

National College of Ireland

**HDip in Computing**

**Distributed Systems**

**Continuous Assessment (CA) Type: Project**

**Smart Farming**

Mindde Zhou

X23120428

# Domain description

This project is an Smart Farming project, which require us to design an microservices system to track some data from the yard or farm. I designed three protos and write four methods of communication methods, which are simple RPC, server-streaming RPC, client-streaming RPC, and bidirectional-streaming RPC. I write four different requests used these four methods to manage my farm:

The project is a micro service project, which contains lots of new technological means. This project can get environment data in the farm, so that we can take control of every land accurately. And we also can let system decides, when it should irrigate, when it should control illumination, etc. But I didn’t accomplish these code in this project. I only write basic methods like get data and control actions mannualy.

In a word, this project is a very basic, simplified Smart Farm system, it has some significance and advanced framework or technologies, like microservices, server registration and socket communication, server link DBMS for data management, etc. It is a prototype of some large project, but still needs more completion.

# **Service definition**

**1 Add sensor(simple):** we can add new sensors to the yard and when we type the sensor ID and district where the sensor should be, the server will return a success or failure response.

**2.StreamSensorData(Server-Streaming):** I wrote a series of data in an csv file to simulate sensors get different types of data from environment, so when I use this service, the server will return one group of data contains humidity, temperature, district etc. The data will sent to client every few seconds.

**3.Irrigation(Client-Streaming):** Same as send sensor data to client, I also write a csv file to contain irrigation requests. When we run the client, the server will accept irrigation request and read csv file, when it read all requests , it will return an success response means that all irrigation is finished. The server will calculate the total volume it takes.

**4.illuminate(Bidirectional-Streaming):** Server reads the csv file to find which land needs illumination and accept request, it will send success information one by one after reading the data.

# Protos

There are three protos in my project as follows: they are playing different role in smart farming.

### Sensor Implimentation

service SensorService {  
 rpc StreamSensorData(SensorRequest) returns (stream SensorResponse);  
 //Server Streaming Sensor Service  
 rpc AddSensor(AddSensorRequest) returns (AddSensorResponse);  
 //Simple Sensor service  
}  
  
message SensorRequest {  
 bool start = 1;//start message  
}  
  
message SensorResponse {  
 int32 farmid = 1;  
 int32 sensorid = 2;  
 int32 districtid = 3;  
 double tempreture = 4;  
 int32 humidity = 5;  
 int32 illumination = 6;  
 //transport response imformation every 5 seconds  
}  
  
message AddSensorRequest {  
 int32 sensorid = 1;  
 string sensortype = 2;  
 int32 districtid = 3;  
 //type sensor information  
}  
  
message AddSensorResponse {  
 bool success = 1;  
 string message = 2;  
 //sensor information stored in an empty csv file.  
}

### illuminate Implimentation

service IlluminateService {  
 rpc illuminate(stream IlluminateRequest) returns (stream IlluminateResponse);  
 //bidirectional Stream illuminate service  
}  
  
message IlluminateRequest {  
 int32 farmid = 1;  
 int32 districtid = 2;  
 int32 duration = 3;  
 //client add request to server  
}  
  
message IlluminateResponse {  
 bool success = 1;  
 string message = 2;  
 //server response streaming information  
}

### Irrigation Implimentation

service IrrigationService{  
 rpc Irrigation(stream IrrigationRequest) returns (IrrigationResponse);  
 //client Streaming irrigation service  
}  
  
message IrrigationRequest{  
 int32 districtid = 1;  
 int32 farmid = 2;  
 int32 volume = 3;  
 //server read all request   
}  
  
message IrrigationResponse{  
 bool success = 1;  
 int32 volume = 2;  
 //return one response ,and calculate total volume  
}

## Services and variables

#### Service1 AddSensor:

**Add sensor(simple):** we can add new sensors to the yard and when we type the sensor ID and district where the sensor should be, the server will return a success or failure response.

#### Variables:

**Request:**

int32 sensorid

string sensortype

int32 districtid

these are sensor informations , we simulate a situation that if we want to add more sensors to the farm, we should add sensors to the system and register them so that we can get data from that new sensor, once we enter the sensor information, we assume that the sensor is activated, and work successfully.

**Response:**

bool success

string message

These two variables is to let client know if the sensor is added successfully or not.

If sensor information is stored in addsensor.csv, we assume that sensor is registered successfully, so server will response a true response and success message.

### Service2 StreamSensorData

**StreamSensorData(Server-Streaming):** I wrote a series of data in an csv file to simulate sensors get different types of data from environment, so when I use this service, the server will return one group of data contains humidity, temperature, district etc. The data will sent to client every few seconds.

### Variables

**Request:**

Boolean start: a Boolean let service start, once server started, it will read csv files add pass the data to the client.

**Response:**

int farmid ;  
int sensorid ;  
int districtid ;

double tempreture;  
int32 humidity ;  
int32 illumination ;  
I wrote a csv file to simulate server grab data from database. These are environment data, we assume that these data are received from farm environment by sensors, and store them in an csv file.

**Request:**

Boolean start;

a Boolean let service start, once server started, it will read csv files add pass the data to the client.

### Service3 Illuminate

**Illuminate(Bidireation Streaming Service)**

Client read csv and send several request to server and server send success information to client. The number of response the server sends depends on the number of request client sends.

#### Variables:

**Request:**

int32 farmid

int32 districtid

int32 duration

These variables defined which district needs illumination, we assume that sensor detected some place needs more lights, so sensor store a data about these places, and Illuminate server read these data and start illuminate.

**Response:**

bool success

string message

Every time it complete illuminate, it will send a success message to client. It is a bidirectional Streaming service.

### Service4 Irrigation

**Irrigation(Client Streaming Service)**

The client will send a series of request to server, but server will only send one message to client saying that all irrigation is completed. Server will process every request. In order to know if server complete all requests or not, I added another variable called volume. Every irrigation request will let server irrigate 100ml water, and when server finishes irrigation, it will add 100ml to the total volume, and return total volume to client, so we can know whether all request are received and proceeded through total volume.

**Request:**

int32 districtid

int32 farmid

int32 volume

Request variables store the irrigation information like district ID and farm ID, volume is set to a constant variabes: 100ml.

**Response:**

bool success

int32 volume

Response only contains success message and total volume, the server will reveive all request and calculate total volume and return to client, to let user check if all request are executed.

## Github

<https://github.com/canadaell/distributed-system-CA.git>

**github commit:**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**Consul:**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**GUI:**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成