**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](mailto:%3chemaprasanthi.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**

**Table of Contents...................................................................................................................................4**

**1 Introduction......................................................................................................................................4**

1.1 Rationale..............................................................................................................................................................4

1.2 Objectives…………….............................................................................................………………………...........4

1.3 System overview………………………………..................................................................................................5

1.4 Product Scope and actors.............................................................................................................................6

1.5 Outcomes………………………………………………………………………………..………………….……............6

**2 System Requirement and Analysis............................................................................................7**

2.1 Proposed System Overview.........................................................................................................................7

2.2 Functional Requirements..............................................................................................................................7

2.2.1 Farmers…………………………………………………………………………...………………………………8

2.2.2 IOT Support……………………………………………………………………………………………..………9

2.2.3 Infrastructure Manager…………………………………………………………………………………..10

2.3 Non Functional Requirements.................................................................................................................10

2.3.1 Software Quality User Interface……………………………………………………………………....10

2.3.2 Scalability………………………………………………………………………………………………………11

2.3.3 Performance…………………………………………………………………………………………………..11

2.3.4 Availability…………………………………………………………………………………………………….11

2.3.5 Security………………………………………………………………………………………………………….11

2.3.6 Maintainability………………………………………………………………………………………………11

**3 System infrastructure and architectures..............................................................................12**

3.1 Cloud-based IOT sensor system infrastructure design..................................................................12

3.1.1 Physical layer of equipment…………………………………………………………………………….13

3.1.2 Cloud based software………………………………………………………………………………………14

3.2 Deployment-Oriented System Infrastructure.....................................................................................15

3.3 System-Oriented Component Functional View..................................................................................17

3.4 Database Design..............................................................................................................................................17

3.4.1 MySQL for Smart Cluster...................................................................................................................17

3.4.2 MySQL for ServiceServer...................................................................................................................18

3.4.3 Redis Queue.............................................................................................................................................19

**4 Cloud-based System Design and Component Interaction Design..................................20**

4.1 Cloud Based System Design........................................................................................................................20

4.2 Component Functions and Data Formats.............................................................................................21

4.2.1 Sensor Network.....................................................................................................................................21

4.2.2 MySQL Database....................................................................................................................................21

4.2.3 NodeJS Backend.....................................................................................................................................21

4.2.4 GUI(ReactJS) ...........................................................................................................................................21

4.3 API Design..........................................................................................................................................................22

4.3.1 API Routes and Functionalities…………………………………………………………………………22

4.3.2 User-Specific API Routes…………………………………………………………………………………..22

**5 Project Deliverables and Schedule............................................................................................23**

**6 Cloud Technology and Validation..............................................................................................24**

6.1 Cloud Technologies........................................................................................................................................24

6.2 Validation Technologies...............................................................................................................................25

**7 References……………………………………………………………………………………………………………26**

1. **Introduction**

Agriculture is indispensable to any country. Much attention is paid to boosting agricultural yield and improving agricultural management. The advent of IOT makes the job easier with the help of Smart Agriculture systems. A lot of work has been completed for Smart Ag systems [1, 2, 3]. Our effort in this project tries to study the important role played by cloud in such a Smart Ag system.

**1.1 Rationale**

Smart AgCloud is an IOT-based smart agriculture infrastructure service management system as a service on a cloud. It is a 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 can view field’s water content, soil nutrition content, moisture level, humidity at his field. Farmer can also see where each sensor is deployed in the field in a map interface. He can add or delete sensors. Farmers can have one or more IOT agriculture networks for their ranches. To achieve this, we are collecting the data from the smart sensors and storing them in a database in the cloud. In this we are using mesh networking for node to node connectivity. This provides and autonomous solution to the farmers.

**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 goal of this project is to collect data from the sensors that are deployed in the field then pass on to the nodes which are installed with mesh networking and then collect the data from the nodes to the cluster, which finally passes the information to the cloud. Here we are installing the sensors to monitor soil, humidy, and water. This application can be used in several level applications. The major objectives can be listed as:

* Sensor installation, registration and tracking
* Data collection from sensors, nodes, clusters, and data management
* Data analysis and report generation
* Secured 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 agriculture. A smart node is a collection of different kinds of sensors. A node has wireless communicate between nodes and clusters for communicating the data of the sensors collected. A cluster has the responsibility to collect the data from the nodes about the senors and communicate with the back end server i.e., into the cloud.

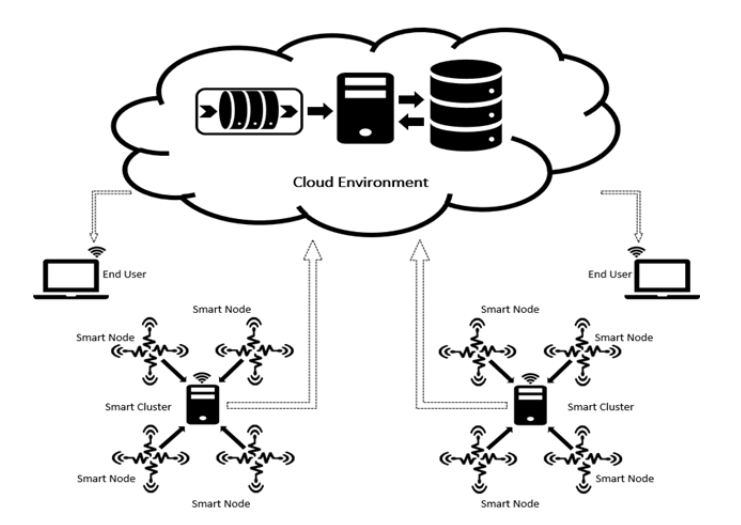


Figure : 1.3 System Overview

**1.4 Product Scope and Actors**

The product will be a scalable distributed system provding efficient maintaianace, retrieval, updation of sensors, nodes and clusters in a cloud-based simulation system.

The targeted actors for this project are farmers since they are the primary owners of the field who cultivate different crops, IOT Support checks and tracks the sensor availability and registration, and IOT Manager governs communication between several sensors, nodes, clusters, and databases. The targeted actors can also be government agencies who would require data to improve their countries income through detailed evaluation of the crop’s production status. This project can be given as a service to not only farmers but also government agencies, pesticides companies, landlords, entrepreneurs and many more to whom ever crop status is required to make crop management easy. 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 different sensors) online through accessing the application
* **IOT Supports** – Can access the system to find out the status of smart nodes and sensors and perform sensor health check
* **Infrastructure manager** – Can setup, configure, and manage smart nodes, smart clusters and their communication to the cloud

**1.5 Outcomes**

This system outcome generates profit not only to farmers but also for the country. 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 and 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 everyone can cultivate the crops with little knowledge on farming with approximately same results.

**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 and node to node communications.

Each node individually should be capable of data collection from sensor and nodes and transfer to other nodes or cluster. Depending on the type of user, different functions can be executed.

The cloud architecture consists of the Web (GUI) server which hosts the GUI for the application in the GUI layer. The Backend layer consists of the backend server hosting the NodeJS backend. The database layer consists of the MYSQL database hosting the data from the sensor network.

**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 his name, location and email address to register for the application.
* Login:This is the login functionality for the farmers. Since the application is in cloud, the farmers won’t be restricted to just one device and will be able to access ranch status irrespective of the location and time.
* List of sensors:By clicking on this link, a farmer will be able to view the listing and type of sensors in his area(ranch). The farmer will be able to see a map of his ranch divided into different areas. Each area will indicate status as green or red indicating whether the status of the area is good or critical respectively. Clicking on the individual sensor will redirect the farmer to view the stats of that 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. This 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. The farmer needs to click a ‘+’ symbol at the top of the page and select the type of sensor to add a new sensor.
* 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 farmer. The stats would show the status of the sensor and the data collected through the sensor. The stats of the sensor would be shown as a meter representing the reading of 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. The map will show different areas of the far. Each region will be either red, orange or green based on whether the area is critical, moderate or in good condition.

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 also to track the sensor status

* 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 tracking 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. 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 the same page as the one that opens when a farmer looks to monitor his sensor stats.
* Node connectivity status: This component is to represent 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 clusters, 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 architecture, health, and data of the nodes, sensors and their connectivity.

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

Sensor:

Each Sensor must be able to collect data from the ranch and send the data to the node.

Node:

Each Node should be able to collect the data from its assigned sensors, communicate with the other other nodes to either send or receive the data, and send the information to the node.

Cluster:

Each clustor should be able to collect the data from the nodes and send t he collected data to the temporary data base and cloud database.

**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 load of user requests using load balancers and auto scaling features by hosting 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 always has to be up and running . Performance will be greatly impacted by the use of concepts like redis and express which handle caching and routing respectively.

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 with a temporary database.

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. Security in the cloud is ensured by using a Firewall to ensure access to only specific ports for different components in the cloud infrastructure.

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 designed Smart Agriculture Network will contain two parts:

* Physical layer of equipment (various sensors)
* Software based on cloud to receive and manage data from sensors

The diagram below shows these two elements:

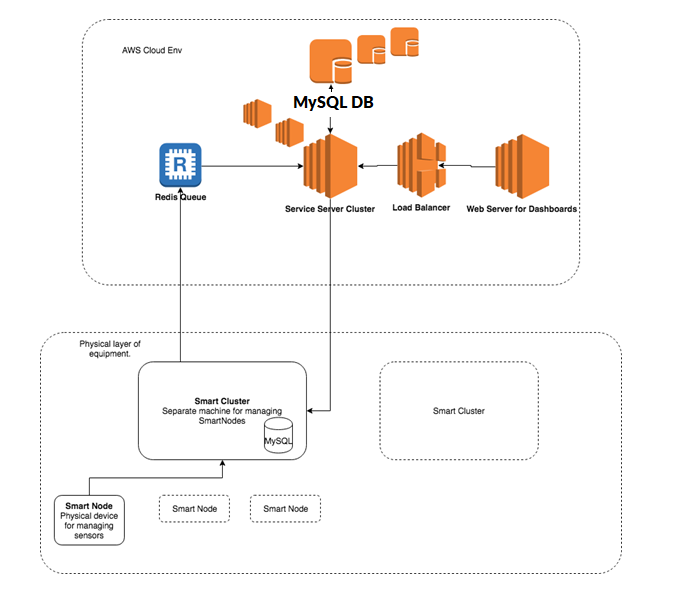


Fig: 3.1 cloud based IOT sensor system infrastructure design

3.1.1 Physical layer of equipment

The Physical layer consists of Smart Sensors, Smart Nodes and Smart Clusters. The responsibility of each of the them is listed below:

SmartNode:

* Collect data from sensors
* Collect health data of each sensors
* Pass on the data collected to the nodes or to the clusters

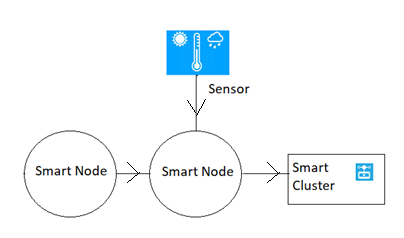


Figure: 3.1.1.a Smart node communication design

SmartCluster:

* Register SmartNodes
* Collect/Recieve data from SmartNodes
* Collect health status data from SmartNodes
* Push data into Queue (Redis Queue)

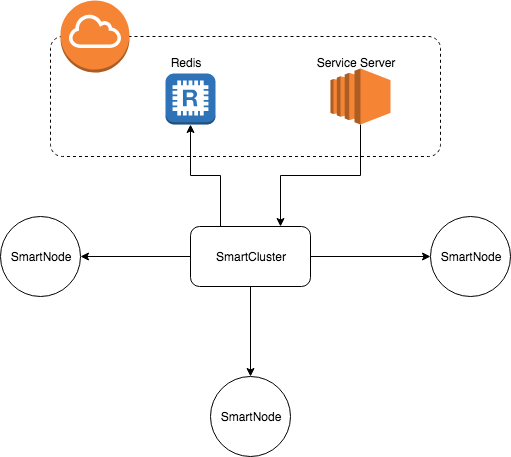


Figure: 3.1.1.b Smart cluster communication design

***For education purpose the Physical layer of equipments will be forged 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, MongoDB)
* Web GUI Server (Node JS)

The responsibility of each is listed below:

Service Server:

* Management and configuration of the SmartClusters
* Collection of the Sensors Data from Queue and push it into DB
* Access of Rest API to view to Sensors Data

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

Queue (Redis)*:*

* Gather collected informatio/data into the queue
* Return 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:

* Storing Sensors Data
* Allowing several kinds of searching requests
* 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) :

* Provides 3 types of dashboard interfaces for clients (Farmers, IOT Support, Infrastructure manager)
* Users Authentication and Authorization
* User management for fourth user type - admin.

**3.2 Deployment-Oriented System Infrastructure**

AWS is our provider for Cloud Infrastructure. Ec2 instances will be used for SmartNodes, SmartClusters, Service Server and Web Server. Docker image containers will be used as a software deployment mechanism.

The deployment process:

* Instantiate ec2 instance in AWS - Amazon Linux **AMI 2018.02.0** with Docker
* ssh into instance
* With appropritate settings run the docker image

Security groups:

* Private access is for security group (SmartNode, SmartCluster, Queue, Service Server)
* Public access is given to the security group (Web Server)

Generally, the Queue should be given public access and SmartCluster should be given with private access for authentication credentials to be able to send data into back-end server. In this project we ignore this part since SmartNodes and SmartClusters are running on Security Group which is private.

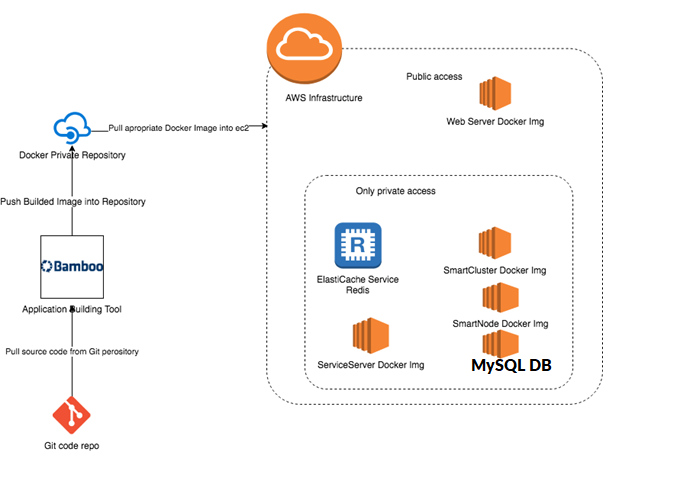


Fig 3.2: Deployment Oriented System Infrastructure

**3.3 System-Oriented Component Functional View**

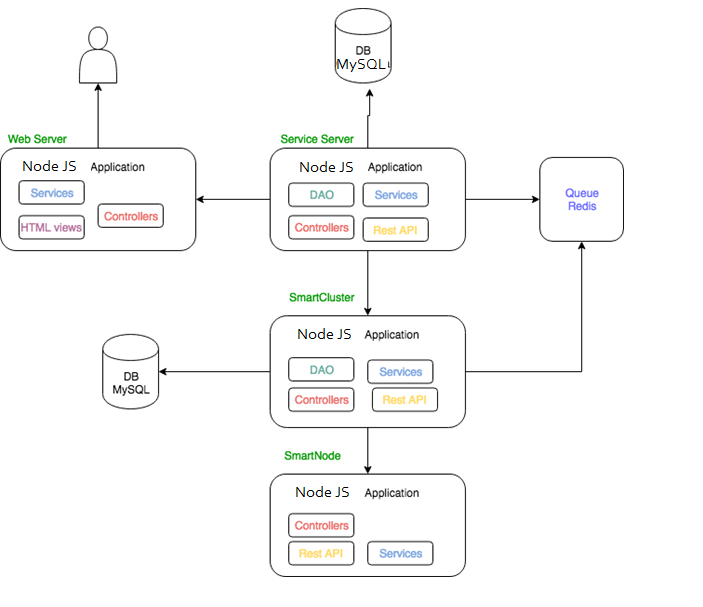


Figure: 3.3 System-Oriented Component Functional View

**3.4 Database design**

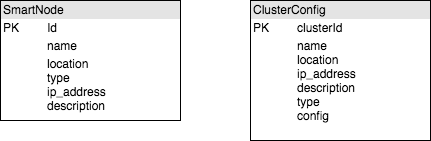
MySQL DB is used for Smart Clusters and ServiceServer.

3.4.1 MySQL for Smart Cluster

This DB dedicates to store the SmartCluster related data like:

* Smart Nodes table(ID, name of the smart node, location, communication, description, type of the node, sensors in that node)
* Smart Cluster table(ID, name of the smart cluster, location, description, config, type, nodes in that cluster)

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

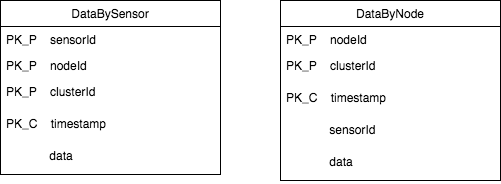


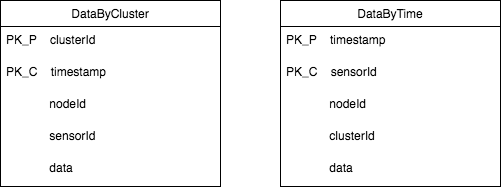
3.4.2 MySQL for ServiceServer

This DB dedicates to store:

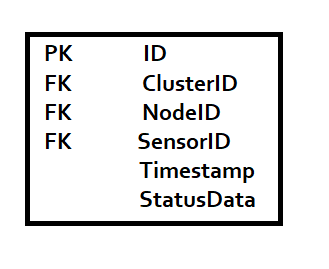
* Time series data of sensors
* Status data of Smart Nodes

Time series data of sensors





Statuses Data

*PK\_P - Primary Key Partitioned, PK\_C - Primary Key Clustered*

3.4.3 Redis Queue

For the data buffering we are implementing Redis Queue. There are 2 types of messages:

* sensors data
* Smart Nodes status 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**

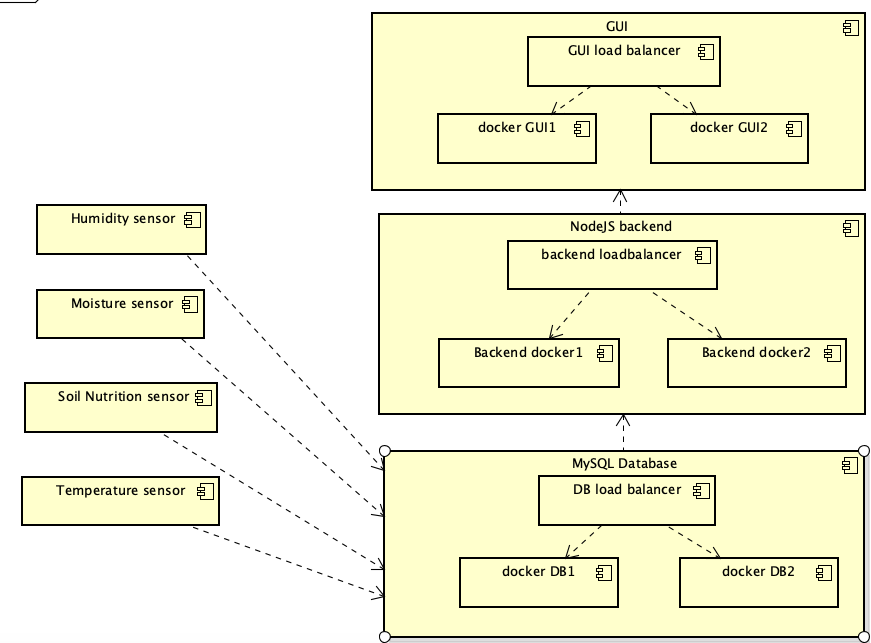
****

Figure 4.1 Cloud Based System Architecture

The above diagram represents the cloud-based architecture of our system. 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 measure and collect data for different parameters such as 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 - Measures humidity in the atmosphere

Moisture sensor - Measures the moisture content in the soil

Soil Nutrition sensor - Measures the nutrient content in the soil

Temperature sensor - Measures the ambient temperature so that the farmer can choose what crops to growthat are more suitable for that temperature

4.2.2 MySQL Database:

The MySQL database component stores the sensor data for the entire application.

The database will interact with the NodeJS backend and the Sensor Network.

**Input ->** Data from the sensor network

**Output ->** Data is sent to the NodeJS backend

4.2.3 NodeJS Backend:

The NodeJS backend interacts the MySQL database and 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 through GUI

4.2.4 GUI(ReactJS)

The GUI component interacts with the end user. It displays data to the user in a user friendly graphical interface through which the user can view and understand the statistics about the ranch and the cloud.

**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 | 1. Node JS component part1 2. Database component part1 |
| 3. | March 30 - April 15 | 1. Node component part2 2. Database component part2 |
| 4. | April 16 - April 30 | 1. Apply application on EC2 2. Run application components on docker 3. Connect different cloud components |
| 5. | May 1 - May 4 | Optimize the cloud deployment |
| 6. | May 5 - May 7 | Work on project document |

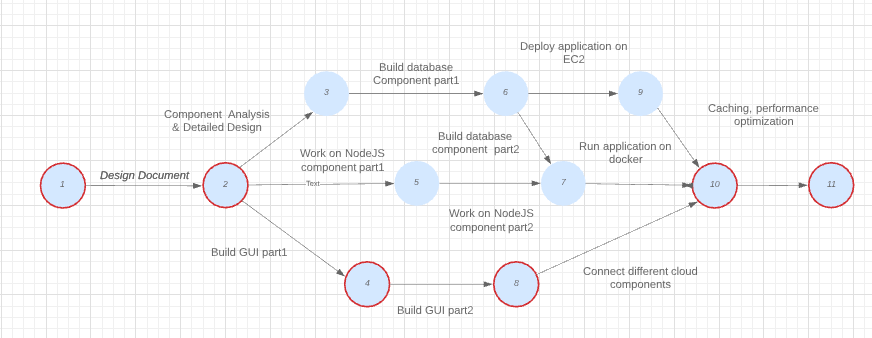
****

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 sharing. It also supports auto-scaling which reduces the developer’s work by a considerable amount and because of its 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 requirements stated in the requirement document. 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.

**7. References**

* [1] H. Channe, S. Kothari, D. Kadam, “Multidisciplinary Model for Smart Agriculture using Internet-of-Things(IOT), Sensors, Cloud-Computing, Mobile-Computing & Big-Data Analysis”, IJCTA Proceedings pp.374--382, 2015.
* [2] M. Hori, E. Kawashima, T. Yamazaki, “Application of Cloud Computing to Agriculture and Prospects in Other Field”, FUJISTU Sci. Tech.J., Vol. 46, No. 4, pp. 446--454, October 201.
* [3] T. Fan, “Smart Agriculture Based on Cloud Computing and IOT”, Journal of Convergence Information Technology (JCIT), Vol. 8, No. 2, Jan 2013.