
accuracy: 0.8679
Epoch 16/100
27/27 [==============================] - 0s 329us/step - loss: 0.3241 -
accuracy: 0.8774
Epoch 17/100
27/27 [==============================] - 0s 307us/step - loss: 0.3215 -
accuracy: 0.8774
Epoch 18/100
27/27 [==============================] - 0s 349us/step - loss: 0.3201 -
accuracy: 0.8821
Epoch 19/100
27/27 [==============================] - 0s 307us/step - loss: 0.3188 -
accuracy: 0.8821
```

```
Epoch 20/100
27/27 [==============================] - 0s 341us/step - loss: 0.3175 -
accuracy: 0.8774
Epoch 21/100
27/27 [==============================] - 0s 328us/step - loss: 0.3172 -
accuracy: 0.8774
Epoch 22/100
27/27 [==============================] - 0s 321us/step - loss: 0.3161 -
accuracy: 0.8774
Epoch 23/100
27/27 [==============================] - 0s 322us/step - loss: 0.3140 -
accuracy: 0.8774
Epoch 24/100
27/27 [==============================] - 0s 311us/step - loss: 0.3133 -
accuracy: 0.8774
Epoch 25/100
27/27 [==============================] - 0s 324us/step - loss: 0.3124 -
accuracy: 0.8821
Epoch 26/100
27/27 [==============================] - 0s 305us/step - loss: 0.3126 -
accuracy: 0.8821
Epoch 27/100
27/27 [==============================] - 0s 325us/step - loss: 0.3110 -
accuracy: 0.8821
Epoch 28/100
27/27 [==============================] - 0s 341us/step - loss: 0.3097 -
accuracy: 0.8821
Epoch 29/100
27/27 [==============================] - 0s 798us/step - loss: 0.3096 -
accuracy: 0.8821
Epoch 30/100
27/27 [==============================] - 0s 485us/step - loss: 0.3081 -
accuracy: 0.8821
Epoch 31/100
27/27 [==============================] - 0s 329us/step - loss: 0.3069 -
accuracy: 0.8821
Epoch 32/100
27/27 [==============================] - 0s 319us/step - loss: 0.3055 -
accuracy: 0.8821
Epoch 33/100
27/27 [==============================] - 0s 333us/step - loss: 0.3053 -
accuracy: 0.8821
Epoch 34/100
27/27 [==============================] - 0s 312us/step - loss: 0.3038 -
accuracy: 0.8821
Epoch 35/100
27/27 [==============================] - 0s 313us/step - loss: 0.3029 -
accuracy: 0.8821
Epoch 36/100
27/27 [==============================] - 0s 328us/step - loss: 0.3026 -
accuracy: 0.8868
Epoch 37/100
27/27 [==============================] - 0s 317us/step - loss: 0.3006 -
accuracy: 0.8868
Epoch 38/100
27/27 [==============================] - 0s 309us/step - loss: 0.2997 -
accuracy: 0.8868
```

```
Epoch 39/100
27/27 [==============================] - 0s 312us/step - loss: 0.3000 -
accuracy: 0.8868
Epoch 40/100
27/27 [==============================] - 0s 320us/step - loss: 0.2981 -
accuracy: 0.8868
Epoch 41/100
27/27 [==============================] - 0s 316us/step - loss: 0.2972 -
accuracy: 0.8868
Epoch 42/100
27/27 [==============================] - 0s 318us/step - loss: 0.2959 -
accuracy: 0.8868
Epoch 43/100
27/27 [==============================] - 0s 310us/step - loss: 0.2968 -
accuracy: 0.8915
Epoch 44/100
27/27 [==============================] - 0s 311us/step - loss: 0.2936 -
accuracy: 0.8868
Epoch 45/100
27/27 [==============================] - 0s 313us/step - loss: 0.2926 -
accuracy: 0.8915
Epoch 46/100
27/27 [==============================] - 0s 311us/step - loss: 0.2911 -
accuracy: 0.8915
Epoch 47/100
27/27 [==============================] - 0s 304us/step - loss: 0.2907 -
accuracy: 0.8915
Epoch 48/100
27/27 [==============================] - 0s 306us/step - loss: 0.2896 -
accuracy: 0.8868
Epoch 49/100
27/27 [==============================] - 0s 305us/step - loss: 0.2883 -
accuracy: 0.8915
Epoch 50/100
27/27 [==============================] - 0s 304us/step - loss: 0.2879 -
accuracy: 0.8868
Epoch 51/100
27/27 [==============================] - 0s 309us/step - loss: 0.2866 -
accuracy: 0.8915
Epoch 52/100
27/27 [==============================] - 0s 305us/step - loss: 0.2850 -
accuracy: 0.8962
Epoch 53/100
27/27 [==============================] - 0s 306us/step - loss: 0.2850 -
accuracy: 0.8962
Epoch 54/100
27/27 [==============================] - 0s 308us/step - loss: 0.2828 -
accuracy: 0.8962
Epoch 55/100
27/27 [==============================] - 0s 311us/step - loss: 0.2828 -
accuracy: 0.8868
Epoch 56/100
27/27 [==============================] - 0s 312us/step - loss: 0.2798 -
accuracy: 0.8915
Epoch 57/100
27/27 [==============================] - 0s 313us/step - loss: 0.2784 -
accuracy: 0.8868
```

```
Epoch 58/100
27/27 [==============================] - 0s 301us/step - loss: 0.2774 -
accuracy: 0.8915
Epoch 59/100
27/27 [==============================] - 0s 310us/step - loss: 0.2769 -
accuracy: 0.8868
Epoch 60/100
27/27 [==============================] - 0s 314us/step - loss: 0.2755 -
accuracy: 0.8962
Epoch 61/100
27/27 [==============================] - 0s 316us/step - loss: 0.2751 -
accuracy: 0.8868
Epoch 62/100
27/27 [==============================] - 0s 315us/step - loss: 0.2734 -
accuracy: 0.8962
Epoch 63/100
27/27 [==============================] - 0s 339us/step - loss: 0.2728 -
accuracy: 0.9009
Epoch 64/100
27/27 [==============================] - 0s 315us/step - loss: 0.2707 -
accuracy: 0.9009
Epoch 65/100
27/27 [==============================] - 0s 321us/step - loss: 0.2697 -
accuracy: 0.9009
Epoch 66/100
27/27 [==============================] - 0s 333us/step - loss: 0.2682 -
accuracy: 0.9009
Epoch 67/100
27/27 [==============================] - 0s 369us/step - loss: 0.2681 -
accuracy: 0.8962
Epoch 68/100
27/27 [==============================] - 0s 339us/step - loss: 0.2669 -
accuracy: 0.9009
Epoch 69/100
27/27 [==============================] - 0s 303us/step - loss: 0.2658 -
accuracy: 0.8962
Epoch 70/100
27/27 [==============================] - 0s 367us/step - loss: 0.2643 -
accuracy: 0.9009
Epoch 71/100
27/27 [==============================] - 0s 322us/step - loss: 0.2630 -
accuracy: 0.9009
Epoch 72/100
27/27 [==============================] - 0s 310us/step - loss: 0.2612 -
accuracy: 0.9009
Epoch 73/100
27/27 [==============================] - 0s 331us/step - loss: 0.2609 -
accuracy: 0.9009
Epoch 74/100
27/27 [==============================] - 0s 320us/step - loss: 0.2593 -
accuracy: 0.9009
Epoch 75/100
27/27 [==============================] - 0s 339us/step - loss: 0.2596 -
accuracy: 0.9009
Epoch 76/100
27/27 [==============================] - 0s 332us/step - loss: 0.2593 -
accuracy: 0.9009
```

```
Epoch 77/100
27/27 [==============================] - 0s 325us/step - loss: 0.2576 -
accuracy: 0.9009
Epoch 78/100
27/27 [==============================] - 0s 320us/step - loss: 0.2555 -
accuracy: 0.9009
Epoch 79/100
27/27 [==============================] - 0s 337us/step - loss: 0.2548 -
accuracy: 0.9009
Epoch 80/100
27/27 [==============================] - 0s 328us/step - loss: 0.2542 -
accuracy: 0.9057
Epoch 81/100
```

```
27/27 [==============================] - 0s 317us/step - loss: 0.2529 -
accuracy: 0.9057
Epoch 82/100
27/27 [==============================] - 0s 331us/step - loss: 0.2521 -
accuracy: 0.9057
Epoch 83/100
27/27 [==============================] - 0s 304us/step - loss: 0.2523 -
accuracy: 0.9009
Epoch 84/100
27/27 [==============================] - 0s 309us/step - loss: 0.2507 -
accuracy: 0.9057
Epoch 85/100
27/27 [==============================] - 0s 303us/step - loss: 0.2497 -
accuracy: 0.9104
Epoch 86/100
27/27 [==============================] - 0s 302us/step - loss: 0.2479 -
accuracy: 0.9151
Epoch 87/100
27/27 [==============================] - 0s 297us/step - loss: 0.2475 -
accuracy: 0.9104
Epoch 88/100
27/27 [==============================] - 0s 309us/step - loss: 0.2451 -
accuracy: 0.9104
Epoch 89/100
27/27 [==============================] - 0s 302us/step - loss: 0.2445 -
accuracy: 0.9104
Epoch 90/100
27/27 [==============================] - 0s 308us/step - loss: 0.2426 -
accuracy: 0.9104
Epoch 91/100
27/27 [==============================] - 0s 309us/step - loss: 0.2420 -
accuracy: 0.9104
Epoch 92/100
27/27 [==============================] - 0s 311us/step - loss: 0.2407 -
accuracy: 0.9104
Epoch 93/100
27/27 [==============================] - 0s 306us/step - loss: 0.2402 -
accuracy: 0.9104
Epoch 94/100
27/27 [==============================] - 0s 303us/step - loss: 0.2390 -
accuracy: 0.9104
Epoch 95/100
27/27 [==============================] - 0s 309us/step - loss: 0.2389 -
accuracy: 0.9151
Epoch 96/100
27/27 [==============================] - 0s 306us/step - loss: 0.2371 -
accuracy: 0.9151
Epoch 97/100
27/27 [==============================] - 0s 308us/step - loss: 0.2360 -
accuracy: 0.9104
Epoch 98/100
27/27 [==============================] - 0s 290us/step - loss: 0.2352 -
accuracy: 0.9151
Epoch 99/100
27/27 [==============================] - 0s 316us/step - loss: 0.2341 -
accuracy: 0.9198
Epoch 100/100
```

```
27/27 [==============================] - 0s 321us/step - loss: 0.2323 -
accuracy: 0.9151
```

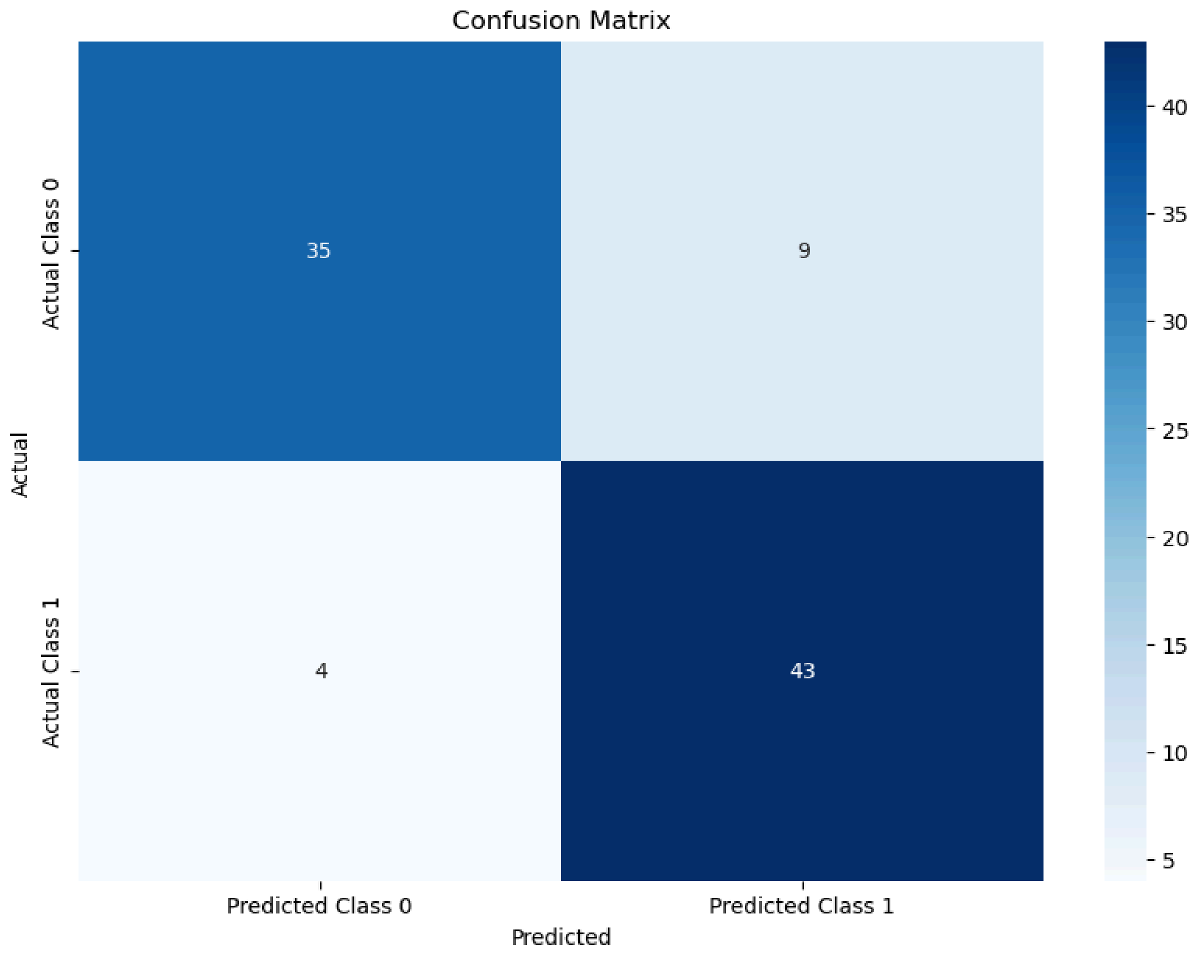
Out[7]:
```
<keras.src.callbacks.History at 0x304132c90>
```

In [8]:
```python
# Perform predictions on the test set
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)  # Convert probabilities to binary predictions
```

```
3/3 [==============================] - 0s 703us/step
```

In [9]:
```python
# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# Plot the confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicte
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
[[35  9]
 [ 4 43]]
```



In [10]:
```
# Calculate accuracy
accuracy = (cm[0][0] + cm[1][1]) / (cm[0][1] + cm[1][0] + cm[0][0] + cm[
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

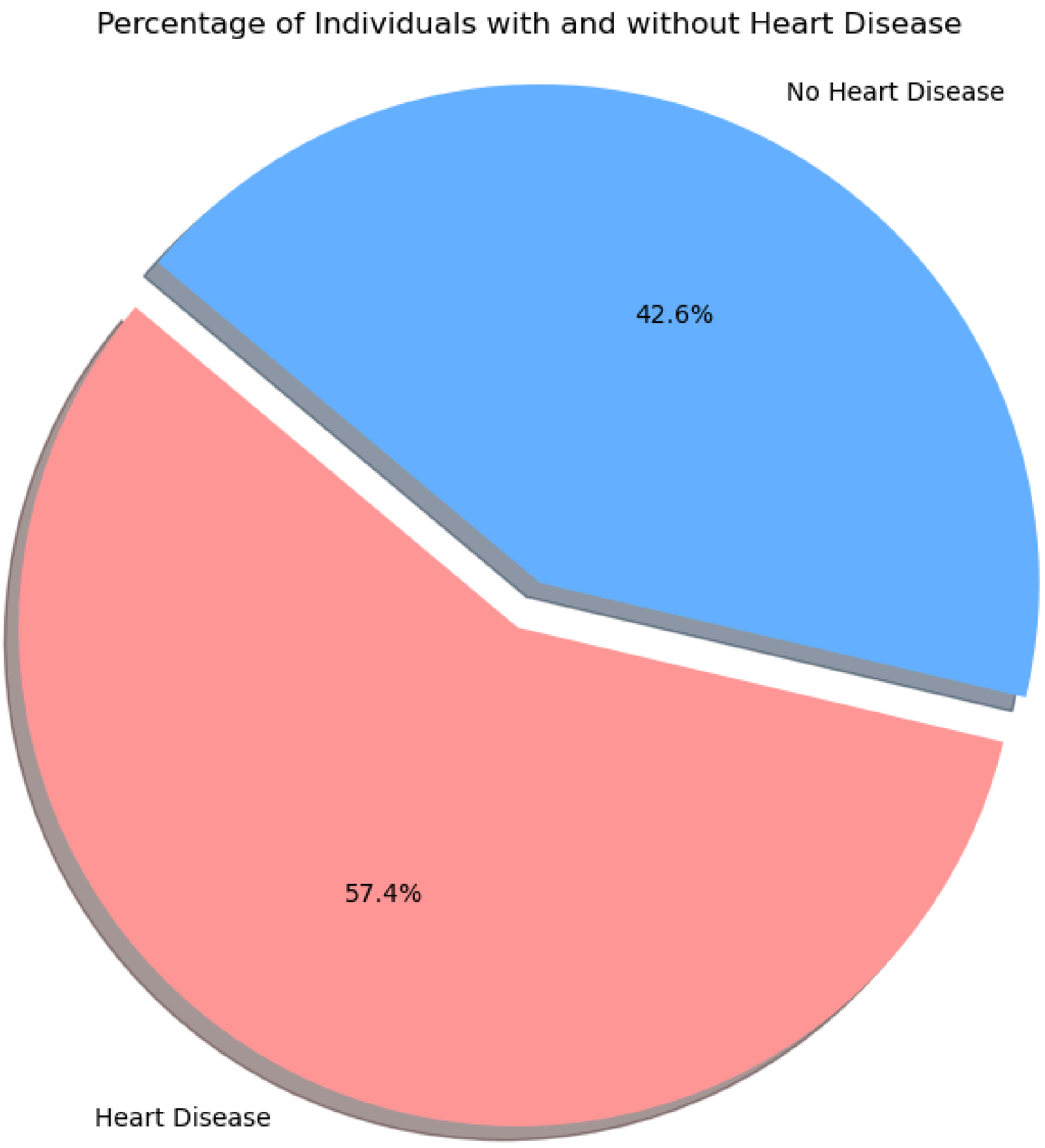
```
Accuracy: 85.71%
```

In [11]:
```
#Checking the first 10 predictions and providing interpretation
for i in range(10):
    prediction = 'Heart Disease' if y_pred[i] else 'No Heart Disease'
    print(f"Person {i+1}: {prediction}")
    if y_pred[i]:
        print("The model predicts that this person has heart disease. Gi
    else:
        print("The model predicts that this person does not have heart d
```

```
Person 1: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 2: Heart Disease
The model predicts that this person has heart disease. Given the mode
l's accuracy, there is a high probability that this prediction is corre
ct. It is advisable to consult with a healthcare professional for a com
prehensive evaluation.
Person 3: Heart Disease
The model predicts that this person has heart disease. Given the mode
l's accuracy, there is a high probability that this prediction is corre
ct. It is advisable to consult with a healthcare professional for a com
prehensive evaluation.
Person 4: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 5: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 6: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 7: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 8: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 9: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
Person 10: No Heart Disease
The model predicts that this person does not have heart disease. The in
dividual is likely healthy with respect to heart disease, but regular c
heck-ups and monitoring are recommended to ensure ongoing heart health.
```

In [12]:
```
# Predict for the entire dataset (training + test set)
X_full = np.concatenate((X_train, X_test), axis=0)
y_pred_full = classifier.predict(X_full)
y_pred_full = (y_pred_full > 0.5)  # Convert probabilities to binary pre

# Calculate the percentage of predictions
total_count = len(y_pred_full)
heart_disease_count = np.sum(y_pred_full)
no_heart_disease_count = total_count - heart_disease_count

heart_disease_percentage = (heart_disease_count / total_count) * 100
no_heart_disease_percentage = (no_heart_disease_count / total_count) * 1

print(f"Percentage of individuals predicted to have heart disease: {hear
print(f"Percentage of individuals predicted not to have heart disease: {

# Plotting the percentages
labels = ['Heart Disease', 'No Heart Disease']
sizes = [heart_disease_percentage, no_heart_disease_percentage]
colors = ['#ff9999','#66b3ff']
explode = (0.1, 0)  # explode the first slice (Heart Disease)

plt.figure(figsize=(8, 8))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a c
plt.title('Percentage of Individuals with and without Heart Disease')
plt.show()
```

```
10/10 [==============================] - 0s 350us/step
Percentage of individuals predicted to have heart disease: 57.43%
Percentage of individuals predicted not to have heart disease: 42.57%
```


Percentage of Individuals with and without Heart Disease
No Heart Disease
42.6%
57.4%
Heart Disease

```
In [ ]:
```