Smart Climate Control System – CA Report

Student ID:

Student Name:

Project Name:

Course:

Module:

Institution:

Submission Date:

# Domain Description

The assigned domain for this project is **Smart Home Automation**, a system that aims to improve home comfort, safety, and energy efficiency. The Smart Climate Control System simulates a smart home environment built using Java and gRPC to demonstrate core distributed systems concepts. Its purpose is to show how real-world smart devices be represented through gRPC-based services and interact with clients through a central control interface. The project implements all four types of gRPC communication styles and presents secure, modular, and interactive simulation of device control via graphical user interfaces (GUI’s).

## Description of the Three Core Services

The Smart Climate Control System presents three interactive gRPC services. Each service runs on a separate server using its own gRPC port. Together, they form a distributed system architecture that demonstrates modular design and real-time interaction.

### Thermostat Service

**Function:** Sets and retrieves the current temperature, enables/disables auto-adjust mode, and streams periodic temperature updates to the client.

**Contribution:** Acts as the main controller for heating and cooling simulation. It gives users the ability to monitor and change temperature settings manually or through automation.

**RPC Styles:** Unary (set/get temperature, auto-adjust toggle), Server Steaming (periodic temperature updates).

### Humidity Control Service

**Function:** Accepts a stream of humidity readings from the client and responds with an aggregated status message presenting the average humidity.

**Contribution:** Helps simulate how smart humidifiers or dehumidifiers might assess and react to ongoing changes in humidity over time.

**RPC Style:** Client Streaming (send multiple humidity readings, receives single summary response).

### Air Quality Monitor Service

**Function:** Allows clients to send room names in a steam and receive multiple air quality alerts asynchronously for each room.

**Contribution:** Provides real-time air quality monitoring across various rooms in a house, helping simulate alerts for smoke, CO2 levels, or ventilation suggestions.

**RPC Style:** Bi-Directional Streaming (stream requests and receives alerts simultaneously).

# Service Definitions and RPC

This section outlines the detailed definitions of each gRPC service implemented in the system. It describes the request and response message structures, RPC methods, and types of communication pattern used in each case.

## Thermostat Service

**Service Name:** Thermostat

**RPC Methods:**

1. **SetTemperature (Unary):** Allows the client to set a temperature manually

* Request: TemperatureRequest {float temperature}
* Response: TemperatureResponse {float currentTemperature}

1. **GetCurrentTemperature (Unary):** Fetches current temperature maintained by the server

* Request: Empty {}
* Response: TemperatureResponse {float currentTemperature}

1. **StreamTemperatureUpdates (Server Streaming):** Continuously streams simulated temperature updates to the client over time

* Request: Empty {}
* Response: stream TemperatureResponse {float currentTemperature}

1. **AutoAdjustMode (Unary):** Enables or disables auto-adjust mode for temperature control

* Request: AutoAdjustRequest {bool enable}
* Response: StatusResponse {string message}

## Humidity Control Service

**Service Name:** AirQualityMonitor

**RPC Method:**

1. **SetHumidityLevel (Client Streaming):** Accepts multiple humidity readings and returns a message summarizing the number of values received and their average.

* Request: stream HumidityRequest {float humidity}
* Response: statusResponse {string message}

## Air Quality Monitor Service

**Service Name:** AirQualityMonitor

**RPC Methods:**

1. **MonitorAirQuality (Bi-Directional Streaming):** For each room name received, the server responds with a sequence of air quality alerts. This simulates real-time environmental monitoring and alerts.

* Request: stream AirQualityCheck {string location}
* Response: stream AirQualityAlert {string alertMessage}

Each service is designed to cover one of the four gRPC communication styles:

* **Unary:** Simple request/response (Thermostat)
* **Server Streaming:** Continuous server push (Thermostat)
* **Client Streaming:** Batch client push with single response (Humidity Control)
* **Bi-Directional Streaming:** Continuous client and server push, real-time communication (Air Quality Monitor)

# Service Implementations

Each service in the system is implemented as a dedicated Java class extending its corresponding gRPC base class. These classes define the actual behaviour and logic.

## ThermostatServiceImpl.java

Implements the logic for all Thermostat RPCs:

* setTemperature:
* getCurrentTemperature:
* streamTemperatureUpdates:
* autoAdjustMode:

Security is handled using JWT-based server-side interceptors to validate client tokens.

## HumidityServiceImpl.java

## AirQualityServiceImpl.java

# Use of Naming Services

# Remote Error Handling and Advanced Features

# Client GUI

# Security Features

# GitHub Repository

# Screenshots

# Conclusion