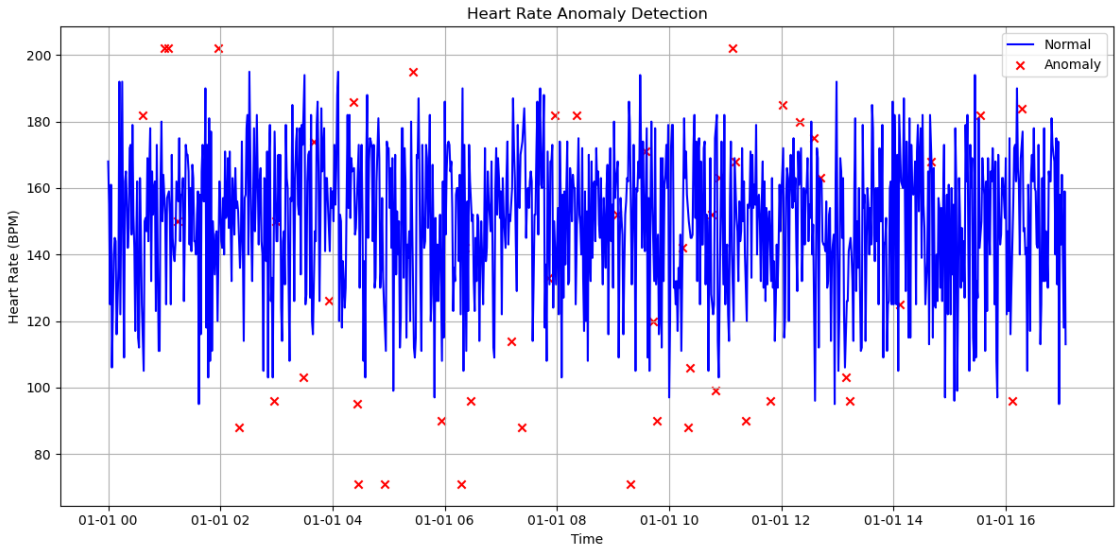
untitled2

August 1, 2025

[18]:
```
#  Imports
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import gradio as gr

#  Load dataset
df = pd.read_csv('heart.csv')

#  Rename for clarity
df.rename(columns={'thalach': 'heart_rate'}, inplace=True)

#  Simulate timestamp (1-minute interval)
df['timestamp'] = pd.date_range(start='2023-01-01', periods=len(df), freq='min')

#  Simulate SpO  feature
np.random.seed(42)  # for reproducibility
df['spo2'] = np.random.normal(loc=97, scale=1.0, size=len(df))  # normal range␣
 ↪around 96-99%

#  Clean & fill missing values
df.ffill(inplace=True)

#  Scale features
scaler = StandardScaler()
df[['heart_rate_scaled', 'spo2_scaled']] = scaler.
 ↪fit_transform(df[['heart_rate', 'spo2']])

#  Train Isolation Forest on both features
model = IsolationForest(contamination=0.05, random_state=42)
df['anomaly'] = model.fit_predict(df[['heart_rate_scaled', 'spo2_scaled']])
df['anomaly'] = df['anomaly'].map({1: 0, -1: 1})  # 0 = normal, 1 = anomaly

#  Plot anomalies
plt.figure(figsize=(12, 6))
```

```
normal = df[df['anomaly'] == 0]
anomalies = df[df['anomaly'] == 1]

plt.plot(normal['timestamp'], normal['heart_rate'], label='Normal',␣
 ↪color='blue')
plt.scatter(anomalies['timestamp'], anomalies['heart_rate'], color='red',␣
 ↪label='Anomaly', marker='x')
plt.xlabel('Time')
plt.ylabel('Heart Rate (BPM)')
plt.title('Heart Rate Anomaly Detection')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

#  Gradio App: Real-time anomaly check
def check_anomaly(heart_rate, spo2):
    sample = pd.DataFrame([[heart_rate, spo2]], columns=['heart_rate', 'spo2'])
    sample_scaled = scaler.transform(sample)
    prediction = model.predict(sample_scaled)[0]
    return "  Anomaly Detected" if prediction == -1 else "  Normal"

#  Launch Gradio Interface
gr.Interface(
    fn=check_anomaly,
    inputs=[
        gr.Slider(50, 200, step=1, label="Heart Rate (BPM)"),
        gr.Slider(90, 100, step=0.1, label="SpO  (%)"),
    ],
    outputs="text",
    title="Health Monitoring - Anomaly Detector",
    live=True
).launch(inline=True)
```




```
* Running on local URL:  http://127.0.0.1:7861
* To create a public link, set `share=True` in `launch()`.
```

```
<IPython.core.display.HTML object>
```

[18]:

[ ]: