Smart Climate Control System – Project Proposal

Introduction

The aim of the Smart Climate Control System is to set a comfortable and energy efficient environment where the users will be able to manage temperature and humidity and know the air quality in their home. The system consists of three services that communicate and interact using gRPC: Thermostat, Humidity Controller and Air Quality Monitor. Users can monitor and control the system by using a simple GUI with buttons that invoke each gRPC communication style. The project will be implemented and tested using Java and NetBeans to showcase an example of gRPC full capabilities in real-world applications.

Application Domain

As part of a Smart Home System, this solution improves climate control by adjusting indoor conditions based on real-time data. It provides:

- **Comfort:** Temperature and humidity adjustments
- **Health and Safety:** Air quality monitoring with alert mechanisms
- **Efficiency:** Smart regulation of climate settings to optimize energy use

Services Overview

1. Thermostat Service

Purpose: Controls heating and cooling based on user preferences and outdoor temperature.

Functionalities:

- `SetTemperature()` : Adjusts the desired room temperature.
- `GetCurrentTemperature()` : Retrieves current temperature.
- `StreamTemperatureUpdates()` : Streams real-time temperature updates.
- `AutoAdjustMode()` : Enables automatic temperature adjustment based on real-time sensor data.

2. Humidity Controller Service

Purpose: Maintains optimal humidity levels to improve comfort and prevent issues like mold growth.

Functionalities:

- `SetHumidityLevel()`: Collects multiple humidity readings before adjusting.
- `GetCurrentHumidity()`: Retrieves current humidity level.
- `EnableAutoMode()`: Enables automatic humidity regulation.

3. Air Quality Monitor Service

Purpose: Monitors air quality and alerts users if pollutants exceed safe levels.

Functionalities:

- `GetAirQuality()`: Provides a real-time air quality status (e.g., "Good", "Moderate", "Poor").
- `MonitorAirQuality()`: Bi-directional streaming for real-time air quality alerts.

gRPC Communication

The system will use all four gRPC invocation styles:

- **Simple RPC:** `SetTemperature()` allows users to set a specific temperature.
- **Server Streaming:** `StreamTemperatureUpdates()` provides real-time temperature updates.
- **Client Streaming:** `SetHumidityLevel()` collects multiple humidity readings before making adjustments.
- **Bi-Directional Streaming:** `MonitorAirQuality()` enables real-time air quality alerts and responses.

GUI Controller

A simple GUI will be implemented using NetBeans to allow the user to interact with the application. It will contain buttons to trigger each of the gRPC invocation styles and a display to show temperature, humidity, air quality, alerts and errors.

Sample `.proto` Definitions

The following proto definitions will be used to create the application. Each one of them defines the gRPC services and the messages exchanged between the client and the server. Each service includes remote procedure calls (RPCs) that specify how clients interact with the system and messages to define the structure of the data exchanged.

1. Thermostat Service

Handles temperature adjustments, retrieves current temperature, streams real-time updates, and enables automatic adjustment.

```
syntax = "proto3";
service Thermostat {
  rpc SetTemperature(TemperatureRequest) returns (TemperatureResponse);
  rpc GetCurrentTemperature() returns (TemperatureResponse);
  rpc StreamTemperatureUpdates() returns (stream TemperatureResponse);
  rpc AutoAdjustMode(AutoAdjustRequest) returns (StatusResponse);
}
message TemperatureRequest {
  float temperature = 1;
}
message TemperatureResponse {
  float currentTemperature = 1;
}
message AutoAdjustRequest {
  bool enable = 1;
}
message StatusResponse {
  string message = 1;
}
```

2. Humidity Controller Service

Collects and process humidity readings, retrieves current humidity, and enables automatic control.

```
syntax = "proto3";
service HumidityControl {
  rpc SetHumidityLevel(stream HumidityRequest) returns (StatusResponse);
  rpc GetCurrentHumidity() returns (HumidityResponse);
  rpc EnableAutoMode(AutoAdjustRequest) returns (StatusResponse);
}
message HumidityRequest {
  float humidity = 1;
}
message HumidityResponse {
  float currentHumidity = 1;
}
message AutoAdjustRequest {
  bool enable = 1;
}
message StatusResponse {
  string message = 1;
}
```

3. Air Quality Monitor Service

Provides real-time air quality status and streams alerts when pollutants exceed safe levels.

```
syntax = "proto3";
service AirQualityMonitor {
  rpc GetAirQuality() returns (AirQualityResponse);
  rpc MonitorAirQuality(stream AirQualityCheck) returns (stream
AirQualityAlert);
}
message AirQualityCheck {
  string location = 1;
}
message AirQualityResponse {
  string status = 1;
}
message AirQualityAlert {
  string alertMessage = 1;
}
```