CÁCH HOẠT ĐỘNG CỦA FILE `Heatmap.js`

Note: Cần đọc file này để biết được cách thức hoạt động của socket và get data realtime

1 : Khi vào trang web, react gọi hàm `componentDidMount()` trong file `Heatmap.js` để kết nối socket

- Vì socket là 1 biến toàn cục nên khi kết nối socket, nó sẽ được lưu vào state `locationData` trên redux store với tên gọi là `state.iotSocket`

```
socketInstance.socket.on('connect', () => {
  socketInstance.socket.emit('connect_mqtt', { id: 'bandata' })
  dispatch(setSocketConnected(true))
})
```

Lúc này socket đã emit và connect đến server mqtt, để get data realtime từ server mqtt ta cần `socket.on` để lắng nghe đúng event mà server mqtt emit ra

```
socketInstance.socket.on('MQTT_LOCATION_RESPONSE', (data) => {
  dispatch(setLocationData(data))
})
```

Sự kiện `MQTT_LOCATION_RESPONSE` là sự kiện mà server mqtt emit ra khi có data mới, khi có data mới thì ta sẽ dispatch action `setLocationData` để lưu data mới vào redux store

Note: Tuy nhiên vì 1 lí do nào đấy mà clean up function ở đây lại không hoạt động như chúng ta mong muốn, nên mình đã tạo ra 1 class để quản lí tốt hơn

- Đầu tiên khai báo class SocketClass với 1 biến socket đúng type của socket.io-client. Sau đấy khởi tạo biến socket trong constructor của class
    - Hàm `createNewSocket()` sẽ tạo ra 1 socket mới và trả về socket đó
    - Hàm `close()` sẽ đóng socket hiện tại và tạo ra 1 socket mới nếu tham số truyền vào là true. Như mình đã nói ở trên thì `clean up socket.close()` có vẻ như nó sẽ phải chờ 1 khoảng trước khi nó hoạt động tiếp.
        - callback trong useEffect chạy lần 1 , lắng nghe socket(có vẻ nó cần thời gian)
        - useEffect chạy lần 2, socket emit data, nhưng lúc này socket bị close rồi nên nó không nhận được data
        - Nên khi tạo cái class này, mình sẽ tạo ra 1 socket mới khi socket cũ bị close. Nhưng để quản lí tốt hơn nên mình có để nó nhận 1 đối số với trạng thái mặc định là true, nếu muốn tắt socket thì truyền vào false

```
export class SocketClass {
  socket: Socket
  constructor() {
    this.socket = this.createNewSocket()
```

```
    }
    createNewSocket() {
      return io(url, {
        transports: ['websocket'],
        reconnection: false
      })
    }
    close(createNew: boolean = true) {
      this.socket.close()
      if (createNew) {
        this.socket = this.createNewSocket()
      }
    }
  }
  export const socketInstance = new SocketClass()
```