Assignment 4
Distributive App

TCP Connection with GUI (Patient Monitor)

Ahmed Gehad Mohamed Aly - 1200387
Nouran Mohamed - 1210087

1. Client (Sender)

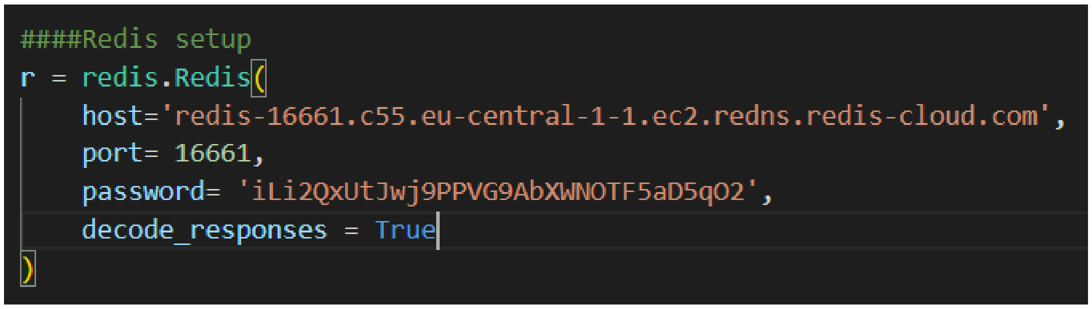
Client sends patient's live vital signs to the server using TCP connection:

- Make sure the **DEST_IP = "Ahmed"** is set to your pc local host
- First, we connect client socket to the server specified by the destination IP and port.
- Patient class with attributes such as id, name, age, heart rate, systolic blood pressure, and diastolic blood pressure is created.
- **def generate_vital_signs(self):** this function generates random vital signs for a patient: heart rate between 60 and 100 beats per minute, systolic blood pressure between 90 and 120 mmHg, and diastolic blood pressure between 60 and 80 mmHg.
- **def get_vital_signs(self):** this function returns a dictionary containing the patient's vital signs in JSON format.
- Then we create instances of the Patient class, specifying their id, name, and age.
- **def client_send():** sends the vital signs of patients to the server in a loop. For each patient, it generates vital signs, converts them to JSON format, encodes the JSON message into bytes, and sends it via the client socket. Then it sleeps for 2 seconds before sending the next set of vital signs.

2. Server (Receives)

Receives vital signs data from clients, stores it in a Redis database:

- Make sure **HOST_IP = "Ahmed"** is set to your pc local host
- Creates a server-side socket and bind the server socket to the specified IP address and port, and then start listening for incoming connections.
- Create an online Redis database then connect the Redis server.
- Set **decode_responses** to True to automatically decode responses to **strings**.



```
####Redis setup
r = redis.Redis(
    host='redis-16661.c55.eu-central-1-1.ec2.redns.redis-cloud.com',
    port= 16661,
    password= 'iLi2QxUtJwj9PPVG9AbXWNOTF5aD5qO2',
    decode_responses = True
)
```

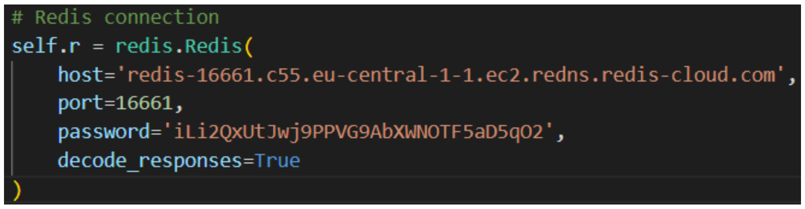
- Setup the Redis connection with the online cloud database you made.

- **def handle_client(client):** This function is called when a client connects. It receives data from the client, decodes it as JSON, then stores the vital signs data in Redis under a unique key.
- Main Server Loop: When a client connects, it accepts the connection and then starts handling the client by calling the **handle_client** function.
- Overall, we set up a server that listens for incoming connections, receives JSON-formatted vital signs data from clients and stores it in a Redis database.

3. Qt Creator GUI

Display with buttons and drop-down menu showing a live plotting graph:

- Class **LivePlotWidget** inherits from FigureCanvas. It's used to display live plots of vital signs data.
- **FigureCanvas** is a Matplotlib class that provides a drawing area for a Matplotlib figure. It is used to embed Matplotlib plots into the (Qt application)
- **super().__init__(self.figure)** Calls the constructor of the parent class **(FigureCanvas)** passing the **self.figure** object.
- Adds a subplot to the figure, initializes a deque **x_data** to store x-axis data, and a counter **x_counter** for time.
- Depending on the value of **index**, it sets up either a heart rate plot or a blood pressure plot.
- Again we make the Redis online database connection, make sure to change it accordingly to the Redis data base you created.



```
# Redis connection
self.r = redis.Redis(
    host='redis-16661.c55.eu-central-1-1.ec2.redns.redis-cloud.com',
    port=16661,
    password='iLi2QxUtJwj9PPVG9AbXWNOTF5aD5qO2',
    decode_responses=True
)
```

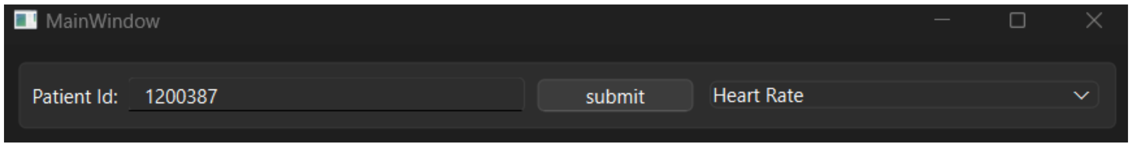
- **def fetch_from_redis(self, key):** fetches data from Redis using the provided key.
- **def update_plot_h(self):** updates the heart rate plot with data fetched from Redis.
- **def update_plot_b(self):** updates the blood pressure plot with data fetched from Redis.
- Initialize the user interface **window self.ui = Ui_MainWindow() self.ui.setupUi(self)**
- Then we handle the button and combo box connections when clicked with the corresponding functions
- **def handle_Selector_Method(self):** handles the combo box index change event and switches between different plots based on the selected index.

- **def id_submit_VER2(self):** handles the submission of the patient ID and updates the corresponding plot widget accordingly.

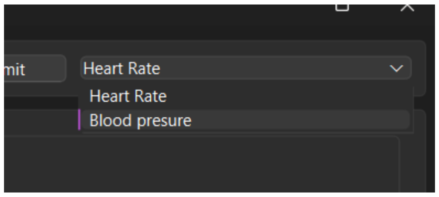
4. User interface Display

Steps on how to use the patient monitor Qt application effectively:

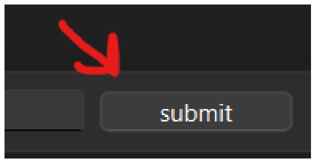
- Make sure to insert the correct patient **id**.



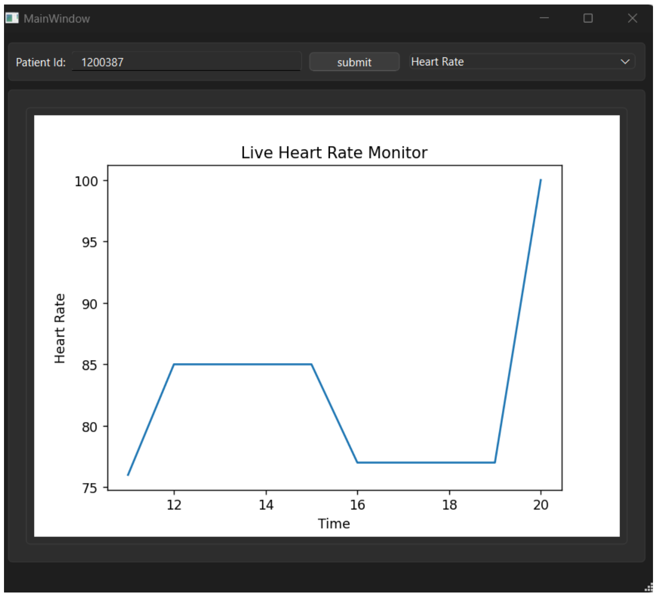
- Select from the **combo box** which vital sign u would like to view either heart rate or blood pressure.



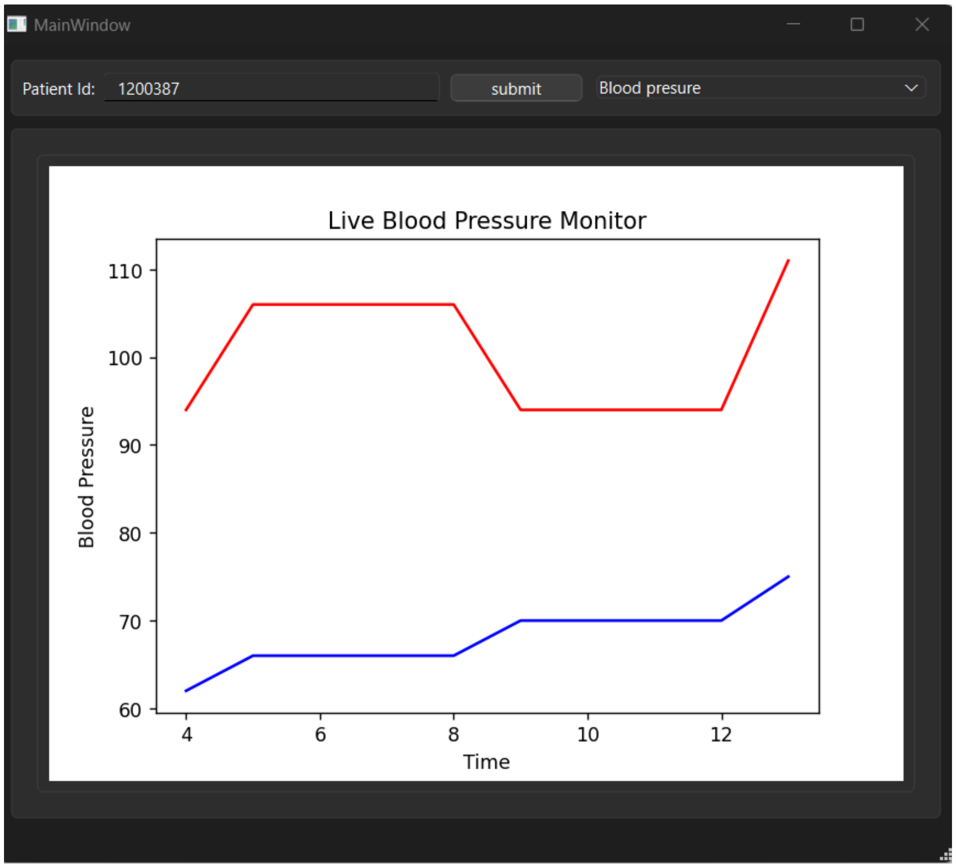
- Press on the **submit** button.



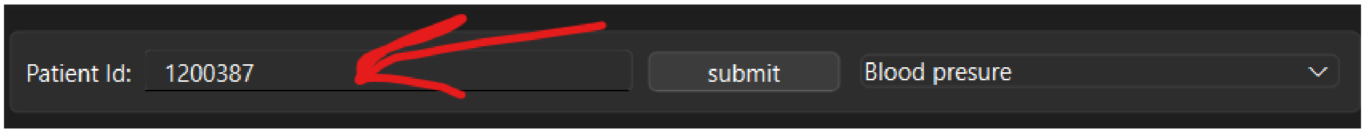
- A live graph will be plotted showing the corresponding patient vital sign.



- Note you can alternate between the 2 vital signs with ease heart rate or blood pressure using the combo box.



- For viewing another patient change the id and repeat the process again



5. References

- https://www.youtube.com/watch?v=Nl5om8Vl85Y&feature=youtu.be
- https://www.youtube.com/watch?v=GxnWPY9GCrw
- https://youtu.be/AHhcwFPQlfQ?si=OIj1vR2cyY493mSN
- https://www.youtube.com/watch?v=Lbfe3-v7yE0&t=3s