# PROJECT REPORT

Project Name: **SMARTFARMER- IOT ENABLED SMART FARMING APPLICATION**

Team ID: **PNT2022TMID04018**

Team: YASHWANTH D (TEAM LEAD)
MUGILAN
ARUN PRAKASH
G PRASANTH

# INTRODUCTION

**SmartFarmer - IoT Enabled Smart Farming Application**

IoT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors. Farmers can monitor all the sensor parameters by using a web or mobile application even if the farmer is not near his field. Watering the crop is one of the important tasks for the farmers. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and controlling the motor pumps from the mobile application itself.

**Project Flow:**

1. The parameters like temperature, humidity, and soil moisture are updated to the Watson IoT platform.
2. The device will subscribe to the commands from the mobile application and control the motors accordingly.
3. APIs are developed using Node-RED service for communicating with Mobile Application.
4. A mobile application is developed using the MIT App inventor to monitor the sensor parameters and control the motors.

**To accomplish this, we have to complete all the activities and tasks listed below**:

1. Create and configure IBM Cloud Services
2. Create IBM Watson IoT Platform
3. Create a device & configure the IBM IoT Platform
4. Create Node-RED service
5. Create a database in Cloudant DB to store all the sensor parameters
6. Develop a python script to publish and subscribe to the IBM IoT platform
7. Configure the Node-RED and create APIs for communicating with mobile application
8. Develop a mobile application to display the sensor parameters and control the motors

**LITERATURE SURVEY**:

1.Dr.D.Saraswathi, 2 P.Manibharathy, 3 R.Gokulnath, 4 E.Sureshkumar, 5 K.Karthikeyan, Automation of Hydroponics Green House Farming using IOT, Middlesex University. Downloaded on September 01,2020 at 10:52:16 UTC from IEEE Xplore.

2.Shabana Parveen, Yashwanth D, Prasanth G, Keerthika M, Pooja R,Sandeep V, Gnanasudharsan A, INTELLIGENTLY OPTIMIZED SYSTEM FOR HYDROPONIC CULTIVATION.

IDEATION PHASE:
REFER:
https://github.com/IBM-EPBL/IBM-Project-23494-1659884207/blob/main/Ideation%20Phase/Ideation_Process.pdf

BRAINSTORMING:
https://github.com/IBM-EPBL/IBM-Project-23494-1659884207/blob/main/Ideation%20Phase/Brainstorm.pdf

EMPATHY MAP:
https://github.com/IBM-EPBL/IBM-Project-23494-1659884207/blob/main/Ideation%20Phase/Empathy%20map.pdf

PROBLEM SOLUTION FIT:
https://github.com/IBM-EPBL/IBM-Project-23494-1659884207/blob/main/Project%20Design%20Phase%20I/Problem%20solution%20fit.pdf

# FUNCTIONAL REQUIREMENT:

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | IoT devices | Sensors and Wifi module. |
| FR-2 | Software | Web UI, Node-red, IBM Watson, MIT app |
| | | |
| | | |
| | | |
| | | |

**Non-functional Requirements:**

Following are the non-functional requirements of the proposed solution.

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | Time consumability is less, Productivity is high. |
| NFR-2 | **Security** | It has low level of security features due to integration of sensor data. |
| NFR-3 | **Reliability** | Accuracy of data and hence it is Reliable. |
| NFR-4 | **Performance** | Performance is high and highly productive. |
| NFR-5 | **Availability** | With permitted network connectivity the application is accessible |
| NFR-6 | **Scalability** | It is perfectly scalable many new constraints can be added |

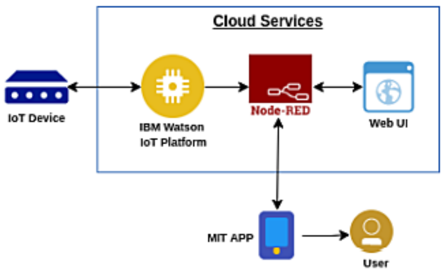
## PROJECT DESIGN AND PLAN:

### Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Example: DFD Level 0

**Example:** (Simplified)



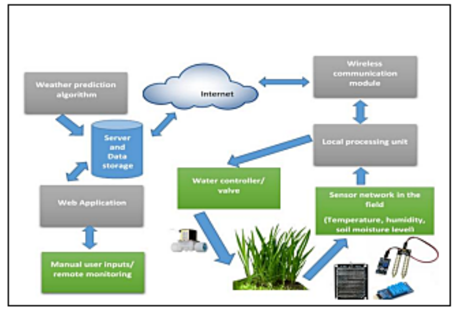




**Table-1 : Components & Technologies:**

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | How user interacts with application e.g. Web UI, Mobile App, Chatbot etc. | MIT app |
| 2. | Application Logic-1 | Logic for a process in the application | Node red/IBM Watson/MIT app |
| 3. | Application Logic-2 | Logic for a process in the application | Node red/IBM Watson/MIT app |
| 4. | Application Logic-3 | Logic for a process in the application | Node red/IBM Watson/MIT app |
| 5. | Database | Data Type, Configurations etc. | MySQL, NoSQL, etc. |
| 6. | Cloud Database | Database Service on Cloud | IBM cloud. |
| 7. | Temperature sensor | Monitors the temperature of the crop | |
| 8. | Humidity sensor | Monitors the humidity | |
| 9. | Soil moisture sensor (Tensiometers) | Monitors the soil temperature | |
| 10. | Weather sensor | Monitors the weather | . |
| 11. | Solar panel | | . |
| 12. | RTC module | Date and time configuration | |
| 13. | Relay | To get the soil moisture data | |

## PROJECT PLANNING AND SCHEDULING:

**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**

Use the below template to create product backlog and sprint schedule

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Hardware and Software | USN-1 | Sensors without wifi - Simulation. . | 2 | High | Yashwanth, Prasanth, Arun Prakash, Mugilan |
| Sprint-2 | Software | USN-2 | IBM Watson IoT platform, Workflows for IoT scenarios using Node-red | 2 | High | Yashwanth, Prasanth, Arun Prakash, Mugilan |
| Sprint-3 | MIT app | USN-3 | To develop an mobile application using MIT | 2 | High | Yashwanth, Prasanth, Arun Prakash, Mugilan |
| Sprint-4 | Web UI | USN-4 | To make the user to interact with software. | 2 | High | Yashwanth, Prasanth, Arun Prakash, Mugilan |

| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points Completed (as on Planned End Date) | Sprint Release Date (Actual) |
|---|---|---|---|---|---|---|
| Sprint-1 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 20 | 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | | 5thNOV 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | | 12thNOV 2022 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | | 14thNOV 2022 |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

**CODING AND SOLUTIONING**:

To make the user to interact with software:

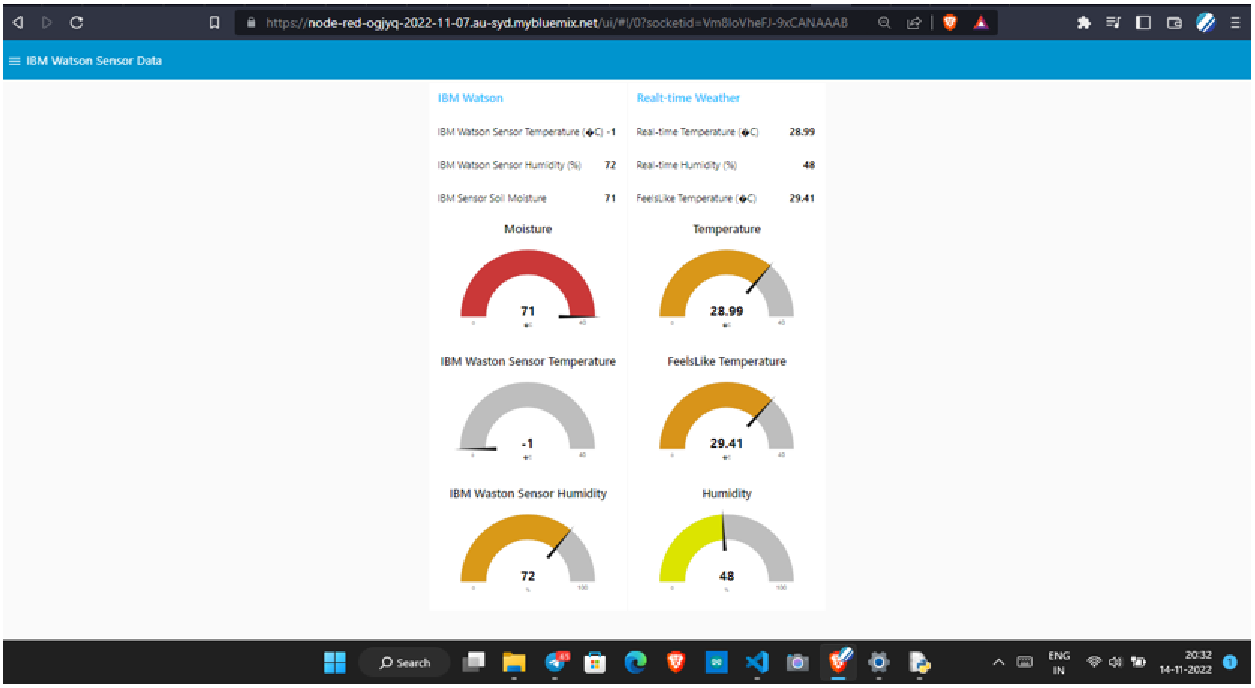
Receiving commands from IBM cloud using Python program:

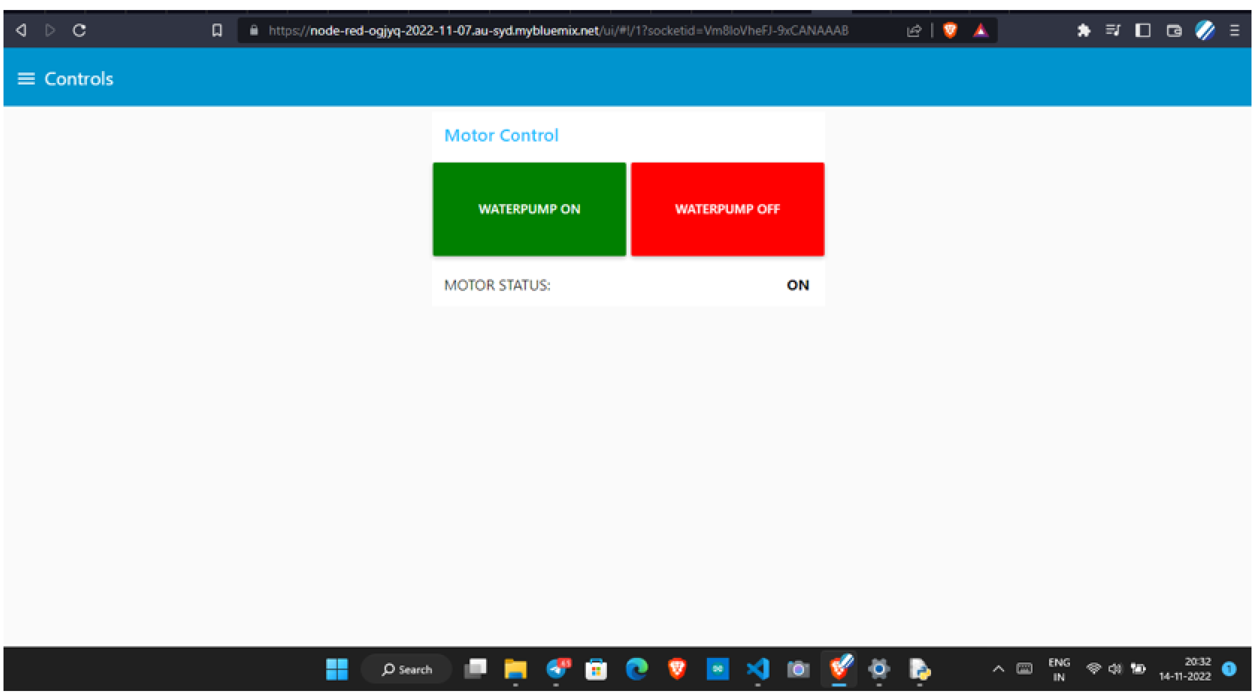
```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig = { "identity":{
"orgId": "04gt4e",
"typeId": "NodeMCU",
"deviceId": "12345"
},
"auth": {
"token": "123456789" }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig,
logHandlers=None)
client.connect ()
def myCommandCallback (cmd) :
```

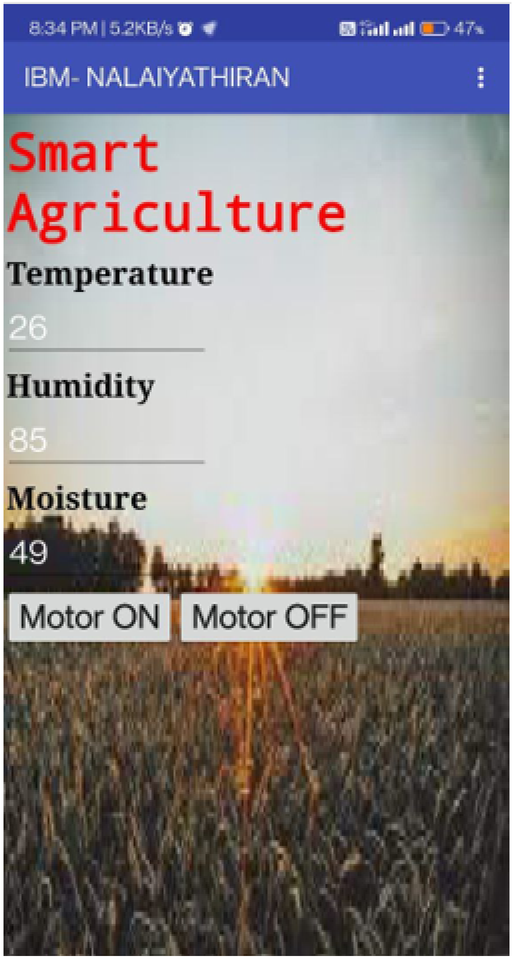
```
    print ("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    if(m=="motoron"):
        print ("Motor is switched on")
    elif(m=="motoroff"):
        print ("Motor is switched OFF")
    print(" ")

while True:
    soil=random.randint(10,100)
    temp=random.randint(-20, 125)
    hum=random.randint(0, 100)
    myData={'soil_moisture': soil, 'temperature':temp, 'humidity':hum}
    client.publishEvent (eventId="status", msgFormat="json", data=myData, qos=0 , onPublish=None)
    print("Published data Successfully: %s", myData)
    time.sleep(2)
    client.commandCallback = myCommandCallback
client.disconnect()
```

**USER INTERFACE – WEB APPLICATION:**






The purpose of this document is to briefly explain the test coverage and open issuesof the Smart Farmer - IoT Enabled Smart Farming Application project at the time of the release to User Acceptance Testing (UAT).

# 1. Defect Analysis

This report showsthe number of resolved or closed bugs at each severity level, and how they were resolved

| Resolution | Severity 1 | Severity 2 | Severity 3 | Severity 4 | Subtotal |
|---|---|---|---|---|---|
| By Design | 10 | 4 | 2 | 3 | 20 |
| Duplicate | 1 | 0 | 3 | 0 | 4 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 11 | 2 | 4 | 20 | 37 |
| Not Reproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 5 | 2 | 1 | 8 |
| Totals | 24 | 14 | 13 | 26 | 77 |

# 2. Test Case Analysis

This report shows the number of test cases that have passed, failed,and untested

| Section | Total Cases | Not Tested | Fail | Pass |
|---|---|---|---|---|
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 51 | 0 | 0 | 51 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 3 | 0 | 0 | 3 |
| Exception Reporting | 9 | 0 | 0 | 9 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

**FEATURES:**

- Comparative real time data from the internet
- Visual graph for easier understanding
- Separate tab for motor control and voice alert on commands
- SMS notification once the value falls below the threshold limit.

## Advantages & Disadvantages

**Advantages**:

• Farms can be monitored and controlled remotely.

• Increase in convenience to farmers.

• Less labour cost.

• Better standards of living.

**Disadvantages**:

• Lack of internet/connectivity issues.

• Added cost of internet and internet gateway infrastructure.

• Farmers wanted to adapt the use of Mobile App.

## Conclusion

Thus the objective of the project to implement an IOT system in order to help farmers to control and monitor their farms has been implemented successfully.

**Future Scope:**

1. Yield prediction using ML models and cloud data.
2. More UI changes and features to improve user accessibility.

Github link: https://github.com/IBM-EPBL/IBM-Project-23494-1659884207
Demo Video link:
https://drive.google.com/file/d/1pnkF51fLuXcUJbE89TZWyLG1J2X_l5Ff/view