Step 1: Import Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
import pickle
```

Step 2: Load Dataset

```
df = pd.read_csv('/content/drive/MyDrive/Heart Disease -  MITM/heart.csv')  # Adjust the path if needed
df.head()
```

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Step 3: Data Overview

```
print("Shape:", df.shape)
print("\nMissing values:\n", df.isnull().sum())
print("\nData Types:\n", df.dtypes)
df.describe()
```

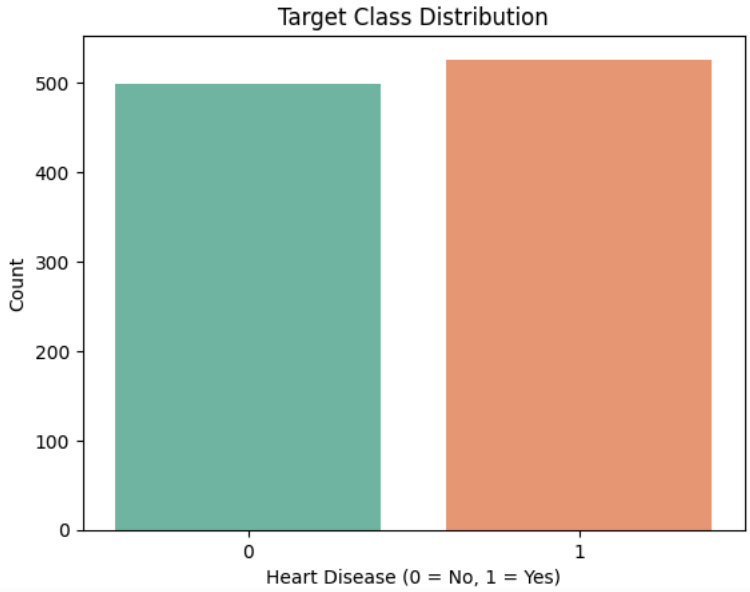
```
Shape: (1025, 14)

Missing values:
 age         0
gender       0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64

Data Types:
 age           int64
gender         int64
cp             int64
trestbps       int64
chol           int64
fbs            int64
restecg        int64
thalach        int64
exang          int64
oldpeak      float64
slope          int64
ca             int64
thal           int64
target         int64
dtype: object
```

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000

Step 4: Target Distribution

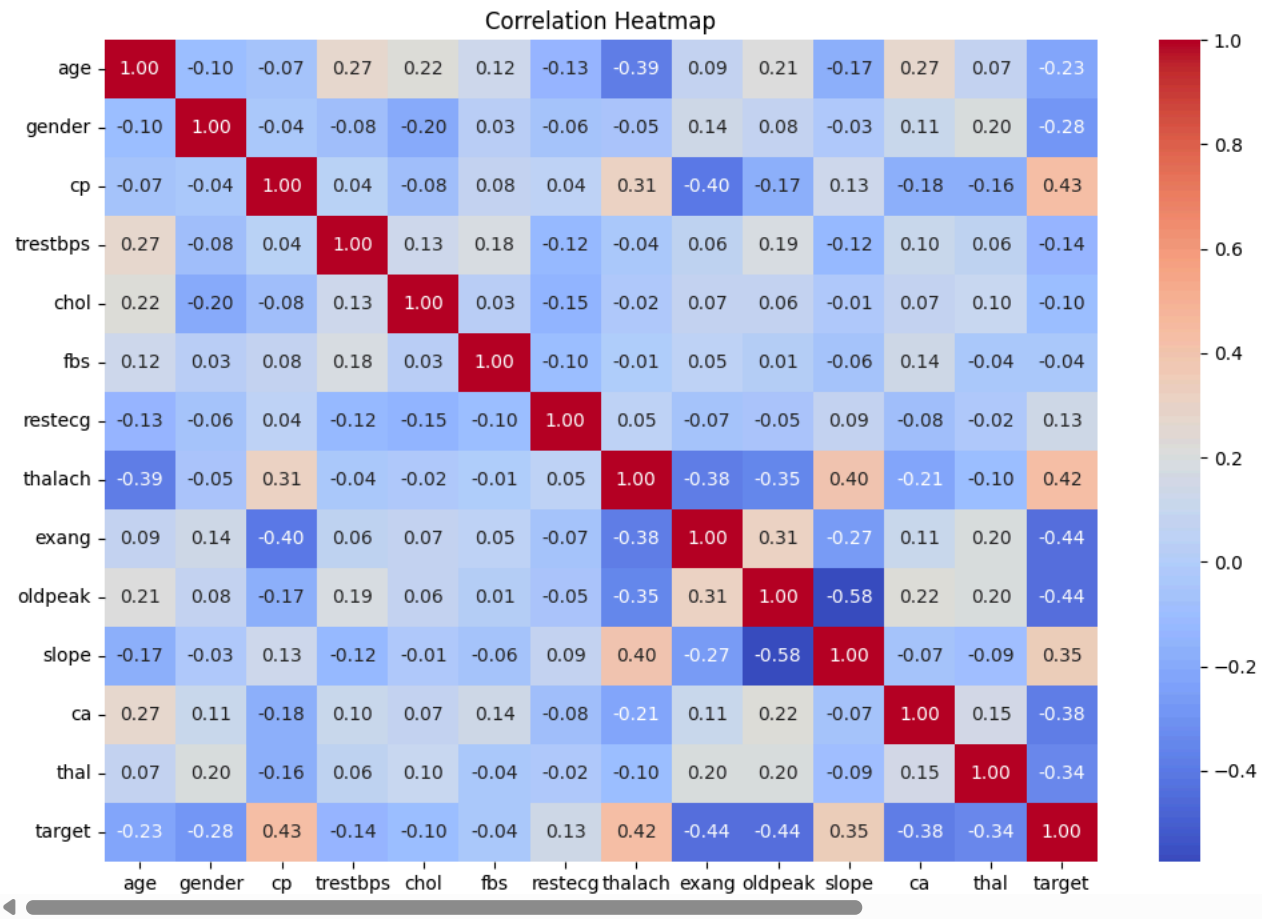
```
sns.countplot(data=df, x='target', palette='Set2')
plt.title("Target Class Distribution")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-4-6243a7e33201>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data=df, x='target', palette='Set2')
```



Step 5: Correlation Heatmap

```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



Step 6: Feature and Label Split

```
X = df.drop('target', axis=1)
y = df['target']
```

Step 7: Train-Test Split (80-20)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 8: Standardisation

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Convert back to DataFrame to retain feature names
X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)
```

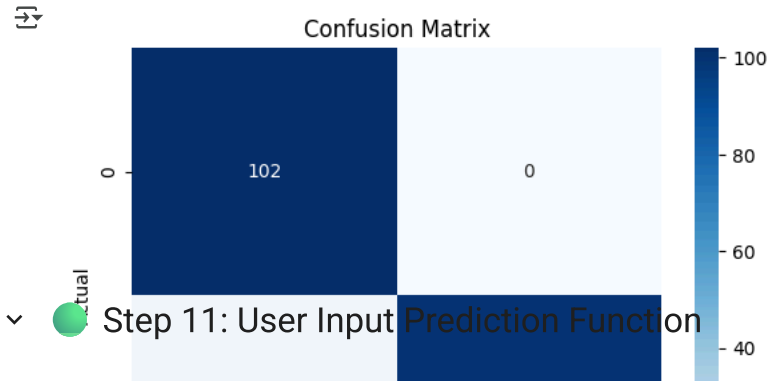
Step 9: Model Training - Random Forest

```
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Step 10: Model Evaluation

```
y_pred = model.predict(X_test)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Classification Report and Accuracy
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```



Step 11: User Input Prediction Function

```
def predict_from_input():
    print("\n🔍 Please enter the following patient details:\n")

    user_data = {}

    user_data['age'] = float(input("1. Age (in years): "))
    user_data['gender'] = float(input("2. Gender (1 = Male, 0 = Female): "))
    user_data['cp'] = float(input("3. Chest Pain Type (0 = Typical Angina, 1 = Atypical Angina, 2 = Non-anginal Pain, 3 = Asymptomatic)
    user_data['trestbps'] = float(input("4. Resting Blood Pressure (in mm Hg): "))
    user_data['chol'] = float(input("5. Serum Cholesterol (in mg/dl): "))
    user_data['fbs'] = float(input("6. Fasting Blood Sugar > 120 mg/dl (1 = Yes, 0 = No): "))
    user_data['restecg'] = float(input("7. Resting ECG Results (0 = Normal, 1 = ST-T Abnormality, 2 = Left Ventricular Hypertrophy): ")
    user_data['thalach'] = float(input("8. Maximum Heart Rate Achieved: "))
    user_data['exang'] = float(input("9. Exercise Induced Angina (1 = Yes, 0 = No): "))
    user_data['oldpeak'] = float(input("10. ST Depression (oldpeak) induced by exercise: "))
    user_data['slope'] = float(input("11. Slope of the ST Segment (0 = Upsloping, 1 = Flat, 2 = Downsloping): "))
    user_data['ca'] = float(input("12. Number of Major Vessels (0 to 4) colored by fluoroscopy: "))
    user_data['thal'] = float(input("13. Thalassemia (0 = Unknown, 1 = Normal, 2 = Fixed defect, 3 = Reversible defect): "))

    # Convert to DataFrame
    input_df = pd.DataFrame([user_data])
    prediction = model.predict(input_df)[0]

    print("\n📋 Prediction Result:")
    if prediction == 1:
        print("🔴 High Risk: The patient is likely to have heart disease.")
    else:
        print("🟢 Low Risk: The patient is not likely to have heart disease.")

# Example call
predict_from_input()
```