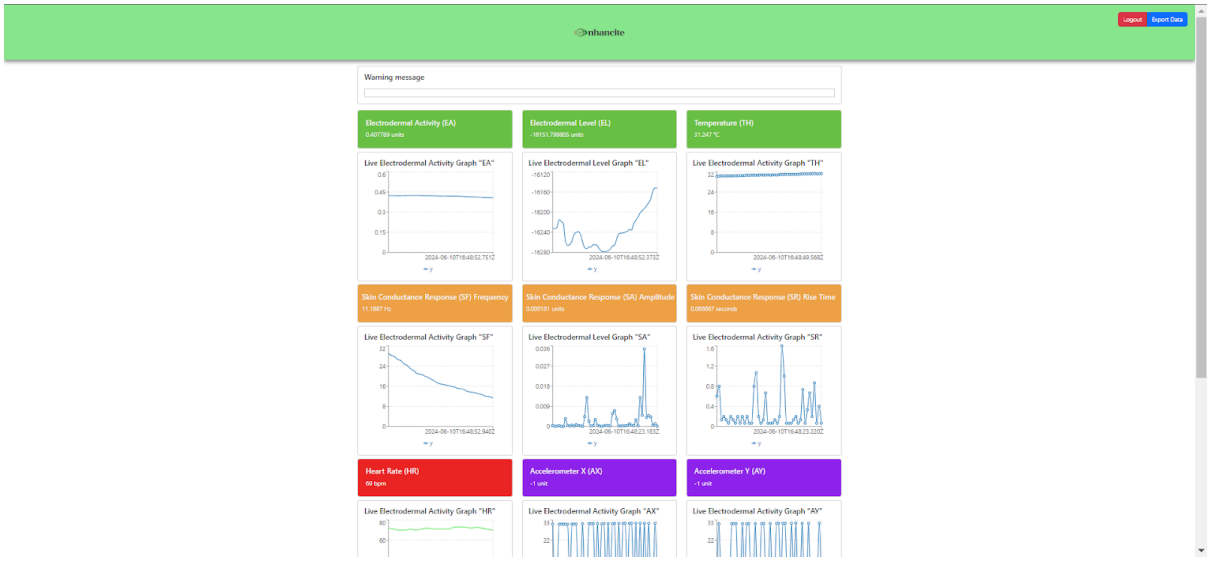
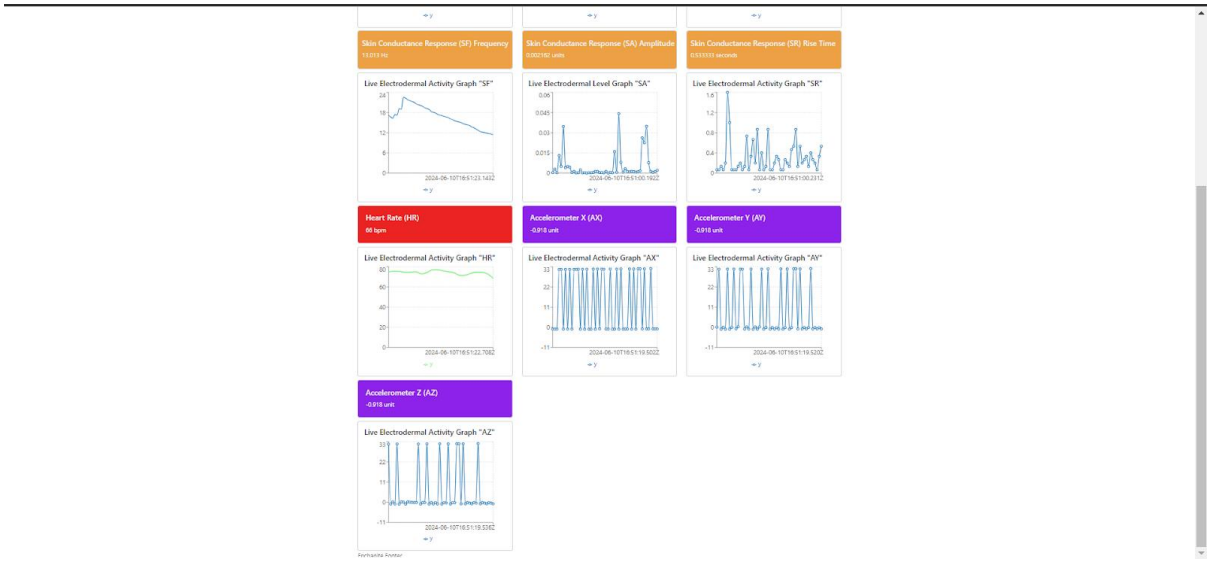
User Story: As a clinician, I want to see other biometric data, so that I can better treat my patients. (5 points)

User Story: As a clinician, I want to see patient data appear on the dashboard in real time, so I can respond to my patients' reactions quickly. (8 points).

Real-time graphs of patient data on ClincDashBoard page

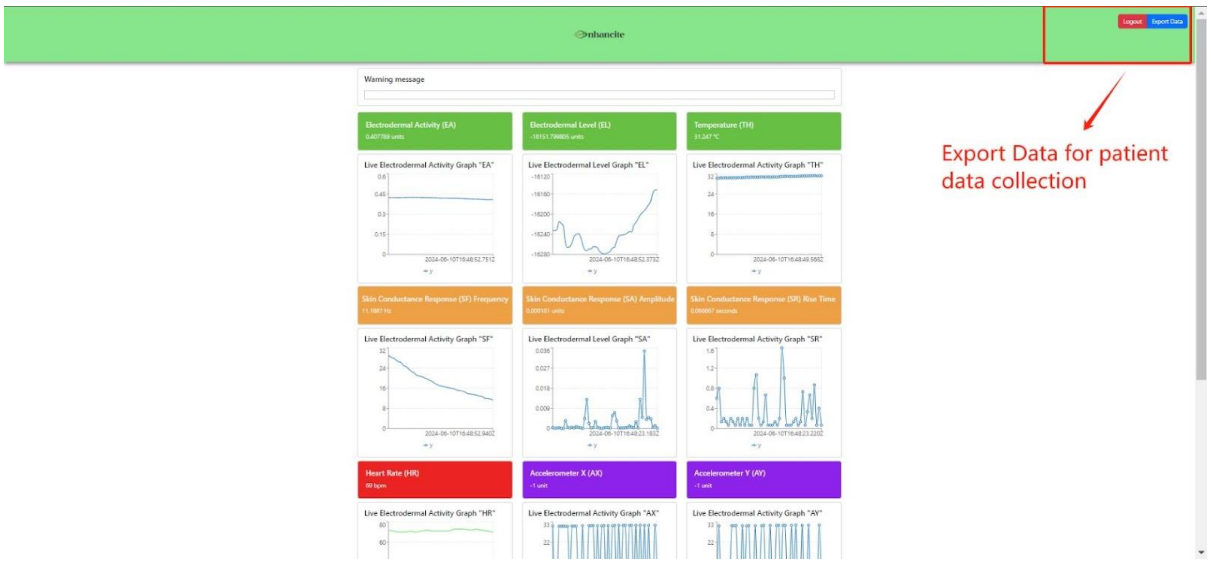
We implemented real-time graphs of 10 data streams on ClinicDashboard page, Clinician can monitor patient data in detail.





User Story: As a clinician, I want to export patient data, so that other people can use the data. (5 points)

Press the Export Data button to collect patient data to the backend



Patient data received by backend

id	date_time_recorded	patientid	sessionid	value
26	2024-04-27 12:31:01.379000	1	1714221051236076000	0.8187
27	2024-04-27 12:31:01.628000	1	1714221051236076000	0.7999
28	2024-04-27 12:31:01.703000	1	1714221051236076000	0.7815
29	2024-04-27 12:31:01.805000	1	1714221051236076000	0.7724
30	2024-04-27 12:31:01.903000	1	1714221051236076000	0.7547
31	2024-04-27 12:31:02.030000	1	1714221051236076000	0.7459
32	2024-04-27 12:31:02.113000	1	1714221051236076000	0.7373
33	2024-04-27 12:31:02.208000	1	1714221051236076000	0.7288
34	2024-04-27 12:31:02.295000	1	1714221051236076000	0.7203
NULL	NULL	NULL	NULL	NULL

User Story: As a clinician, I want to be alerted when a patient's vitals exceed threshold, so that I do not overexpose my patients. (5 points)

Heart rate alert system

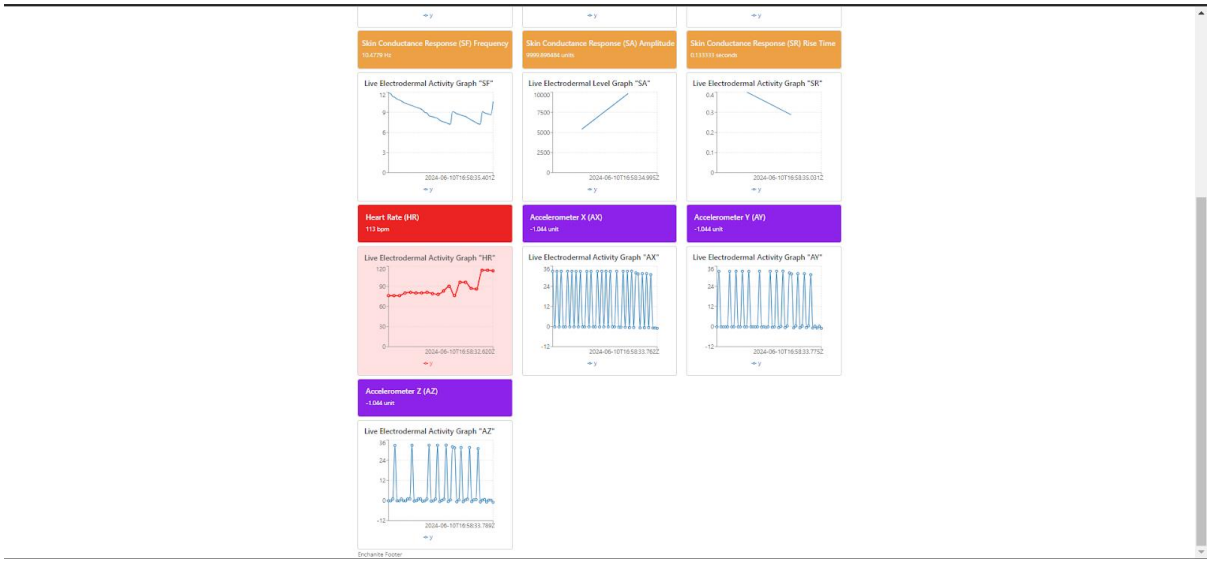
If the heart rate exceeds 100 beats per minute, the line turns red. Conversely, if the heart rate drops below 50 beats per minute, the line turns blue.

In addition to the line color, we have implemented a blinking effect to further emphasize abnormal heart rate values:

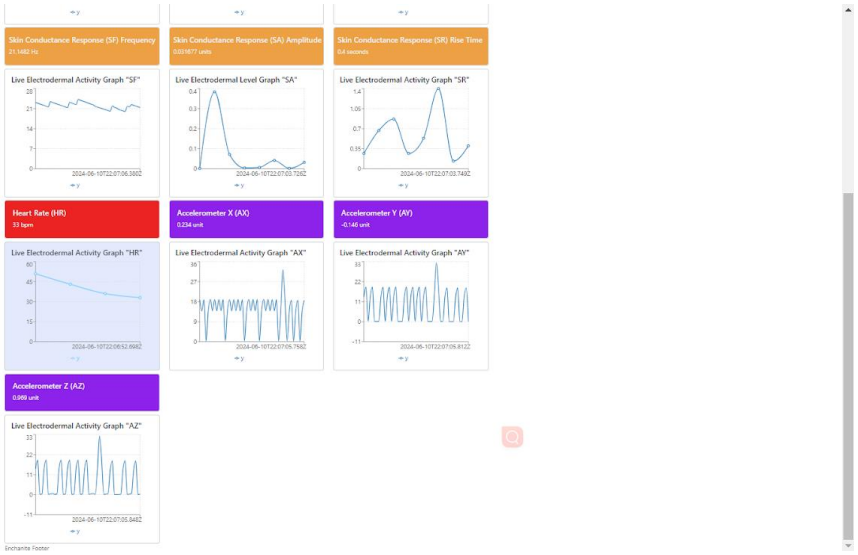
When the heart rate exceeds 100 beats per minute, the heart rate display blinks in red. Similarly, if the heart rate falls below 50 beats per minute, the display blinks in blue.

We also made an abnormal heart rate recorder, when abnormal heart rate appeared, the recorder will record the abnormal heart rate and the Time of occurrence, and show the message through message box

High heart rate alert:



Low Heart rate:



Abnormal Heart rate recorder:

User Story: As a clinician, I want to have the sensor automatically connect to Enhancite without any third-party programs, so that the program is easy to use. (8 points)

To automatically connect to Enhencite, we made a udpport.py python program to receive data automatically sent by Emotibit device from UDP port, and we made a UdpPortControl.py python program to control udpport.py. But Users still need to use the EmotiBit Oscilloscope to send data to the UDP port.

udpport.py:

udpport.py > udp_to_websocket

```python
import socket
import asyncio
import websockets
websocket_connections = set()
async def udp_to_websocket():
    udp_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    udp_sock.bind(('0.0.0.0', 12346))
    udp_sock.setblocking(False)  # Make the socket non-blocking

    # Start the WebSocket server
    async with websockets.serve(handler, "localhost", 8765,
ping_timeout=30, close_timeout=10):
        print("WebSocket server started...")
        while True:
            await asyncio.sleep(0.016)  # Small delay to yield control to the loop
            try:
                data, addr = udp_sock.recvfrom(2048)
                print(f"Received UDP data: {data} from {addr}")
                await forward_data(data)
            except BlockingIOError:
                continue  # No data available, continue the loop
            except Exception as e:
                print(f"Unexpected error: {e}")  # Log unexpected errors
async def handler(websocket, path):
   websocket_connections.add(websocket)
   try:
       await websocket.wait_closed()
   finally:
       websocket_connections.remove(websocket)
async def forward_data(data):
    # Convert data to string
    data_str = data.decode()
    # Create a list of websockets that failed to receive the data
    failed_websockets = set()
    # Create a copy of the set for iteration to prevent modification errors
    connections = set(websocket_connections)
    for websocket in connections:
        if websocket.open:
            try:
                await websocket.send(data_str)
            except websockets.exceptions.ConnectionClosed:
                failed_websockets.add(websocket)
    # Remove failed websockets from the original set
    websocket_connections.difference_update(failed_websockets)

if __name__ == "__main__":
    asyncio.run(udp_to_websocket())

```

UdpPortControl.py:

UdpPortControl.py > ...

```python
from flask import Flask, jsonify
import subprocess
import os
import signal
import platform

# from flask_cors import CORS

app = Flask(__name__)

process = None

@app.route('/start-udp', methods=['POST'])
def start_udp():
    global process
    if process is None:
        if platform.system() == 'Windows':
            # you can input your own python.exe path to replace python, make sure the path is correct
            process = subprocess.Popen(['c:/Users/jayye999/Desktop/PP1/Enhancite/.venv/Scripts/python.exe', './udpport.py'], creationflags=subprocess.CREATE_NEW_PROCESS_GROUP)
        else:
            # you can input your own python.exe path to replace python, make sure the path is correct
            process = subprocess.Popen(['c:/Users/jayye999/Desktop/PP1/Enhancite/.venv/Scripts/python.exe', './udpport.py'], preexec_fn=os.setsid)
        return jsonify({'message': 'UDP script started'}), 200
    return jsonify({'message': 'UDP script already running'}), 400

@app.route('/stop-udp', methods=['POST'])
def stop_udp():
    global process
    if process:
        if platform.system() == 'Windows':
            os.kill(process.pid, signal.CTRL_BREAK_EVENT)
        else:
            os.killpg(os.getpgid(process.pid), signal.SIGTERM)
        process = None
        return jsonify({'message': 'UDP script stopped'}), 200
    return jsonify({'message': 'No UDP script running'}), 400


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```