SmartAgCloud

Design Document

CMPE 281
Spring 2019

Professor:
Jerry Gao <Jerry.gao@sjsu.edu>

Term Project Id - Team 1
Hemaprasanthi Mutyala <hemaprasanthi.mutyala@sjsu.edu>
Lakshmi Kameswari Maduri <lakshmikameswari.nishthala@sjsu.edu>
Maahi Chatterjee <maahi.chatterjee@sjsu.edu >
Wenyan He <wenyan.he@sjsu.edu>

Table of Contents

1.Introduction

1.1 Rationale

Smart AgCloud is an IOT-based smart agriculture infrastructure service management system as a service on a cloud. It supports large-scale IOT-based agriculture system infrastructure services for farmers allowing them to have more control over their farm with less human effort and more promising results by predicting the future through real time data from sensors deployed in the field. Each farmer could select install and deploy one or more IOT agriculture networks for their greenhouses or ranch fields. To achieve that, an infrastructure at a unit level is required. This infrastructure should collect, store and process information at the most fundamental level of operation. The IOT-based smart agriculture infrastructure management system encapsulates this logical premise and provides an autonomous agriculture system.

1.2 Objectives

A smart agriculture uses different types of electronic data collection sensors to supply information which is used to manage each part of the ranch efficiently, cultivate different crops in a regulated environment and predict the future through the data from different sensors. The purpose .The goal here is to create a system that gets various readings from sensors installed on fields for research like humidity prediction, moisture control, soil nutrition calculation, and used for various commercial level applications. The major objectives can be listed as:

- Sensor installation, registration and tracking
- Data collection from sensors, nodes and data management
- Data analysis and report generation
- Protected data access to different uses

1.3 System Overview

This project is designed to develop, implement, and validate an IOT-based cloud infrastructure system as a SaaS for Smart Streets in a smart city. Each smart street is a smart node equipped with a set of sensors. Each smart node is in turn connected to a cluster which will be used to control the connected smart nodes and support the communications with the back-end server to send the collected sensor data for all nodes. Each smart node has wireless communication capability which supports node-to-cluster communications.

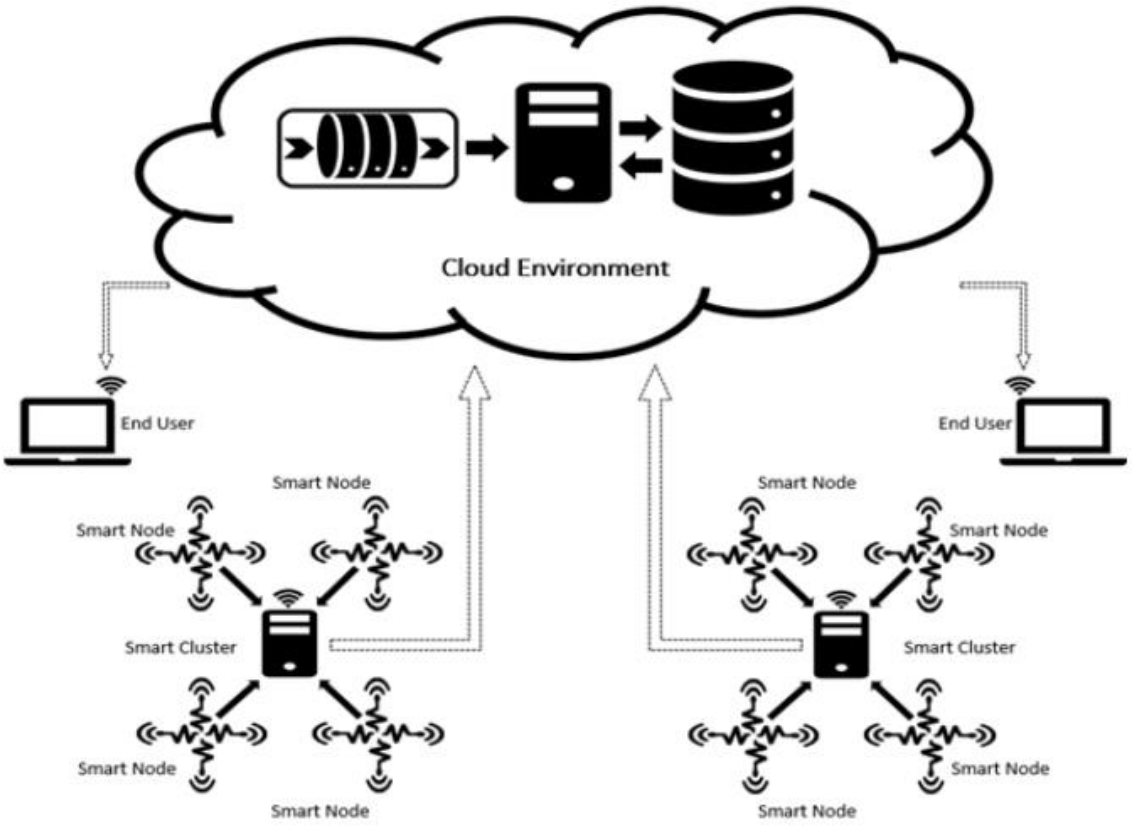


Figure : System Overview

1.4 Product Scope and Actors

The designed product would be able to implement a scalable distributed system which provides support to efficiently retrieve, maintain and update sensor data of the various (nodes) via a cluster node which is connected to a database server in a cloud-based simulation system.
The intended actors for the smart agriculture system are farmers since they are the primary owners of the field who cultivate different crops, IOT Supports because they check the sensor availability, and IOT Manager since he governs communication between several sensors, nodes and databases. The intended actors can be government agencies who would require information to enhance their countries income through detailed evaluation of the crop's status. This system can also be provided as a service to commercial units such as pesticides agencies, local entrepreneurs and many more, who want a complete analysis of the area for better crop management. This system includes the following types of users:

- **Farmer** - Can access the system to check and configure their own IOT networks with smart nodes (with diverse sensors) online by accessing SmartAgCloud to check their IOT sensor status and statistic data – The persons who access the system to find out the status of smart nodes and sensors.
- **IOT Supports** – Can access the system to find out the status of smart nodes and sensors.
- **Infrastructure manager** – Can setup, configure, and manage smart nodes, cluster nodes, and sensors, as well as their connections.

1.5 Outcomes

This system outcome generates profit not only farmers but also for the country as a whole. The system enables to reduce the human effort by representing the required data digitally from sensors. The output of this system provides farmers with the information of different aspects of the field like moisture, sunlight, soil nutrition level to have full control of their field so as to configure the requirements as per the type of the crop cultivated.
This system gives the flexibility to add more sensors or nodes if required. IOT Support has the responsibility of the sensor's health. They can check the sensors availability through heart beats and take required actions immediately. The data flow is governed by Infrastructure manager. Hence the whole system is well developed with shared responsibilities. Irrespective of the experience of the farmer and human effort each one can

grow the crops with little knowledge on farming. The IoT generates massive amounts of data, and cloud computing provides a pathway for that data to travel to its destination. It eliminates the cost of maintenance for data bases, servers.

2. System Requirement And Analysis

2.1 Proposed System Overview

The proposed system is an IOT-based smart agriculture infrastructure service management system. It has been named **SmartAgCloud.** The system will be hosted as a service on a cloud-based architecture and will support large scale on-demand IOT-based services for farmers.

Three users are targeted for the application and all the 3 users will be able to perform different tasks based on their roles. Our application would need a setup on a ranch and the network will consist of smart nodes. Each smart node is installed and equipped with a set of agricultural sensors for moisture, humidity, temperature etc and will have wireless communication capability which supports node to cluster communications.

Each node individually should be capable of data collection and transfer. Depending on the type of user, different functions can be executed.

2.2 System Functional Requirements and Analysis

Continuous real time data monitoring is a very important requirement for this system. The basic purpose of this application is to make monitoring their fields easier and more efficient for the farmers by providing the convenience of an application and the statistics and status of the field and crops.
The greatest selling point of the application would be an easy to use Graphical User Interface with the assumption that the user has next to null knowledge about web-based applications.

Based on the user, the different functions are listed below:

2.2.1 Farmers

- Sign up: This feature will allow the farmers to register themselves in the capacity of farmers to use the application. It will be in the form of an application where the farmer will be prompted to enter their location and then will be guided to configure the sensors.
- Login: This is the login functionality for the farmers. In this way, the farmers won't be restricted to just one device and will be able to access their account through any device and from anywhere.
- List of sensors: By clicking on this link, a farmer will be able to view the listing and type of sensors in his area(glasshouse/ranch). Clicking on the sensor will redirect the farmer to view the stats of that particular sensor.
- Add sensor request: If a farmer wants to add another sensor, he will need to make a request to the IOT support who would be responsible for the installation and configuration. Through this link, he will send a request to the IOT support group with the details of the kind of sensor he needs. Once approved and configured by the IOT support, the sensor would be visible to the farmer on his listings.
- Remove sensor request: If a farmer wants to remove a sensor, he will need to make a request to the IOT support who would be responsible for the uninstallation. Through this link, he will send a request to the IOT support group with the details of the sensor he needs to remove. Once approved and removed by the IOT support, the sensor would not be visible to the farmer on his listings.
- View sensor stats: This component will just be a page displaying the present stats of the sensors installed for a particular farmer. The stats would show the status of the sensor and the data collected through the sensor.
- Map: This is the most interesting feature provided to the farmers. On clicking the link for a map, the farmer would be redirected to their ranch map, showing the location of all the sensors in their ranch. Clicking on any of the sensors would lead the farmer to the page displaying the stats of that sensor.

2.2.2 IOT Support **:** An IOT support group will be responsible for receiving the requests from the farmer for adding or updating a sensor.

- Sign up: This is the registration option for an IOT support group member. Based on the clearance level (a code would be provided by the infrastructure manager), the member can register himself as an IOT support group member and will gain access rights for all the farmers listed under him.
- Login: After successful registration as an IOT support member, he can login into his account and view the list of farmers under him, requests for adding or removing sensors and viewing the status of the sensors.
- List of farmers: This link will display the list of farmers under the member of the IOT support group and the sensors used by the farmer.
- Pending requests: This section will show the IOT support member with all the requests for either adding or removing a sensor from a farmer's list of sensors.
- Add a sensor: If there is a request from a farmer to add a sensor, the IOT support group will be responsible for installing and configuring the sensor. Since, this is just a simulation software for the IOT sensors, the configuration details would need to be fed manually but in case of a real time scenario, picking the sensor from a list of sensors would fill in the configuration details by reading the sensor. ***Configuration of a sensor is mostly the job of an infrastructure manager where in he can configure the sensor within a node using a floorplan.***
- Delete Sensor: Similarly, if an IOT support group member receives a request for removing a sensor from a farmer's network, he can do so by simply selecting the sensor and clicking delete. The sensor and the sensor data would be removed from the farmer's dashboard and account. Since this is a simulation software, the sensor would only be removed from the database. However, in real time situations, manual uninstallation will also be required.
- View Sensor data: After selecting the farmer, the IOT support group member will be able to view the stats and status of the sensors listed under the farmer. This would open up the same page as the one that opens when a farmer looks to monitor his sensor stats.
- Node connectivity status: This component is simply to show if all the nodes are connected and are in sync. Since this is a simulation software, real life data will have to be used to replace the hard-coded data used for the prototype.

2.2.3 Infrastructure Manager: The infrastructure manager is responsible for the setup, configuration and management of the smart nodes and the smart sensors.

- List of users and roles: This module will display the list of users registered with the app and their corresponding roles as a farmer or an IOT support member.

- Manage Cluster Node: As the IOT support group approves a sensor addition request or deletion request, the infrastructure manager gets a notification about it and he is responsible for attaching the IOT sensor to a node.
- View Nodes: This component will simply give an overview of the nodes and the sensors that belong to each node.

2.3 Non Functional Requirements

Non-functional requirements include features and criteria that define the operation of an application or a software instead of the functions. It covers the features describing the scalability, speed and maintainability of the application and helps in determining the appropriate infrastructure to be used and the quality that can be expected of the application. For the **SmartAgCloud,** the non-functional requirements are stated.

2.3.1 Software Quality User Interface

The quality of the software will be decided by the ease of use for the user. Since the target group of customers is mostly farmers, it is important to ensure that the user interface is structured and designed in such a way that it becomes easier for the farmers to navigate through the application and perform the required tasks.

2.3.2 Scalability

The application needs to be highly scalable. It is important for the application to handle multiple user requests and maintain a safe point for data collection. Therefore, to ensure this, we will be distributing the server using load balancers and host the application on AWS thereby implementing the concept of Infrastructure as a Service.

2.3.3 Performance

One of the major concerns with a distributed system architecture is the performance so that heavy load of requests on the servers do not slow it down. This application is a monitoring application therefore, it has to be up and running at all times. Performance will be greatly impacted by the use of concepts like redis and express.

2.3.4 Availability

As an application to help the famers with their everyday chores, it is important that the application is highly available and accessible to the farmers for their use. They must be aware of the existence of this application and the sources from which it can be accessed. One of the major setbacks with the application would be a lack of availability of internet connection and therefore, it becomes important to ensure that no loss in data takes place and the dependency on internet is significantly less. Hence, we focus on building this application with a mesh node architecture.

2.3.5 Security

Since there is data involved with the application, security of the data also becomes an important factor for consideration. Actor accounts would be password protected and any password that is stored in the database would be parsed through hashing algorithms.

2.3.6 Maintainability

The major areas of concern for maintainability include physical maintenance of the sensors as the sensors would be exposed to the environment and have more likelihood of getting corroded or damaged. Apart from the that, in terms of software, always upgrading the quality of sensors and the features in the application would also play an important role.

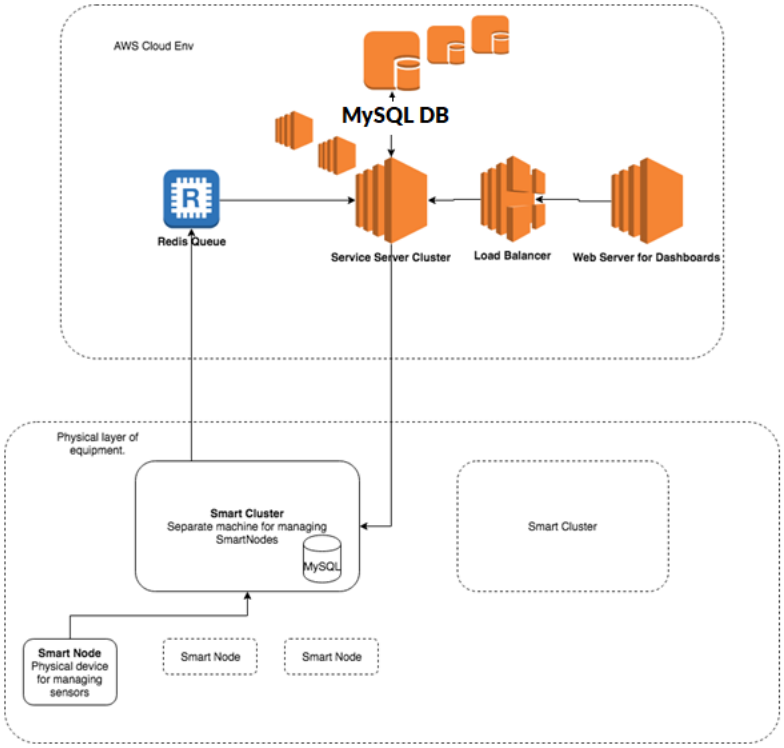
3. System infrastructure and architectures

3.1 Cloud-based IOT sensor system infrastructure design

The real Smart Agriculture Network will contain from two main parts:

- ∉ Physical layer of equipment
- ∉ Cloud based software to collect and manage data and equipment

The diagram below shows those two elements:

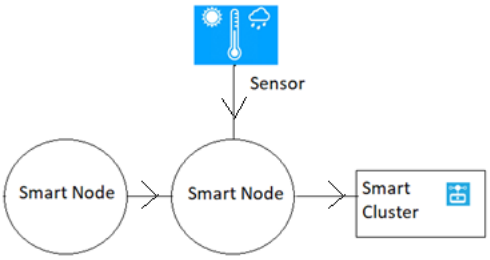


3.1.1 Physical layer of equipment

The Physical layer of equipment contain Smart Nodes and Smart Clusters. The responsibility of each is listed below:

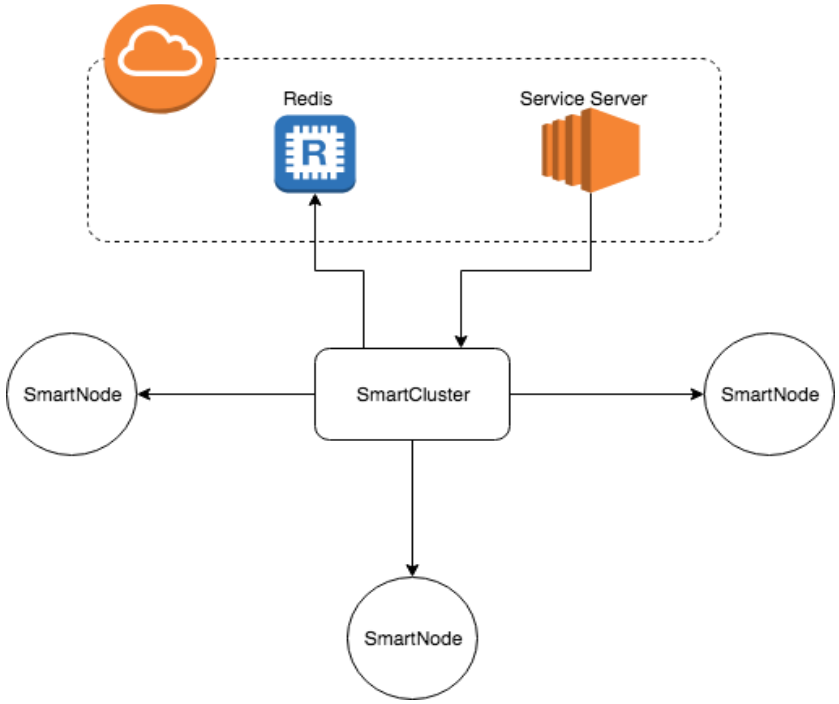
SmartNode:

- ∉ Collect data from sensors
- ∉ Collect health data of sensors
- ∉ Allow disable/enable, connect/disconnect different (supported) types of sensors
- ∉ Pass the information to other nodes



SmartCluster:

- Register SmartNodes
- Collect data from SmartNodes with the intervals determined by configuration
- Collect health data from SmartNodes
- Push data into Queue (Redis Queue)



For education purpose the Physical layer of equipments will be forge with cloud based applications which will mock the real devices by implementing appropriate API interfaces.
All below architecture and analysis will be based on this statement!

3.1.2 Cloud based software

The Cloud based software will contain from the next main components:

- Service Server (Node JS)
- Queue (Redis)
- DataBase (MySQL)
- Web Server (Node JS)

The responsibility of each is listed below:

Service Server:

- ∉ Manage and configure SmartClusters
- ∉ Pulling and transform Sensors Data and Health data from Queue and store it in DB
- ∉ Expose Rest API to access to Sensors Data

In terms of support horizontal scalability all Service Servers will be behind Load Balancer.

Queue (Redis)*:*

- ∉ Accumulate data in the queue
- ∉ Give back data from the queue

The main purpose to use Queue is to prevent data loss in case of Service Server maintenance or crashing.

DataBase (MySQL) responsibility:

- ∉ Store Sensors Data and HealthData
- ∉ Provisioning different types of searching requests
- ∉ Provisioning good horizontal scalability

In terms of horizontal scalability, which is for my opinion is very important for such type of data (sensors time series data) the most suitable DB is MySQL.

Web Server (Node JS) :

- ∉ Provide 3 types of dashboard for users (Farmers, IOT Support, Infrastructure manager)
- ∉ Users Authentication and Authorization
- ∉ User management functionality for 4th type of user - admin.

3.2 Deployment-Oriented System Infrastructure

Our Cloud Infrastructure Provider is AWS. For SmartNodes, SmartClusters, Service Server and Web Server will be used ec2 instances.

As a software deployment mechanism the docker image containing will be used.

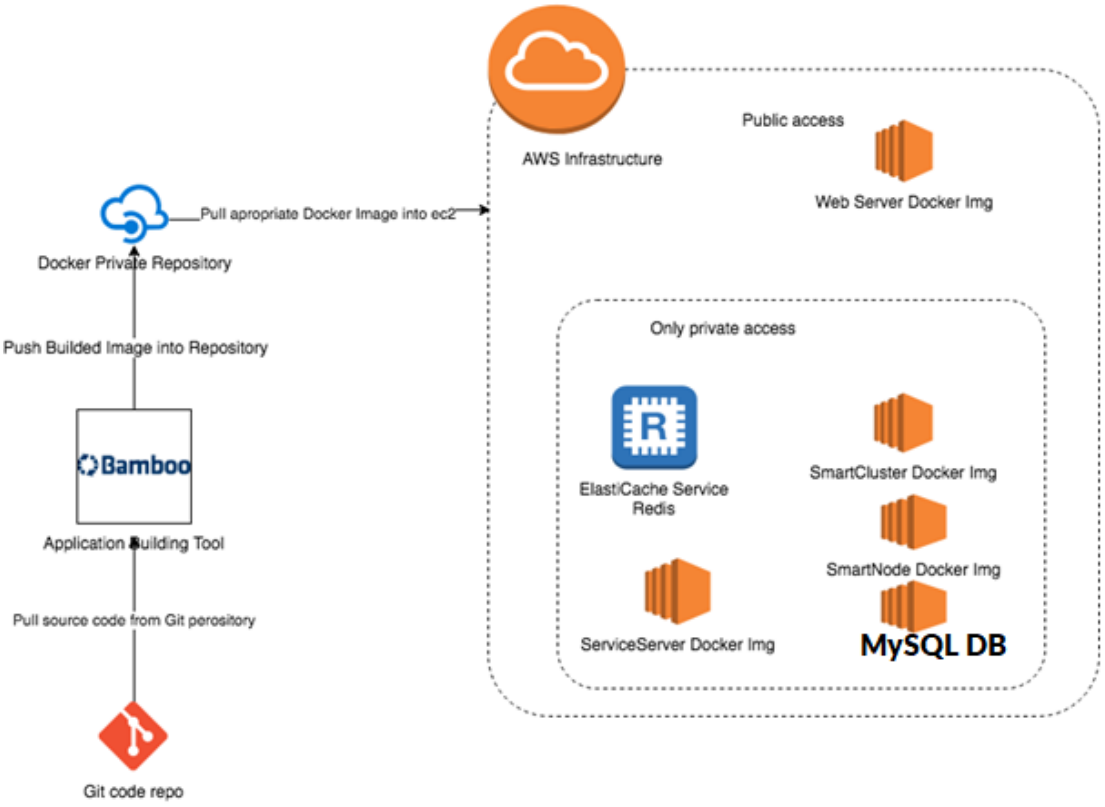
The deployment process:

- Instantiate ec2 instance in AWS - Amazon Linux **AMI 2018.03.0** with Docker
- ssh into instance
- Run Docker image with appropriate settings

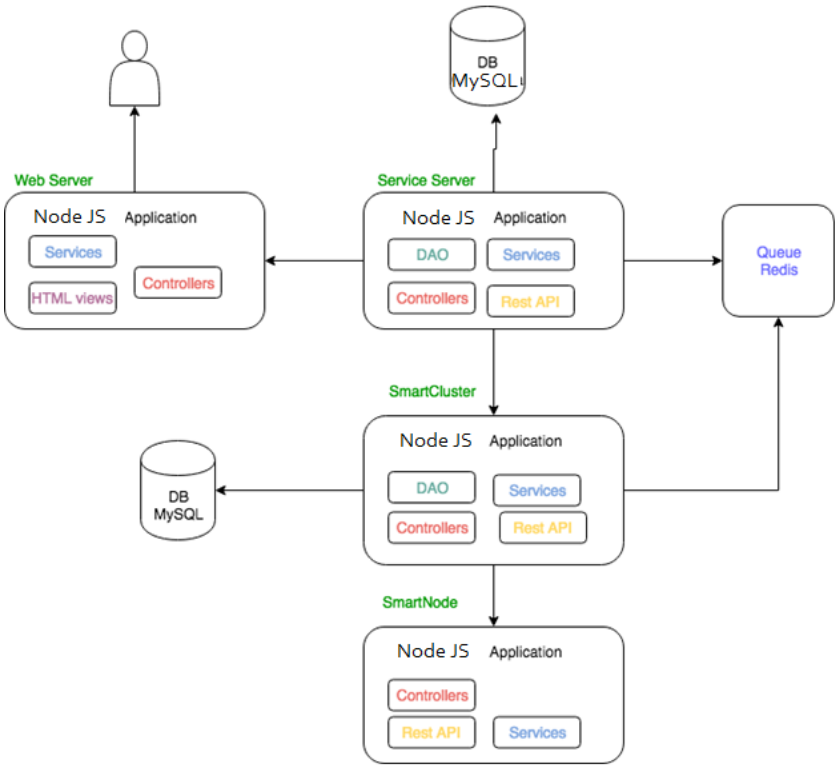
Security groups:

- Private access security group (SmartNode, SmartCluster, Queue, Service Server)
- Public access security group (Web Server)

In real life the Queue should have public access and SmartCluster should use authentication credentials to be able to push data into. In our education project we don't need it since SmartNodes and SmartClusters will be run in Private Security Group.



3.3 System-Oriented Component Functional View



3.4 Database design

In our project will be used two types of DB: MySQL for Smart Cluster and MySQL for ServiceServer.

3.4.1 MySQL for Smart Cluster

This DB dedicates to store the SmartCluster related data like:

- ∉ Smart Nodes data (id, name, location, description, type, ip_address)
- ∉ Smart Cluster data (clusterId, name, location, description, config, ip_address, type)

Config data will contain: Smart Nodes status ping intervals, sensors data collecting intervals.

SmartNode	
PK	Id
	name
	location
	type
	ip_address
	description

ClusterConfig	
PK	clusterId
	name
	location
	ip_address
	description
	type
	config

3.4.2 MySQL for ServiceServer

This DB dedicates to store:

- ∉ sensors time series data
- ∉ Smart Nodes statuses data

Sensors time series data

DataBySensor	
PK_P	sensorId
PK_P	nodeId
PK_P	clusterId
PK_C	timestamp
	data

DataByNode	
PK_P	nodeId
PK_P	clusterId
PK_C	timestamp
	sensorId
	data

DataByCluster	
PK_P	clusterId
PK_C	timestamp
	nodeId
	sensorId
	data

DataByTime	
PK_P	timestamp
PK_C	sensorId
	nodeId
	clusterId
	data

Statuses Data

StatusCheck	
PK_P	clusterId
PK_C	timestamp
	statusData

PK_P - Primary Key Partitioned
PK_C - Primary Key Clustered

3.4.3 Redis Queue

In purpose of buffering the data we are going to use Redis Queue. There are 2 types of messages:

- sensors data
- Smart Nodes statuses data

As data structure the JSON will be used:

```
{
        “messageType”: MessageType,
        “content”: {contentJsonObject}
}

enum MessageType:{
        STATUS, SENSOR_DATA
}
```

4. Cloud-based System Design and Component Interaction Design

4.1 Cloud Based System Design

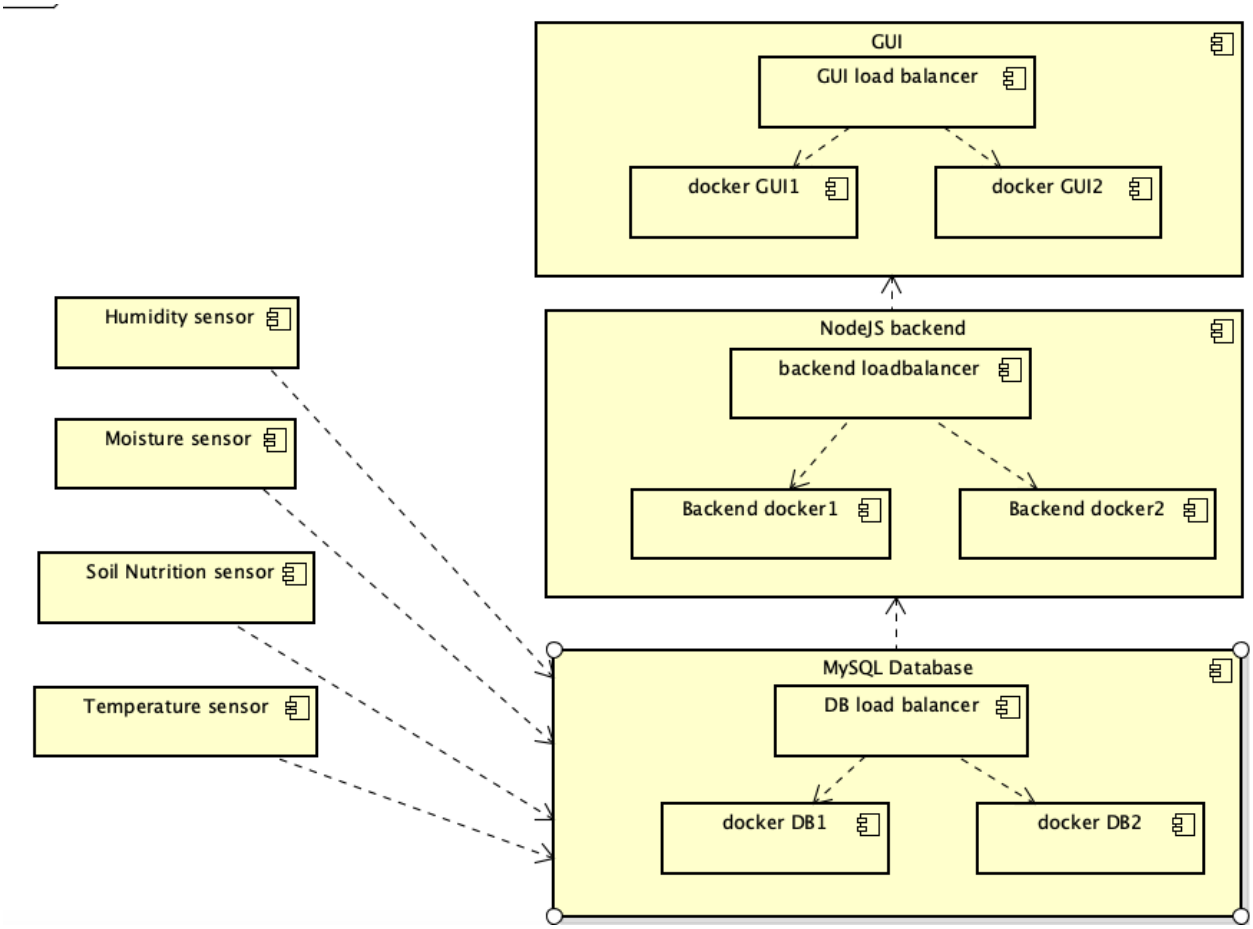


Figure 4.1 Cloud Based System Architecture

The above diagram represents the cloud-based architecture of our project. The detailed functions of each component are listed in the following section.

4.2 Component Functions and Data Formats

4.2.1 Sensor Network:

The Sensor Network is used to sense and collect data for different parameters like temperature, humidity, moisture and nutrients in soil. The following are the sensors in the network and their data format and functionality:

Sensor Name	**Data Format**
Humidity Sensor	grams per cubic meter(g/m3)
Moisture Sensor	grams per cubic meter(g/m3)
Soil Nutrition Sensor	pH
Temperature Sensor	Farenhiet

Humidity sensor - Senses humidity in the atmosphere
Moisture sensor - Detect and measure the moisture content in the soil
Soil Nutrition sensor - Measure the nutrient content in the soil
Temperature sensor - Measure the temperature so that the farmer can grow crops suitable for that temperature

4.2.2 MySQL Database:

The MySQL database component will store the sensor data for the entire application. The database will interact with the NodeJS backend and also the Sensor Network.

Input -> Data from the sensor network
Output -> Data is output to the NodeJS backend

4.2.3 NodeJS Backend:

The NodeJS backend interacts the MySQL database and also the GUI. The data from the database is processed and output to the GUI to display to the user.

Input -> Data from the database
Output -> data to be displayed to the GUI

4.2.4 GUI(ReactJS)

The GUI component interacts with the end user. It displays data to the user in a convenient way in which the user can view and understand.

4.3 API Design

The API of our application represents how the end-users can access data from the application. The final routes will vary slightly from the ones mentioned below.

4.3.1 API Routes and Functionalities

API Route	**Description**
/login	Asks for username and password and allows users to login to the application
/SignUp	Allows new users to register for the application

4.3.2 User-Specific API Routes

Farmers:

/map	Displays map of sensors. Allows users to add a sensor
/sensors	Lists all the sensors of each type(ex: humid sensor1, humidity sensor2, etc.,)
/sensor-data	Displays a table showing the status of each sensor and its location, area of coverage, sensor data, etc.,
/weather	Displays details about weather in the

	region

IoT-Support routes:

/sensors/sensorID/status	Displays status(on, off) of each sensor in the network
/cluster/status	displays the status of smart cluster to the user
/central-node/status	Displays the status of central-node to the end-user

5. Project Deliverables and Schedule

The following is the schedule for the project deliverables:

#	Project Timeline	Deliverable
1.	Feb 27 - March 12	Design Document
2.	March 12 - March 29	a. Node JS component part1 b. Database component part1
3.	March 30 - April 15	a. Node component part2 b. Database component part2
4.	April 16 - April 30	a. Apply application on EC2 b. Run application components on docker c. Connect different cloud components

5.	May 1 - May 4	Optimize the cloud deployment
6.	May 5 - May 7	Work on project document

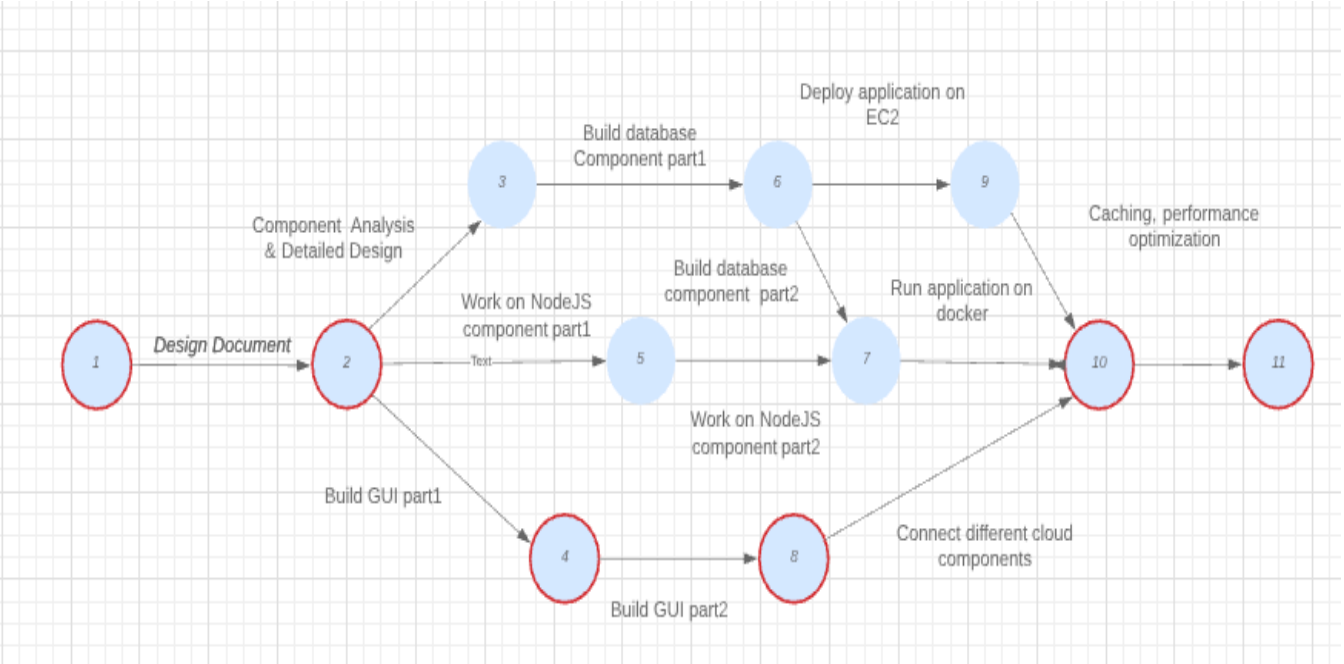


Figure 5.1 PERT Chart depicting the Project Schedule

6. Cloud Technology and Validation

6.1 Cloud Technology

The cloud technology that we would use for this project is Amazon Web Services for hosting the application. AWS is an Infrastructure As A Service and we will use it too instantiate multiple EC2 instances for servers and clients to support a Distributed Architecture.

The application is a full stack software designed using the Javascript framework. The backend will be in node js and the client side is designed in react js. Some components of the frontend will support react-redux with Apache Kafka as the messaging queue. The major database in the application is MySql but some of the data will be stored in MongoDB to increase performance and to show the scalablitiy of the application.

Amazon MSK ie Managed Streaming for Kafka may also be used depending on the schedule of the application for supporting Kafka Messaging. The reasons for selecting AWS as our application hosting technology are as follows:

- Pay-as-you-go: The best feature of using AWS that the billing is completely based on the usage of the resources. AWS provides several kinds and variety of resources to choose from depending on the application that we intend to deploy on it. Therefore, it removes the dependency of an application on the local machine as there might be some resources that are overutilized while some might be underutilized. AWS provides us with the exact amount of resources that we need for the application.
- Scalability: AWS helps in making our application scalable by providing services like load balancing and database sharding. It also supports auto-scaling which reduces the developer's work by a considerable amount and because of it's elasticity in such domains, AWS becomes the best choice.
- Security and Network: AWS has recognized certifications for security like HIPPA and SAS70 which is a huge assurance as code plagiarism is very common. The applications have a variety of options to configure the application like bandwidth and protocols.
- Flexibility: The very option of picking our own configuration for deployment and designing any application makes AWS one of the most flexible platforms available. Right from storage, memory to network, developers are free to configure their own platforms for their applications. It is under our control.
- Reliability: AWS is highly reliable as the instances are rapidly commissioned. It makes the application safe as there is very less or no latency or loss in efficiency.

6.2 Validation Environment

Validation is one of the most important and under-rated phases of the software development cycle. It's however very important to make sure that the final application matches the requirement stated for the application. It's also important to measure the performance of the application after deployment on AWS.

Below are the mentioned validation environments we will be following:

- JMeter: Apache JMeter is a load testing tool especially for analyzing and measuring the performance of different services provided by any web application. It is an open source Java application and covers several categories of tests like load testing, functional requirements, regression testing and performance testing. We will be validating the software with and without deployment on the cloud and with and without redis implementation with the SQL tables.
- Manual Testing: Manual testing will be performed by individual team members with their respective components as they implement new functionalities. This would include different approaches of white box testing.