Backend API Documentation

Overview

This backend API provides functionality for a health monitoring system that allows physicians to set thresholds for key health metrics (e.g., heart rate, blood pressure, temperature), monitor real-time data, and trigger alerts when patient data falls outside the thresholds.

Base URL

http://localhost:8080/api

Endpoints

---

1. Alert Management

1.1 Get Alerts

- **Endpoint**: /alerts
- **Method**: GET
- **Description**: Retrieve all unreviewed alerts.
- **Response**:
  - Returns an array of alert objects with patient IDs, metrics, and their values.

1.2 Mark Alert as Reviewed

- **Endpoint**: /alerts/{alertId}/review
- **Method**: PUT
- **Description**: Marks a specific alert as reviewed.
- **Parameters**:
  - alertId: The ID of the alert to be marked as reviewed.
- **Response**:
  - Status 200 OK on success.

---

2. Mock Data Management

2.1 Get Mock Data

- **Endpoint**: /mock-data
- **Method**: GET
- **Description**: Fetch the most recent mock patient data from the simulator.
- **Response**:
  - Returns the latest mock patient data including patient ID, heart rate, blood pressure, and temperature.

2.2 Set Patient ID

- **Endpoint**: /mockdata/set-patient-id
- **Method**: POST
- **Description**: Updates the patient ID for generating mock data.

- **Parameters**:
  - patientId: The patient ID to be set.
- **Response**:
  - Success message confirming the update.
  - **Error**: Returns a 404 if patient ID doesn't exist.

---

3. Threshold Management

3.1 Set Thresholds

- **Endpoint**: /thresholds
- **Method**: POST
- **Description**: Set thresholds for a patient's health metrics (heart rate, blood pressure, temperature).
- **Body**:

```
{
 "patientId": "17",
 "heartRate": { "min": 60, "max": 100 },
 "bloodPressure": { "min": 120, "max": 180 },
 "temperature": { "min": 36, "max": 38 }
}
```

- **Response**:
  - Status 200 OK with a success message.

3.2 Get Thresholds by Patient ID

- **Endpoint**: /thresholds/{patientId}
- **Method**: GET
- **Description**: Fetch the thresholds set for a specific patient.
- **Parameters**:
  - patientId: The ID of the patient to retrieve thresholds for.
- **Response**:
  - Returns the threshold object with patient ID, heart rate, blood pressure, and temperature ranges.

Error Handling

Resource Not Found Exception

- **Class**: ResourceNotFoundException.java
- **Description**: Handles cases where resources (like alerts, thresholds, or patient data) are not found in the database.

Model Descriptions

1. Alert Model

- **Class**: Alert.java

- **Description**: Represents the structure of an alert, including patient ID, metric name, metric value, and whether the alert has been reviewed.
- **Fields**:
    - id: Unique identifier for the alert.
    - patientId: The patient associated with the alert.
    - metricName: The name of the metric (e.g., heart rate, blood pressure).
    - metricValue: The value of the metric.
    - isReviewed: Boolean flag indicating whether the alert has been reviewed.

2. Threshold Model

- **Class**: Threshold.java
- **Description**: Represents the threshold settings for a patient's health metrics.
- **Fields**:
    - id: Unique identifier for the threshold.
    - patientId: The ID of the patient for whom thresholds are set.
    - heartRateMin: Minimum heart rate threshold.
    - heartRateMax: Maximum heart rate threshold.
    - bloodPressureMin: Minimum blood pressure threshold.
    - bloodPressureMax: Maximum blood pressure threshold.
    - temperatureMin: Minimum temperature threshold.
    - temperatureMax: Maximum temperature threshold.

Services Overview

1. AlertService.java

- **Description**: Service class responsible for managing alert operations such as creating, fetching, and updating alert records.
- **Methods**:
    - getLatestMockData (): Return the latest mock data generated by the simulator.
    - processPatientData(PatientDataDTO): Processes incoming patient data and generates alerts if the data exceeds the thresholds.
    - getUnreviewedAlerts(String patientId): Returns all unreviewed alerts for a specific patient.
    - markAlertAsReviewed(Long alertId): Marks an alert as reviewed.

2. PatientDataSimulator.java

- **Description**: Generates mock patient data periodically and sends it to the AlertService to process potential alerts.
- **Methods**:
    - simulatePatientData(): Simulates mock data at fixed intervals and triggers alert processing.
    - setPatientId(): Setter method to change the patient ID dynamically.

3. ThresholdService.java

- **Description**: Service class for managing threshold operations.
- **Methods**:
    - saveThresholds(ThresholdDto): Sets or updates threshold values for a patient.

- getThresholdsByPatientId(String patientId): Fetches the thresholds set for a specific patient.

Repository Interfaces

1. AlertRepository.java

- **Description**: Repository interface for managing the persistence and retrieval of alert records.
- **Methods**:
  - findByPatientIdAndIsReviewedFalse(String patientId): Retrieves all unreviewed alerts for a given patient.

2. ThresholdRepository.java

- **Description**: Repository interface for managing the persistence and retrieval of threshold settings.
- **Methods**:
  - findByPatientId(String patientId): Retrieves threshold settings for a specific patient.

DTO (Data Transfer Objects)

1. PatientDataDTO.java

- **Description**: DTO class used to transfer patient health data.
- **Fields**:
  - patientId: The ID of the patient.
  - heartRate: The current heart rate of the patient.
  - bloodPressure: The current blood pressure of the patient.
  - temperature: The current temperature of the patient.

2. ThresholdDto.java

- **Description**: DTO class used to transfer threshold data when setting or updating threshold values.
- **Fields**:
  - patientId: The ID of the patient.
  - heartRateMin: The minimum heart rate threshold.
  - heartRateMax: The maximum heart rate threshold.
  - bloodPressureMin: The minimum blood pressure threshold.
  - bloodPressureMax: The maximum blood pressure threshold.
  - temperatureMin: The minimum temperature threshold.
  - temperatureMax: The maximum temperature threshold.

Technologies Used

- **Spring Boot**: Backend framework.
- **Hibernate/JPA**: ORM framework for database interaction.
- **MySQL**: Database for storing patient data, thresholds, and alerts.
- **REST API**: For exposing endpoints to the frontend.

Setup Instructions

1. Clone the repository.
2. Navigate to the project root directory.
3. Update the application.properties file with your MySQL database connection details.
4. Run mvn clean install to build the project.
5. Use mvn spring-boot:run to start the backend server.
6. The API will be accessible at http://localhost:8080/api.