Front End Documentation

app.component.ts

- **Purpose**: This is the root component for the application. It is responsible for displaying the overall structure and layout of the application.
- **Functionality**: The AppComponent serves as the container for the various views within the app. It uses the router-outlet to dynamically render different components based on the current route. It also subscribes to notifications (if implemented).

app.component.html

- **Purpose**: The HTML template for the root component.
- **Functionality**: Contains the base layout of the application. It uses <router-outlet> to switch between the different pages or views in the application.

app-routing.module.ts

- **Purpose**: Defines the routes for navigating between different pages in the application.
- **Functionality**: Maps URLs to their respective components. For example, it routes /thresholds to the ThresholdsComponent and /alerts to the AlertComponent.

app.module.ts

- **Purpose**: The main module for the Angular application.
- **Functionality**: Imports and declares all components and services. It sets up routing, HTTP communication, and form handling within the app. It serves as the entry point for Angular to bootstrap the app.

---

alert.component.ts

- **Purpose**: Displays the list of alerts triggered by patient health data that exceeds or falls below the set thresholds.
- **Functionality**: It regularly checks for unreviewed alerts and displays them. Alerts can be marked as "reviewed" by the physician. It fetches alerts from the backend and provides UI interactions for the physician to manage alerts.

alert.component.html

- **Purpose**: The HTML template for the AlertComponent.
- **Functionality**: Displays a list of unreviewed alerts with an option to mark them as reviewed. Each alert includes details like the patient ID, health metric, and the out-of-range value.

---

mock-data.component.ts

- **Purpose**: Simulates patient data and displays it on the frontend.
- **Functionality**: Polls mock patient data from the backend every 5 seconds. The mock data includes heart rate, blood pressure, and temperature. The component displays these values and allows the user to change the patient ID for which data is being simulated.

mock-data.component.html

- **Purpose**: The HTML template for the MockDataComponent.
- **Functionality**: Displays the patient's heart rate, blood pressure, and temperature as mock data. It also includes an input field for changing the patient ID, which is used to simulate data for a different patient.

---

thresholds.component.ts

- **Purpose**: Allows the physician to set threshold values for patient health metrics like heart rate, blood pressure, and temperature.
- **Functionality**: It collects user input for minimum and maximum values of health metrics and sends these values to the backend. It validates the input to ensure that the maximum values are greater than the minimum values and that the patient ID is valid.

thresholds.component.html

- **Purpose**: The HTML template for the ThresholdsComponent.
- **Functionality**: Provides a form for the user to input the patient ID and the min/max thresholds for heart rate, blood pressure, and temperature. It displays validation messages when incorrect values are entered and submits the form to the backend for processing.

---

alert.service.ts

- **Purpose**: Handles communication between the frontend and the backend for managing alerts.
- **Functionality**: Provides methods to:
  - Fetch unreviewed alerts from the backend.
  - Mark alerts as reviewed.
  - Fetch mock data from the backend (for simulation purposes).

---

thresholds.service.ts

- **Purpose**: Handles communication between the frontend and the backend for setting thresholds.
- **Functionality**: Provides methods to:
  - Send threshold data (heart rate, blood pressure, temperature) to the backend for a specific patient.
  - Fetch threshold data for a particular patient (if needed).

Frontend Configuration and Running

Prerequisites:

- **Node.js**: Ensure Node.js and npm (Node Package Manager) are installed.
- **Angular CLI**: Ensure the Angular CLI is installed globally.

```
npm install -g @angular/cli
```

Steps:

1. **Clone or Download the Project**:
   - Make sure you have the frontend files on your local machine.
2. **Install Dependencies**:
   - Navigate to your frontend project directory and install the required npm dependencies:

```
cd your-angular-project-directory
npm install
```

3. **Run the Frontend Application**:
   - Start the Angular development server using the following command:

```
ng serve
```

   - The application will be accessible at http://localhost:4200/ in your browser