

IoT Based Smart Crop Protection System for Agriculture

M.E.T ENGINEERING COLLEGE

(Chenbagaraamanputhooor,629 304)



NOVEMBER 2022

TEAM ID-PNT2022TMID51631

TEAM MEMBERS:

- A.Athisaya Selva Prakash (961319106003)
- B.Iswariya (961319106006)

- S.Indhumathi (961319106005)
- P.Vinithiya (961319104008)

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

1 INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2 LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References.
- 2.3 Problem Statement Definition

3 IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4 REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5 PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

7 CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

8 TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9 RESULTS

10 ADVANTAGES & DISADVANTAGES

11 CONCLUSION

12 FUTURE SCOPE

13 APPENDIX

13.1 Source Code

13.1.1 Motor.py.

13.1.2 Sensor.py

13.2 GitHub &Project Demo Link

1.INTRODUCTION

1.1 PROJECT OVERVIEW

IoT tendencies are often utilized in smart farming to boost the standard of agriculture . But our productivity remains enormously diminutive as associated to world standards . Societies after pastoral areas drift to a municipal extent for her lucrative commerce besides they can't deliberate on crofting. In detail, moderate smart irrigation systems are utilized to afford the solution for dissimilar variety of plants in spite of getting the solution for moisture related issues Weather conditions like temperature, humidity and moisture are difficult to check manually frequently overcome all these a new system is proposed constructed on cloud of Effects.Wildlife requisite overlaps personage laypeople, creating fee to inhabitants and cultivated field. Wild animals regularly ruin eminence of crops.The low productiveness is mainly due to the reasons, the crop ruined by means of untamed animals and yield ruined by way of nature object.Cultivators are experiencing numerous challenges for attaining more production due to unexpected encounters of animals, slight sorts of species, beetles, some hazardous snakes and weather circumstances. Within the existing system, electrical protection is used to give up untamed animal assaults on vegetation which leads to the death of animals.The surveillance and monitor of the tiny species, bugs and snakes are tough because of their aspect and flora of effort. A well-known wild animal safety observation that may final for a lot of Fencing is years. However, utilizing fences as a train is often. Therefore, earlier than deciding on an acceptable fence, it is vital to examine native law regulations . The high quality of fencing depends upon the material and structure. Counting on how it's made and what it's made from, some everlasting fences can last as long as 30 years. Previously buying electrical fences, it is very meaningful to be certain that they're allowed to be used in the precise area, and for defense towards endangered animal species. Further more, it is steered that electrical fences are marked with a warning signal to stop any doable human contact. Climatic conditions be keen on temperature, humidity and moisture are troublesome.

1.2 PURPOSE

The purpose is to grant monitoring device for crop safety to animal outbreaks and environment circumstances . This supports to preserve stretch and cash by dipping the physical exertion, else obligatory if the cultivators themselves have to afford guard for their crops with their endless physical administration . Wildlife regularly wreck eminence crops, because of which annual manufacturing of vegetation reduces inflicting monetary victims to cultivators . Agriculturalist suicide is huge bother due to less harvest . This low harvest is duet the circumstance of two most significant purposes i.e. Crop wrecked via untamed animals and Crop wrecked by meteorological conditions . The ranchers will treasure these SMS containing location. The prime thing of this task is to furnish a great reply to this distress . Each time either the wild animal or species are identified through PIR sensor which stimulates the web camera and gives rise to alert the buzzer in the locality, associates to the farmer direct to the cloud . When the moisture content is inferior to a terrifying level the sensor planted makes the water pumps to turn on . This ensures the complete safety of crops from animals also as from the weather conditions thus prevent the farmers .

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

IOT tendencies are often utilized in smart farming to boost the standard of agriculture . Farming the pillar of supports our country to the general commercial development. But our productivity is extremely low as associated to world standards. People from rural areas drift to an urban area for other worthwhile trades and they can't concentrate on agriculture . There are many disadvantages of the current traditional agricultural methods namely costlier and manual monitoring of the agriculture field. Specifically, small-scale smart irrigation systems are utilized to provide the solution for dissimilar variety of plants in spite of getting the solution for moisture related issues Weather conditions like temperature, humidity and moisture are difficult to check manually frequently. Farmer suicide is turning into big problem due to low productiveness amongst farms . This low productiveness is due to the fact of two main reasons, Crop ruined by means of untamed weather conditions untamed animal attacks, small types of species, insects, some hazardous snakes and weather circumstances.

2.2 REFERENCES

- 1 Krunal Mahajan¹, Riya Parate², Ekta Zade³, Shubham Khante⁴, Shishir Bagal⁵,” REVIEW PAPER ON SMART CROP PROTECTION SYSTEM”, International Research Journal of Engineering and Technology (IRJET), Volume: 08, issue 02 Feb 2021.
- 2 Dr.M. Chandra, Mohan Reddy, Keerthi Raju KamakshiKodi, BabithaAnapalliMounikaPulla, “SMART CROP PROTECTION SYSTEM FROM LIVING OBJECTS AND FIRE USING ARDUINO”, Science, Technology and Development, Volume IX Issue IX, pg.no 261- 265, Sept 2020.
- 3 Anjana, Sowmya, Charan Kumar, Monisha, Sahana, “Review on IoT in Agricultural Crop Protection and Power Generation”, International Research Journal of Engineering and Technology (IRJET) , Volume 06, Issue 11 ,Nov 2019.
- 4 G. NaveenBalaji, V. Nandhini, S. Mithra, N. Priya, R. Naveena, “IOT based smart crop monitoring in farmland”, Imperial Journal of Interdisciplinary Research (IJIR), Volume 04, Issue 01, Nov 2018.
- 5 P.Rekha, T.Saranya, P.Preethi, L.Saraswathi, G.Shobana, “Smart AGRO Using ARDUINO and GSM”, International Journal of Emerging Technologies in Engineering Research (IJETER) Volume 5, Issue 3, March 2017

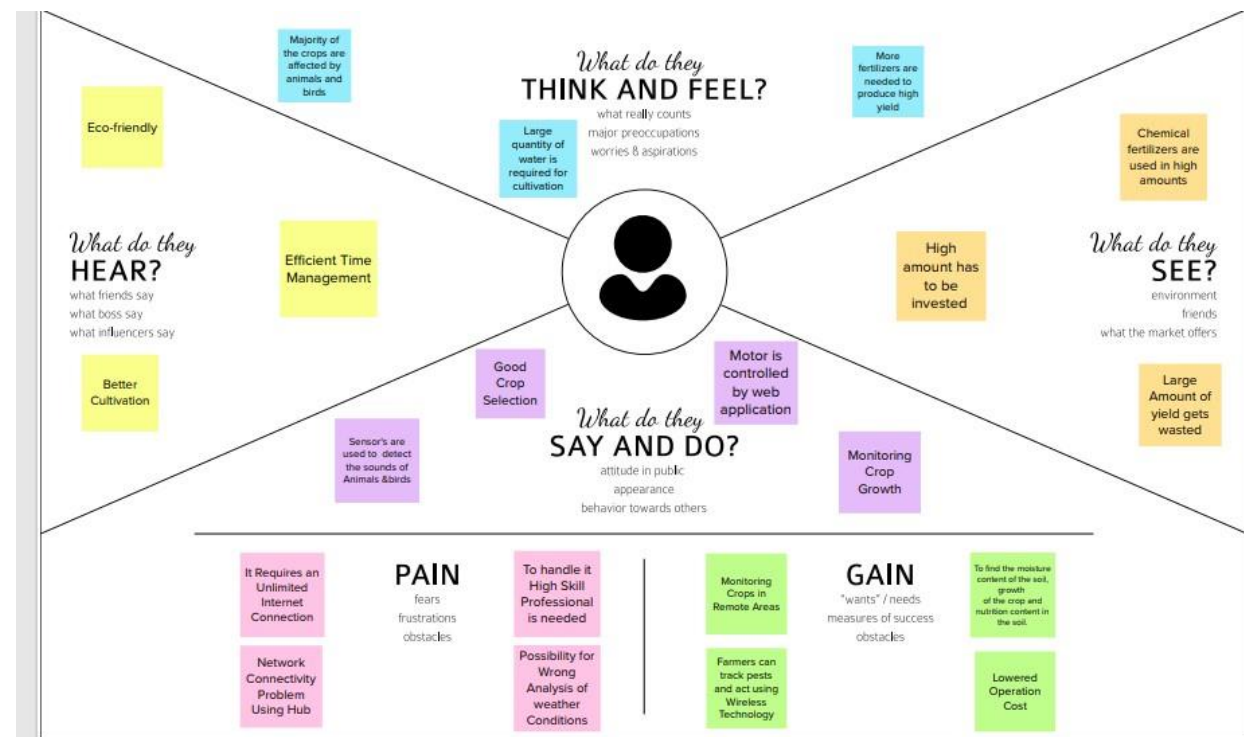
2.3 PROBLEM STATEMENT DEFINITION

Within the existing system, electrical fencing is used to give up untamed animal assaults on agricultural vegetation which leads to the death of animals . The fundamental objective is to provide a fantastic answer to this problem, so that losses incurred will be minimized and farmers will have an accurate crop yield . This low productivity is because of the fact of two most important motives i.e. Crop destroyed via untamed animals and Crop damaged by using nature object.The main objective of this assignment is to furnish a

fantastic answer to this trouble, as a result with the purpose of the economic losses incurred through the support of our farmers are minimized to get truthful crop yield .This ensures complete security of vegetation from animals and defending the farmers loss. In the proposed system Raspberry Pi, PIR sensor, web camera, ultrasonic sensor, LDR sensor, temperature sensor, humidity sensor, moisture sensor, buzzer and monitor are used . This field of this effort remains towards withdraw to monitor the system for crop security conflicting to subconscious occurrences and meteorological conditions When the moisture content is below a critical level which is determined by the sensor planted in the fields, as the system is automated the water pumps are switched on . This ensures complete safety of crops from animals also as from the weather conditions thus prevent the farmers loss.

3.IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



Agriculture is the backbone of our country that contributes to 45% of the GDP that is responsible for the enhancement of country's economy. This IOT based Crop Protection System aims on building an integrated module for improving the efficiency of the present agricultural modules. A smart way of automating farming process can be called as Smart Agriculture. Precision agriculture is one of the most famous applications of IoT in the agricultural sector and numerous organizations are leveraging this technique around the world. By implying an automated system, it is possible to eliminate threats to the crops by reducing the human intervention. The major emphasis will be on providing favourable atmosphere for plants. These agricultural automated systems will help in managing and maintain safe environment especially the agricultural areas. Environment real time Monitoring is an important factor in smart farming. Graphical User Interface based software will be provided to control the hardware system and the system will be an isolated environment, equipped with sensors like temperature sensor, humidity sensor. The controllers will be managed by a master station which will communicate with the human interactive software. This IOT based system will provide smart interface to the farmers and can increase the level of production than the current scenario.

3.2 IDEATION & BRAINSTORMING

- Implementation of water level sensor
- Using Solar panels to generate energy



- Usage of organic fertilizers to increase yield
- Usage of IR sensors to detect wild animals
- Selecting good quality seeds

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Protecting crops from <u>insects</u> , animals <u>and</u> other factors using pest <u>sprayer</u> , sound system <u>and</u> automatic drip irrigation.
2.	Idea / Solution description	Using <u>moisture meter</u> , <u>automatic</u> sprayer of <u>pesticide</u> , <u>automatic</u> DC motor <u>and</u> sensors are placed for <u>protect</u> crops.
3.	Novelty / Uniqueness	Water stagnation and scarcity <u>is</u> maintained <u>every</u> movement in field and <u>growth</u> of plants are monitored with <u>mobile</u> phone.
4.	Social Impact / Customer Satisfaction	Improved and <u>high</u> yield crops are <u>obtained</u> . <u>Farmers</u> work is reduced with automation.
5.	Business Model (Revenue Model)	This makes agriculture easier and profit is attained more by using this technique.
6.	Scalability of the Solution	This solution will <u>gives</u> high performance for proper maintenance.

3.4 PROBLEM SOLUTION FIT

1. CUSTOMER SEGMENT(S)

1. Farmers who need improved yield with smart automation will use this technique.
2. Gardeners also make this choice to improve their farm

2. JOBS-TO-BE-DONE / PROBLEMS

Which jobs-to-be-done (or problems) do you address for your

Jobs to be done

1. Setting the apparatus and maintaining.
2. Proper monitoring for energy resource.

Problems

1. Environment and social impact of automation in agriculture- This cause reduction of human empowerment.
2. Distribution- Hard to reach in remote villages.
3. Cost – Setting the system in low budget is difficult.

6. CUSTOMER

What constraints prevent your customers from taking action or limit their choices

1. Pest control over the internal process.
2. Agricultural sector lack information of high adoption in IOT .
3. For security implementation of automation ,cost are not satisfied by farmers

9. PROBLEM ROOT CAUSE

What is the real reason that this problem exists? What is the back story behind the

1. Analyzing and giving solution.
2. The most common mistake people makes when equipment error or human error is to be identified.

5. AVAILABLE SOLUTIONS

What solutions have you tried in the past? What or need to get the job done? What have they tried in the past? What have you tried in the past? What have you tried in the past?

1. Ask for customer needs and preferences
2. Offer a solution.
3. Understand the needs of farmer.
4. Pros:
Wide spread to all.
Increased profit.
5. Cons:

7. BEHAVIOUR

i.e. directly related: find the right solar panel installer, calculate usage and benefits, understand the customer's needs, find the right solution

1. Identify the troubles.
2. Understand the problems arising.
3. Make suitable choice of solutions.
4. Implement in field.
5. Monitor continuously.

3. TRIGGERS

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a

1. Through advertisements customers are triggered in automation.
2. Automation in agriculture are influenced by cinema ,government programs and by social platforms.

10. YOUR SOLUTION

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.

If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

1. Environment and social impact of automation in agriculture – make profit by innovative agriculture in smart way.
2. Distribution – make awareness in rural areas and make wider.
3. Cost – use cooling systems, high quality

8. CHANNELS of BEHAVIOUR**8.1 ONLINE**

What kind of actions do customers take online? Extract online channels from #7

This article highlights the potential of wireless sensors and IOT in agriculture, as well as challenges expected to be faced when integrating this technology

8.2 OFFLINE

What kind of actions do customers take

4. EMOTIONS: BEFORE / AFTERHow do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design

offline? Extract offline channels from #7 and use them for customer development.

1. This project will provide protection from animals

4. EMOTIONS: BEFORE / AFTER

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before

1. Crops were severely affected by extreme heat, heavy rainfall, animal grazing and other factors.

After

1. By this method, plants are protected from all factors that affect plants.

offline? Extract offline channels from #7
and use them for customer development.

1. This project will provide protection from animals through sound system.
2. Kills insects through automatic spray system.
3. We protect crops from excessive heat through bogie system.
4. Crop yield can be increased by monitoring

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

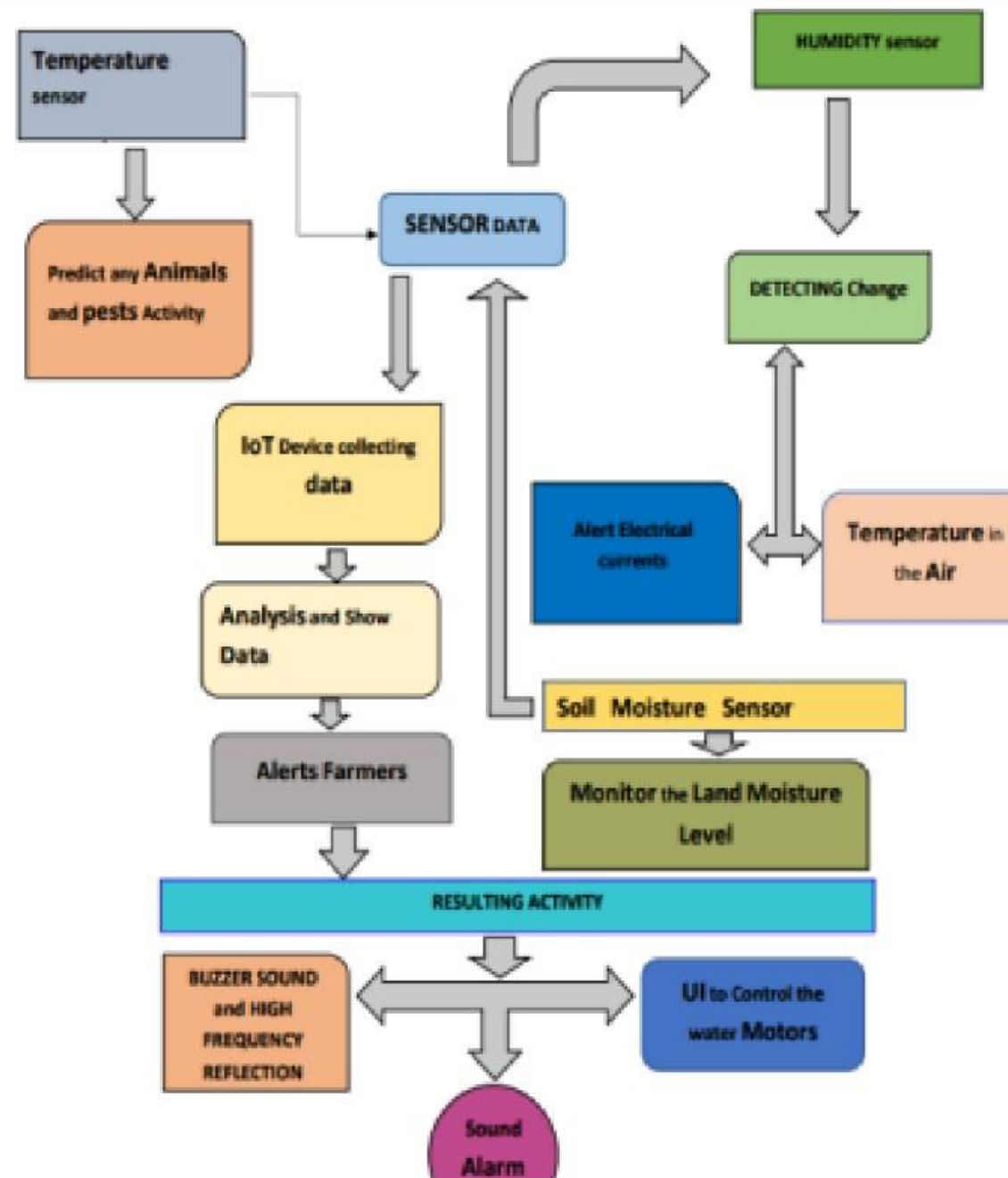
FR-NO	FUNCTIONAL REQUIREMENTS	SUB-REQUIREMENTS
FR-1	Fertilizing frame service	Documentation requirements and assisting information
FR-2	Economical service	Assisting information
FR-3	Technology assessment service	Selecting fertilizing features
FR-4	Feature assessment service	Updated technical information and machinery selection
FR-5	Information acquisition service	Assisting information about fertilizing rules
FR-6	Farm and field customizing service	Potential data acquisition service
FR-7	Field inspection	Spatial field information
FR-8	Field observation service	Analysed risks
FR-9	Assisting remote controlling	Inspecting and controlling fertilizing task
FR-10	Assisting “operational performance service”	Economical analysis of current technology

4.2 NON-FUNCTIONAL REQUIREMENT

NRF.NO	NON FUNCTIONAL REQUIREMENTS	DESCRIPTION
NRF-1	Usability	To use new technologies and increase the quantity and quality
NRF-2	Security	Protect the field from animals.
NRF-3	Reliability	Increasing the demand for food with minimum resources
NRF-4	Performance	Maintain good yield and provide sustainable quantity
NRF-5	Availability	Agricultural fences are quite an effective wild animal protection
NRF-6	Scalability	The develop system will not harmful and injurious to animals as well as human beings.

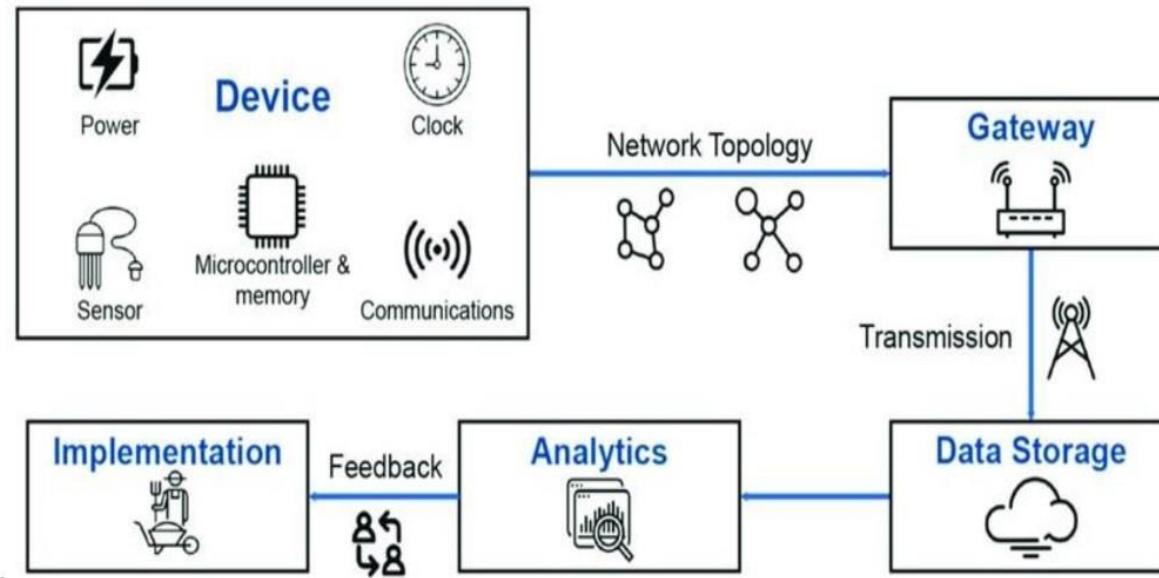
5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTIONS & TECHNICAL ARCHITECTURE





a

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Farmer)	Maintaining Fields	USN-1	As a user, I can monitor the growth of crops and protect the crops against animals	I can maintain the fields with less labor	High	Sprint-1
	Analyzing Problems	USN-2	As a user, I collect the required information about the problems on agriculture fields	I can ask my field owner directly.	Low	Sprint-2
		USN-3	As a user, I can monitor the moisture level in soil and solve the problems by using Smart IOT System	I can take remedial action immediately	High	Sprint-1
Project Designers	Identifying the problem and provide solutions	USN-4	As a user, I can sense the water level and flame in the field using sensor and monitor using IOT	I can perform this actions via IoT.	Medium	Sprint-1
		USN-5	As a user, I can make services for Irrigation, pesticides, Fertilization, and Soil preparation	I can solve this problem using IOT	High	Sprint-1
			As a user, I can monitor the field against animal attacks using a camera interface module and appropriate actions can be taken	I can monitor the field continuously.	Medium	Sprint-2
Customer (Field Maintainer)	Problem solutions	USN-6	As a user, areas can be monitored from a remote place	Checking Process	Medium	Sprint-3
	Application	USN-7	As a user, I can respond to the problems in the fields immediately	Continuous monitoring and remedial actions.	Medium	Sprint-3
	Final Process	USN-8	This proposed smart IOT-based crop protection device is found to be cost-effective and efficient	I can take necessary action if required.	Medium	Sprint-4

6.PROJECT PLANNING & SCHEDULING

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinthiya P.
Sprint-1	Login	USN-2	As a user can login using the email and password.	2	High	Athisaya Selva Prakash A Iswariya B. Indhumathi S. Vinthiya P.
Sprint-1		USN-3	Create the IBM Cloud services which are being used in this project.	6	High	Athisaya Selva Prakash A Iswariya B. Indhumathi S. Vinthiya P.
Sprint-1		USN-4	Configure the IBM Cloud services which are being used in completing this project.	4	Medium	Athisaya Selva Prakash A Iswariya B. Indhumathi S. Vinthiya P.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2		USN-5	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices, so create the IBM Watson IoT platform.	5	High	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinithiya P.
Sprint-2		USN-6	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watson IoT platform and get the device credentials.	4	Medium	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinithiya P.
Sprint-2		USN-7	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.	10	High	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinithiya P.
Sprint-3		USN-8	To create a web application create a Node-RED service.	10	High	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinithiya P.
Sprint-3		USN-9	Launch the cloudant DB and create a database to store the image URL	4	Medium	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinithiya P.

Sprint-4		USN-10	Create a cloud object storage service, create a bucket to store the images, and configure the bucket settings.	5	Medium	Athisaya Selva Prakash A. Iswariya B. Indhumathi S. Vinthiya P.
Sprint-4		USN-11	Develop a python script.	6	Medium	

PROJECT TRACKER, VELOCITY & BURNDOWN CHART: (4 MARKS)

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	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

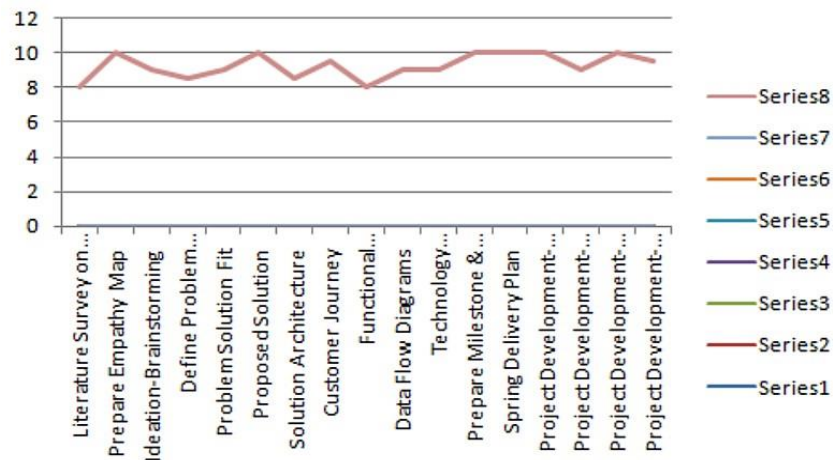
VELOCITY:

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)

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

BURNDOWN CHART:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burndown charts can be applied to any project containing measurable progress overtime



7.CODING & SOLUTIONING (EXPLAIN THE FEATURES ADDED IN THE PROJECT ALONG WITH THE CODE):

7.1 FEATURE 1

```

1 Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:00:36) [MSC v.1933 64 bit (AMD64)] on win32
2 Type "help", "copyright", "credits" or "license()" for more information.
3 import cv2
4 import numpy as np
5 import wiot.sdk.device
6 import playsound
7 import random
8 import time
9 import datetime
10 import boto3
11 from ibm_botocore.client import Config, ClientError
12
13 #CloudantDB
14 from cloudant.client import Cloudant
15 from cloudant.error import CloudantException
16 from cloudant.result import Result, ResultByKey
17 from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
18 from clarifai_grpc.grpc.api import service_pb2_grpc
19 stub = service_pb2_grpc.V2Stub(ClarifaiChannel.get_grpc_channel())
20 from clarifai_grpc.grpc.api import service_pb2, resource_pb2
21 from clarifai_grpc.grpc.api.status import status_code_pb2
22
23 #This is how you authenticate
24 metadata = (('authorization', 'key 0620e202302b4508b90eab7efe7475e4'),)
25 COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
26 COS_API_KEY_ID = "g5d4q08E1gW4TWCJ3dHfEzga1Qj0bE82AJM1AOHb"
27 COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
28 COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/c2fa2836eaf3434bbc8b5b58fefff3f0:62e450fd-4c82-4153-ba41-ccb53adb8111::"
29 clientdb = cloudant("apikey:M2nJldmwtJ016V53LAVUCqPac2aHfTn1j1xXvtdGK3Bn", "88cc5f47c1a28afbf8ad16161583f5a", url="https://d6c89f97-cf91-48b7-b14b-c99b2fe27c2f-bluemix.clouda
30 clientdb.connect()
31
32 #Create resource
33 cos = boto3.resource("s3",
34                      aws_api_key_id=COS_API_KEY_ID,
35                      aws_secret_access_key=COS_SECRET_KEY,
36                      endpoint_url=COS_ENDPOINT,
37                      region_name="jp-tok",
38                      verify=False)
39
40 #Create bucket
41 bucket = cos.create_bucket(Bucket="clarifai-grpc-api", CreateConfiguration=CreateBucketConfiguration(
42     LocationConstraint="jp-tok"))
43
44 #Upload file to bucket
45 file_path = "clarifai-grpc-api"
46 file_name = "clarifai-grpc-api"
47 file_size = 1024
48 file_data = os.urandom(file_size)
49 file_obj = io.BytesIO(file_data)
50 cos.upload_fileobj(file_obj, bucket.name, file_name)
51
52 #Download file from bucket
53 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
54 file_data = file_obj.read()
55 file_size = len(file_data)
56 print("File size: " + str(file_size))
57
58 #Delete file from bucket
59 cos.delete_object(Bucket=bucket.name, Key=file_name)
60
61 #List files in bucket
62 paginator = cos.get_paginator('list_objects_v2')
63 page_iterator = paginator.paginate(Bucket=bucket.name)
64 for page in page_iterator:
65     for obj in page.get('Contents', []):
66         print(obj['Key'])
67
68 #Get file metadata
69 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
70 meta = file_obj.meta
71 print("Metadata: " + str(meta))
72
73 #Get file content
74 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
75 file_data = file_obj.read()
76 print("File content: " + str(file_data))
77
78 #Get file size
79 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
80 file_size = file_obj.content_length
81 print("File size: " + str(file_size))
82
83 #Get file type
84 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
85 file_type = file_obj.content_type
86 print("File type: " + str(file_type))
87
88 #Get file last modified date
89 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
90 last_modified = file_obj.last_modified
91 print("Last modified: " + str(last_modified))
92
93 #Get file etag
94 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
95 etag = file_obj.e_tag
96 print("ETag: " + str(etag))
97
98 #Get file ACL
99 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
100 acl = file_obj.acl
101 print("ACL: " + str(acl))
102
103 #Get file permissions
104 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
105 permissions = file_obj.permissions
106 print("Permissions: " + str(permissions))
107
108 #Get file owner
109 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
110 owner = file_obj.owner
111 print("Owner: " + str(owner))
112
113 #Get file creator
114 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
115 creator = file_obj.creator
116 print("Creator: " + str(creator))
117
118 #Get file last accessed date
119 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
120 last_accessed = file_obj.last_accessed
121 print("Last accessed: " + str(last_accessed))
122
123 #Get file last modified date
124 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
125 last_modified = file_obj.last_modified
126 print("Last modified: " + str(last_modified))
127
128 #Get file etag
129 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
130 etag = file_obj.e_tag
131 print("ETag: " + str(etag))
132
133 #Get file ACL
134 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
135 acl = file_obj.acl
136 print("ACL: " + str(acl))
137
138 #Get file permissions
139 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
140 permissions = file_obj.permissions
141 print("Permissions: " + str(permissions))
142
143 #Get file owner
144 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
145 owner = file_obj.owner
146 print("Owner: " + str(owner))
147
148 #Get file creator
149 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
150 creator = file_obj.creator
151 print("Creator: " + str(creator))
152
153 #Get file last accessed date
154 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
155 last_accessed = file_obj.last_accessed
156 print("Last accessed: " + str(last_accessed))
157
158 #Get file last modified date
159 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
160 last_modified = file_obj.last_modified
161 print("Last modified: " + str(last_modified))
162
163 #Get file etag
164 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
165 etag = file_obj.e_tag
166 print("ETag: " + str(etag))
167
168 #Get file ACL
169 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
170 acl = file_obj.acl
171 print("ACL: " + str(acl))
172
173 #Get file permissions
174 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
175 permissions = file_obj.permissions
176 print("Permissions: " + str(permissions))
177
178 #Get file owner
179 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
180 owner = file_obj.owner
181 print("Owner: " + str(owner))
182
183 #Get file creator
184 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
185 creator = file_obj.creator
186 print("Creator: " + str(creator))
187
188 #Get file last accessed date
189 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
190 last_accessed = file_obj.last_accessed
191 print("Last accessed: " + str(last_accessed))
192
193 #Get file last modified date
194 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
195 last_modified = file_obj.last_modified
196 print("Last modified: " + str(last_modified))
197
198 #Get file etag
199 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
200 etag = file_obj.e_tag
201 print("ETag: " + str(etag))
202
203 #Get file ACL
204 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
205 acl = file_obj.acl
206 print("ACL: " + str(acl))
207
208 #Get file permissions
209 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
210 permissions = file_obj.permissions
211 print("Permissions: " + str(permissions))
212
213 #Get file owner
214 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
215 owner = file_obj.owner
216 print("Owner: " + str(owner))
217
218 #Get file creator
219 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
220 creator = file_obj.creator
221 print("Creator: " + str(creator))
222
223 #Get file last accessed date
224 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
225 last_accessed = file_obj.last_accessed
226 print("Last accessed: " + str(last_accessed))
227
228 #Get file last modified date
229 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
230 last_modified = file_obj.last_modified
231 print("Last modified: " + str(last_modified))
232
233 #Get file etag
234 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
235 etag = file_obj.e_tag
236 print("ETag: " + str(etag))
237
238 #Get file ACL
239 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
240 acl = file_obj.acl
241 print("ACL: " + str(acl))
242
243 #Get file permissions
244 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
245 permissions = file_obj.permissions
246 print("Permissions: " + str(permissions))
247
248 #Get file owner
249 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
250 owner = file_obj.owner
251 print("Owner: " + str(owner))
252
253 #Get file creator
254 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
255 creator = file_obj.creator
256 print("Creator: " + str(creator))
257
258 #Get file last accessed date
259 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
260 last_accessed = file_obj.last_accessed
261 print("Last accessed: " + str(last_accessed))
262
263 #Get file last modified date
264 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
265 last_modified = file_obj.last_modified
266 print("Last modified: " + str(last_modified))
267
268 #Get file etag
269 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
270 etag = file_obj.e_tag
271 print("ETag: " + str(etag))
272
273 #Get file ACL
274 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
275 acl = file_obj.acl
276 print("ACL: " + str(acl))
277
278 #Get file permissions
279 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
280 permissions = file_obj.permissions
281 print("Permissions: " + str(permissions))
282
283 #Get file owner
284 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
285 owner = file_obj.owner
286 print("Owner: " + str(owner))
287
288 #Get file creator
289 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
290 creator = file_obj.creator
291 print("Creator: " + str(creator))
292
293 #Get file last accessed date
294 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
295 last_accessed = file_obj.last_accessed
296 print("Last accessed: " + str(last_accessed))
297
298 #Get file last modified date
299 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
300 last_modified = file_obj.last_modified
301 print("Last modified: " + str(last_modified))
302
303 #Get file etag
304 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
305 etag = file_obj.e_tag
306 print("ETag: " + str(etag))
307
308 #Get file ACL
309 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
310 acl = file_obj.acl
311 print("ACL: " + str(acl))
312
313 #Get file permissions
314 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
315 permissions = file_obj.permissions
316 print("Permissions: " + str(permissions))
317
318 #Get file owner
319 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
320 owner = file_obj.owner
321 print("Owner: " + str(owner))
322
323 #Get file creator
324 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
325 creator = file_obj.creator
326 print("Creator: " + str(creator))
327
328 #Get file last accessed date
329 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
330 last_accessed = file_obj.last_accessed
331 print("Last accessed: " + str(last_accessed))
332
333 #Get file last modified date
334 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
335 last_modified = file_obj.last_modified
336 print("Last modified: " + str(last_modified))
337
338 #Get file etag
339 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
340 etag = file_obj.e_tag
341 print("ETag: " + str(etag))
342
343 #Get file ACL
344 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
345 acl = file_obj.acl
346 print("ACL: " + str(acl))
347
348 #Get file permissions
349 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
350 permissions = file_obj.permissions
351 print("Permissions: " + str(permissions))
352
353 #Get file owner
354 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
355 owner = file_obj.owner
356 print("Owner: " + str(owner))
357
358 #Get file creator
359 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
360 creator = file_obj.creator
361 print("Creator: " + str(creator))
362
363 #Get file last accessed date
364 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
365 last_accessed = file_obj.last_accessed
366 print("Last accessed: " + str(last_accessed))
367
368 #Get file last modified date
369 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
370 last_modified = file_obj.last_modified
371 print("Last modified: " + str(last_modified))
372
373 #Get file etag
374 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
375 etag = file_obj.e_tag
376 print("ETag: " + str(etag))
377
378 #Get file ACL
379 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
380 acl = file_obj.acl
381 print("ACL: " + str(acl))
382
383 #Get file permissions
384 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
385 permissions = file_obj.permissions
386 print("Permissions: " + str(permissions))
387
388 #Get file owner
389 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
390 owner = file_obj.owner
391 print("Owner: " + str(owner))
392
393 #Get file creator
394 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
395 creator = file_obj.creator
396 print("Creator: " + str(creator))
397
398 #Get file last accessed date
399 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
400 last_accessed = file_obj.last_accessed
401 print("Last accessed: " + str(last_accessed))
402
403 #Get file last modified date
404 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
405 last_modified = file_obj.last_modified
406 print("Last modified: " + str(last_modified))
407
408 #Get file etag
409 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
410 etag = file_obj.e_tag
411 print("ETag: " + str(etag))
412
413 #Get file ACL
414 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
415 acl = file_obj.acl
416 print("ACL: " + str(acl))
417
418 #Get file permissions
419 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
420 permissions = file_obj.permissions
421 print("Permissions: " + str(permissions))
422
423 #Get file owner
424 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
425 owner = file_obj.owner
426 print("Owner: " + str(owner))
427
428 #Get file creator
429 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
430 creator = file_obj.creator
431 print("Creator: " + str(creator))
432
433 #Get file last accessed date
434 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
435 last_accessed = file_obj.last_accessed
436 print("Last accessed: " + str(last_accessed))
437
438 #Get file last modified date
439 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
440 last_modified = file_obj.last_modified
441 print("Last modified: " + str(last_modified))
442
443 #Get file etag
444 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
445 etag = file_obj.e_tag
446 print("ETag: " + str(etag))
447
448 #Get file ACL
449 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
450 acl = file_obj.acl
451 print("ACL: " + str(acl))
452
453 #Get file permissions
454 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
455 permissions = file_obj.permissions
456 print("Permissions: " + str(permissions))
457
458 #Get file owner
459 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
460 owner = file_obj.owner
461 print("Owner: " + str(owner))
462
463 #Get file creator
464 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
465 creator = file_obj.creator
466 print("Creator: " + str(creator))
467
468 #Get file last accessed date
469 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
470 last_accessed = file_obj.last_accessed
471 print("Last accessed: " + str(last_accessed))
472
473 #Get file last modified date
474 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
475 last_modified = file_obj.last_modified
476 print("Last modified: " + str(last_modified))
477
478 #Get file etag
479 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
480 etag = file_obj.e_tag
481 print("ETag: " + str(etag))
482
483 #Get file ACL
484 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
485 acl = file_obj.acl
486 print("ACL: " + str(acl))
487
488 #Get file permissions
489 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
490 permissions = file_obj.permissions
491 print("Permissions: " + str(permissions))
492
493 #Get file owner
494 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
495 owner = file_obj.owner
496 print("Owner: " + str(owner))
497
498 #Get file creator
499 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
500 creator = file_obj.creator
501 print("Creator: " + str(creator))
502
503 #Get file last accessed date
504 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
505 last_accessed = file_obj.last_accessed
506 print("Last accessed: " + str(last_accessed))
507
508 #Get file last modified date
509 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
510 last_modified = file_obj.last_modified
511 print("Last modified: " + str(last_modified))
512
513 #Get file etag
514 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
515 etag = file_obj.e_tag
516 print("ETag: " + str(etag))
517
518 #Get file ACL
519 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
520 acl = file_obj.acl
521 print("ACL: " + str(acl))
522
523 #Get file permissions
524 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
525 permissions = file_obj.permissions
526 print("Permissions: " + str(permissions))
527
528 #Get file owner
529 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
530 owner = file_obj.owner
531 print("Owner: " + str(owner))
532
533 #Get file creator
534 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
535 creator = file_obj.creator
536 print("Creator: " + str(creator))
537
538 #Get file last accessed date
539 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
540 last_accessed = file_obj.last_accessed
541 print("Last accessed: " + str(last_accessed))
542
543 #Get file last modified date
544 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
545 last_modified = file_obj.last_modified
546 print("Last modified: " + str(last_modified))
547
548 #Get file etag
549 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
550 etag = file_obj.e_tag
551 print("ETag: " + str(etag))
552
553 #Get file ACL
554 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
555 acl = file_obj.acl
556 print("ACL: " + str(acl))
557
558 #Get file permissions
559 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
560 permissions = file_obj.permissions
561 print("Permissions: " + str(permissions))
562
563 #Get file owner
564 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
565 owner = file_obj.owner
566 print("Owner: " + str(owner))
567
568 #Get file creator
569 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
570 creator = file_obj.creator
571 print("Creator: " + str(creator))
572
573 #Get file last accessed date
574 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
575 last_accessed = file_obj.last_accessed
576 print("Last accessed: " + str(last_accessed))
577
578 #Get file last modified date
579 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
580 last_modified = file_obj.last_modified
581 print("Last modified: " + str(last_modified))
582
583 #Get file etag
584 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
585 etag = file_obj.e_tag
586 print("ETag: " + str(etag))
587
588 #Get file ACL
589 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
590 acl = file_obj.acl
591 print("ACL: " + str(acl))
592
593 #Get file permissions
594 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
595 permissions = file_obj.permissions
596 print("Permissions: " + str(permissions))
597
598 #Get file owner
599 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
600 owner = file_obj.owner
601 print("Owner: " + str(owner))
602
603 #Get file creator
604 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
605 creator = file_obj.creator
606 print("Creator: " + str(creator))
607
608 #Get file last accessed date
609 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
610 last_accessed = file_obj.last_accessed
611 print("Last accessed: " + str(last_accessed))
612
613 #Get file last modified date
614 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
615 last_modified = file_obj.last_modified
616 print("Last modified: " + str(last_modified))
617
618 #Get file etag
619 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
620 etag = file_obj.e_tag
621 print("ETag: " + str(etag))
622
623 #Get file ACL
624 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
625 acl = file_obj.acl
626 print("ACL: " + str(acl))
627
628 #Get file permissions
629 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
630 permissions = file_obj.permissions
631 print("Permissions: " + str(permissions))
632
633 #Get file owner
634 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
635 owner = file_obj.owner
636 print("Owner: " + str(owner))
637
638 #Get file creator
639 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
640 creator = file_obj.creator
641 print("Creator: " + str(creator))
642
643 #Get file last accessed date
644 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
645 last_accessed = file_obj.last_accessed
646 print("Last accessed: " + str(last_accessed))
647
648 #Get file last modified date
649 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
650 last_modified = file_obj.last_modified
651 print("Last modified: " + str(last_modified))
652
653 #Get file etag
654 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
655 etag = file_obj.e_tag
656 print("ETag: " + str(etag))
657
658 #Get file ACL
659 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
660 acl = file_obj.acl
661 print("ACL: " + str(acl))
662
663 #Get file permissions
664 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
665 permissions = file_obj.permissions
666 print("Permissions: " + str(permissions))
667
668 #Get file owner
669 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
670 owner = file_obj.owner
671 print("Owner: " + str(owner))
672
673 #Get file creator
674 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
675 creator = file_obj.creator
676 print("Creator: " + str(creator))
677
678 #Get file last accessed date
679 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
680 last_accessed = file_obj.last_accessed
681 print("Last accessed: " + str(last_accessed))
682
683 #Get file last modified date
684 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
685 last_modified = file_obj.last_modified
686 print("Last modified: " + str(last_modified))
687
688 #Get file etag
689 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
690 etag = file_obj.e_tag
691 print("ETag: " + str(etag))
692
693 #Get file ACL
694 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
695 acl = file_obj.acl
696 print("ACL: " + str(acl))
697
698 #Get file permissions
699 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
700 permissions = file_obj.permissions
701 print("Permissions: " + str(permissions))
702
703 #Get file owner
704 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
705 owner = file_obj.owner
706 print("Owner: " + str(owner))
707
708 #Get file creator
709 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
710 creator = file_obj.creator
711 print("Creator: " + str(creator))
712
713 #Get file last accessed date
714 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
715 last_accessed = file_obj.last_accessed
716 print("Last accessed: " + str(last_accessed))
717
718 #Get file last modified date
719 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
720 last_modified = file_obj.last_modified
721 print("Last modified: " + str(last_modified))
722
723 #Get file etag
724 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
725 etag = file_obj.e_tag
726 print("ETag: " + str(etag))
727
728 #Get file ACL
729 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
730 acl = file_obj.acl
731 print("ACL: " + str(acl))
732
733 #Get file permissions
734 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
735 permissions = file_obj.permissions
736 print("Permissions: " + str(permissions))
737
738 #Get file owner
739 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
740 owner = file_obj.owner
741 print("Owner: " + str(owner))
742
743 #Get file creator
744 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
745 creator = file_obj.creator
746 print("Creator: " + str(creator))
747
748 #Get file last accessed date
749 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
750 last_accessed = file_obj.last_accessed
751 print("Last accessed: " + str(last_accessed))
752
753 #Get file last modified date
754 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
755 last_modified = file_obj.last_modified
756 print("Last modified: " + str(last_modified))
757
758 #Get file etag
759 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
760 etag = file_obj.e_tag
761 print("ETag: " + str(etag))
762
763 #Get file ACL
764 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
765 acl = file_obj.acl
766 print("ACL: " + str(acl))
767
768 #Get file permissions
769 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
770 permissions = file_obj.permissions
771 print("Permissions: " + str(permissions))
772
773 #Get file owner
774 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
775 owner = file_obj.owner
776 print("Owner: " + str(owner))
777
778 #Get file creator
779 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
780 creator = file_obj.creator
781 print("Creator: " + str(creator))
782
783 #Get file last accessed date
784 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
785 last_accessed = file_obj.last_accessed
786 print("Last accessed: " + str(last_accessed))
787
788 #Get file last modified date
789 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
790 last_modified = file_obj.last_modified
791 print("Last modified: " + str(last_modified))
792
793 #Get file etag
794 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
795 etag = file_obj.e_tag
796 print("ETag: " + str(etag))
797
798 #Get file ACL
799 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
800 acl = file_obj.acl
801 print("ACL: " + str(acl))
802
803 #Get file permissions
804 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
805 permissions = file_obj.permissions
806 print("Permissions: " + str(permissions))
807
808 #Get file owner
809 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
810 owner = file_obj.owner
811 print("Owner: " + str(owner))
812
813 #Get file creator
814 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
815 creator = file_obj.creator
816 print("Creator: " + str(creator))
817
818 #Get file last accessed date
819 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
820 last_accessed = file_obj.last_accessed
821 print("Last accessed: " + str(last_accessed))
822
823 #Get file last modified date
824 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
825 last_modified = file_obj.last_modified
826 print("Last modified: " + str(last_modified))
827
828 #Get file etag
829 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
830 etag = file_obj.e_tag
831 print("ETag: " + str(etag))
832
833 #Get file ACL
834 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
835 acl = file_obj.acl
836 print("ACL: " + str(acl))
837
838 #Get file permissions
839 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
840 permissions = file_obj.permissions
841 print("Permissions: " + str(permissions))
842
843 #Get file owner
844 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
845 owner = file_obj.owner
846 print("Owner: " + str(owner))
847
848 #Get file creator
849 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
850 creator = file_obj.creator
851 print("Creator: " + str(creator))
852
853 #Get file last accessed date
854 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
855 last_accessed = file_obj.last_accessed
856 print("Last accessed: " + str(last_accessed))
857
858 #Get file last modified date
859 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
860 last_modified = file_obj.last_modified
861 print("Last modified: " + str(last_modified))
862
863 #Get file etag
864 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
865 etag = file_obj.e_tag
866 print("ETag: " + str(etag))
867
868 #Get file ACL
869 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
870 acl = file_obj.acl
871 print("ACL: " + str(acl))
872
873 #Get file permissions
874 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
875 permissions = file_obj.permissions
876 print("Permissions: " + str(permissions))
877
878 #Get file owner
879 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
880 owner = file_obj.owner
881 print("Owner: " + str(owner))
882
883 #Get file creator
884 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
885 creator = file_obj.creator
886 print("Creator: " + str(creator))
887
888 #Get file last accessed date
889 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
890 last_accessed = file_obj.last_accessed
891 print("Last accessed: " + str(last_accessed))
892
893 #Get file last modified date
894 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
895 last_modified = file_obj.last_modified
896 print("Last modified: " + str(last_modified))
897
898 #Get file etag
899 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
900 etag = file_obj.e_tag
901 print("ETag: " + str(etag))
902
903 #Get file ACL
904 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
905 acl = file_obj.acl
906 print("ACL: " + str(acl))
907
908 #Get file permissions
909 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
910 permissions = file_obj.permissions
911 print("Permissions: " + str(permissions))
912
913 #Get file owner
914 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
915 owner = file_obj.owner
916 print("Owner: " + str(owner))
917
918 #Get file creator
919 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
920 creator = file_obj.creator
921 print("Creator: " + str(creator))
922
923 #Get file last accessed date
924 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
925 last_accessed = file_obj.last_accessed
926 print("Last accessed: " + str(last_accessed))
927
928 #Get file last modified date
929 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
930 last_modified = file_obj.last_modified
931 print("Last modified: " + str(last_modified))
932
933 #Get file etag
934 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
935 etag = file_obj.e_tag
936 print("ETag: " + str(etag))
937
938 #Get file ACL
939 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
940 acl = file_obj.acl
941 print("ACL: " + str(acl))
942
943 #Get file permissions
944 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
945 permissions = file_obj.permissions
946 print("Permissions: " + str(permissions))
947
948 #Get file owner
949 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
950 owner = file_obj.owner
951 print("Owner: " + str(owner))
952
953 #Get file creator
954 file_obj = cos.get_object(Bucket=bucket.name, Key=file_name)
955 creator = file_obj.creator
956 print("Creator: " + str(creator))
957
958 #Get
```

```

37         config=Config(signature_version="oauth"),
38         endpoint_url=COS_ENDPOINT
39     )
40 def multi_part_upload(bucket_name, item_name, file_path):
41     try:
42         print("Starting file transfer for {0} to bucket: {1}\n".format(item_name, bucket_name))
43         #set 5 MB chunks
44         part_size = 1024 * 1024 * 5
45         #set threshold to 15 MB
46         file_threshold = 1024 * 1024 * 15
47         #set the transfer threshold and chunk size
48         transfer_config = ibm_boto3.s3.transfer.TransferConfig(
49             multipart_threshold=file_threshold,
50             multipart_chunksize=part_size
51         )
52         #the upload_fileobj method will automatically execute a multi-part upload
53         #in 5 MB chunks size
54         with open(file_path, "rb") as file_data:
55             cos.Object(bucket_name, item_name).upload_fileobj(
56                 Fileobj=file_data,
57                 Config=transfer_config
58             )
59             print("Transfer for {0} Complete!\n".format(item_name))
60     except ClientError as be:
61         print("CLIENT ERROR: {0}\n".format(be))
62     except Exception as e:
63         print("Unable to complete multi-part upload: {0}".format(e))
64
65 def myCommandCallback(cmd):
66     print("Command received: %s" % cmd.data)
67     command=cmd.data['command']
68     print(command)
69     if(command=="lighton"):
70         print('lighton')
71     elif(command=="lightoff"):
72         print('lightoff')
73     elif(command=="motoron"):
74         print('motoron')

```

```

74     print('motoron')
75     elif(command=="motoroff"):
76         print('motoroff')
77     myConfig = {
78         "identity": {
79             "orgId": "chytun",
80             "typeId": "NodeMCU",
81             "deviceId": "12345"
82         },
83         "auth": {
84             "token": "12345678"
85         }
86     }
87     client = wiot.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
88     client.connect()
89
90     database_name = "sample"
91     my_database = clientdb.create_database(database_name)
92     if my_database.exists():
93         print(f'({database_name}) successfully created.')
94     cap=cv2.VideoCapture("garden.mp4")
95     if(cap.isOpened()==True):
96         print('File opened')
97     else:
98         print('File not found')
99
100    while(cap.isOpened()):
101        ret, frame = cap.read()
102        gray = cv3.cvtColor(frame, cv2.COLOR_BGR2GRAY)
103        imS= cv2.resize(frame, (960,540))
104        cv2.imwrite('ex.jpg',imS)
105        with open("ex.jpg", "rb") as f:
106            file_bytes = f.read()
107        #This is the model ID of a publicly available General model. You may use any other public or custom model ID.
108        request = service_pb2.PostModelOutputsRequest(
109            model_id='e9359dbe6ee44dbc8842ebe97247b201',
110            inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.Image(base64=file_bytes))

```

```

110    ...     inputs=[resources_pb2.Input(data=resources_pb2.Data(image=resources_pb2.Image(base64=file_bytes))
111    ...     ))
112    ...     response = stub.PostModelOutputs(request, metadata=metadata)
113    ...     if response.status.code != status_code_pb2.SUCCESS:
114    ...         raise Exception("Request failed, status code: " + str(response.status.code))
115    ...     detect=False
116    ...     for concept in response.outputs[0].data.concepts:
117    ...         #print('%12s: %.f' % (concept.name, concept.value))
118    ...         if(concept.value>0.98):
119    ...             #print(concept.name)
120    ...             if(concept.name=="animal"):
121    ...                 print("Alert! Alert! animal detected")
122    ...                 playsound.playsound('alert.mp3')
123    ...                 picname=datetime.datetime.now().strftime("%Y-%m-%d-%H-%M")
124    ...                 cv2.imwrite(picname+'.jpg',frame)
125    ...                 multi_part_upload('Dhakshesh', picname+'.jpg', picname+'.jpg')
126    ...                 json_document={"link":COS_ENDPOINT+'/'+'Dhakshesh+'/'+'picname+'.jpg'}
127    ...                 new_document = my_database.create_document(json_document)
128    ...                 if new_document.exists():
129    ...                     print(f"Document successfully created.")
130    ...                     time.sleep(5)
131    ...                     detect=True
132    ...     moist=random.randint(0,100)
133    ...     humidity=random.randint(0,100)
134    ...     myData={'Animal':detect,'moisture':moist,'humidity':humidity}
135    ...     print(myData)
136    ...     if(humidity>90):

```

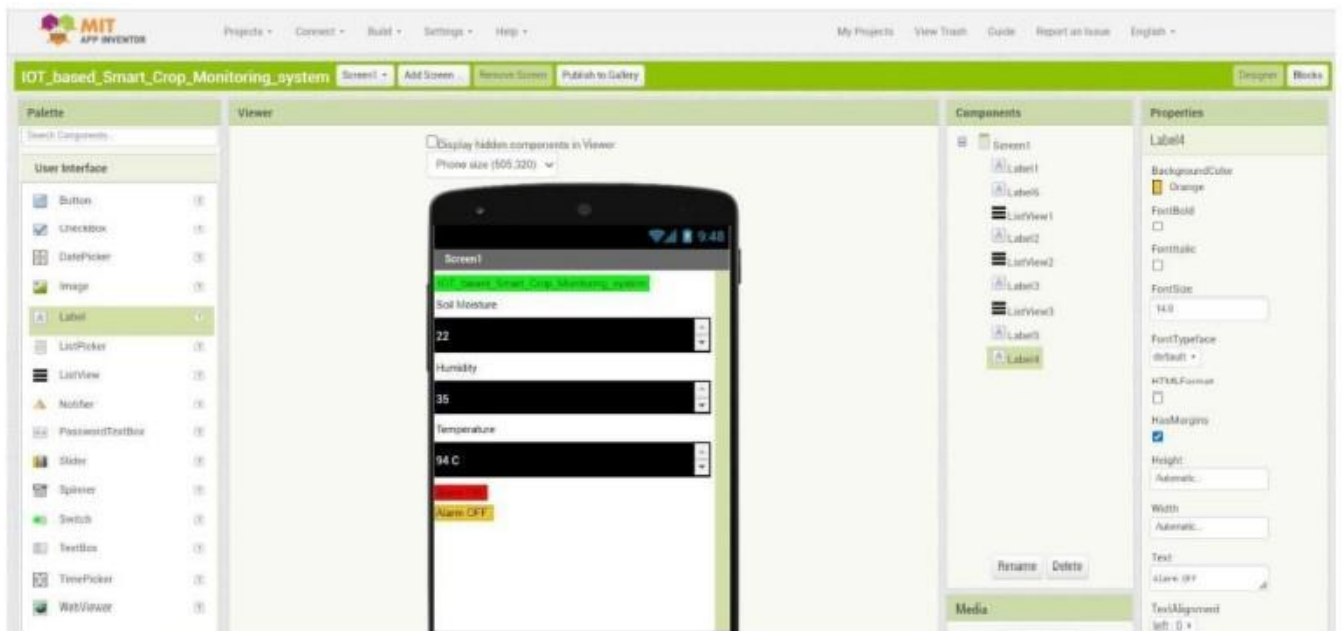
```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/HP/Desktop/crop/crop_protect.py =====
2021-04-06 12:52:19,640 wiotp.sdk.device.client.DeviceClient INFO Connecte
d successfully: d:hj5fmy:NodeMCU:12345
'sample' successfully created.
File opened
{'Animal': False, 'moisture': 17, 'humidity': 41}
Publish Ok..
{'Animal': False, 'moisture': 84, 'humidity': 16}
Publish Ok..
{'Animal': False, 'moisture': 48, 'humidity': 43}
Publish Ok..
{'Animal': False, 'moisture': 0, 'humidity': 3}
Publish Ok..
{'Animal': False, 'moisture': 73, 'humidity': 68}
Publish Ok..
{'Animal': False, 'moisture': 26, 'humidity': 26}
Publish Ok..
{'Animal': False, 'moisture': 96, 'humidity': 59}
Publish Ok..
I
```

7.2 FEATURE 2

MIT APP INVENTOR TO DESIGN THE APP



CUSTOMIZING THE APP INTERFACE TO DISPLAY THE VALUES:



8. TESTING

8.1.TEST CASES

Defect Analysis

This report shows the 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	11	4	2	2	19
Duplicate	1	1	2	0	4
External	2	3	0	1	6
Fixed	10	2	3	20	35
Not Reproduced	0	0	2	0	2
Skipped	0	0	2	1	3
Won't Fix	0	5	2	1	8
Totals	24	15	13	25	77

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	5	0	1	4
Client Application	47	0	2	45

Security	3	0	0	3
Outsource Shipping	2	0	0	2
Exception Reporting	11	0	2	9
Final Report Output	5	0	0	5
Version Control	3	0	1	2

9.RESULTS

Thus the IOT based Smart Crop Protection has been build successfully with the help of MIT app, Node.Js, and node red. And the output has been tested and verified using MIT app.

The problem of crop vandalization by wild animals and fire has become a majorsocial problem in currenttime.

It requires urgent attention as no effectivesolution exists till date for this problem. Thus this project carries a great social relevance as it aims to address this problem. This project willhelp farmers in protecting their orchards and fields and save them from significant financial losses and will save them from the unproductive efforts that they endure for the protection their fields. This will also helpthem in achievingbetter crop yields thus leading to theireconomic wellbeing.

10.ADVANTAGES AND DISADVANTAGES

Advantage:

Controllable food supply.you might have droughts or floods, but if you are growing the crops and breeding them to be hardier, you have a better chanceof not straving. It allows farmers to maximize yields using minimumresources such as water,fertilizers.

Disadvantage:

The main disadvantage is the time it can take to process the information.in order to keep feeding people as the population grows you have to radically change theenvironment of the planet.

11.CONCLUSION:

The aim of this project is to make the life and work of the farmer much easier. This can be achieved using the technique - Precision Farming, this involves autonomous monitoring of crops and other environmental parameters which has an effect on the crop, these environmental conditions are:

1. Environmental Humidity
2. Environmental Temperature.
3. Soil Moisture.
4. Rain Sensing.

Above mentioned are some of the conditions monitored autonomously, threshold parameters for various crops are automatically set upon user input of crop variety to be monitored. By this system one could achieve a good yield and better nutritional crops in their agricultural produce.

12. FUTURE SCOPE:

Future scope of our project relies on the farmers and their feedbacks, in future we are planning to add the following features:

1. One device one farm - Cover the entire farm area with a single device.

2. Pest monitoring system.
3. Estimated yield calculator.
4. Estimated time of cultivation.
5. Individual cloud management dashboard.

13.APPENDIX

SOURCE CODE MOTOR.PY impor

```
time import sys import
```

```
ibmiotf.application # to install pip
```

```
install ibmiotf import ibmiotf.device
```

```
# Provide your IBM Watson Device  
Credentials organization = "63004g"
```

```
# replace the ORG ID deviceType =  
"MainDevice" # replace the Device  
type deviceId = "9344022806" #  
replace Device ID authMethod =  
"token"
```

```
authToken = "a-63004g-86womzydrf" # Replace the authtoken
```

```
def myCommandCallback(cmd): # function for
```

```
    Callback if cmd.data['command'] ==
```

```
    'motoron':
```

```
        print("MOTOR ON IS RECEIVED")
```

```
    elif cmd.data['command'] ==
```

```
        'motoroff':
```

```
            print("MOTOR OFF  
            IS RECEIVED")
```

```
    if cmd.command ==
```

```
"setInterval":
```

```
if 'interval'
```

```
not in
```

```
cmd.data:
```

```
    print("Error - command is missing required information:  
            'interval'")
```

```
el
```

```
    s
```

```
    e
```

```
    :
```

```
    i
```

```
    n
```

```
    t
```

```
    e
```

```
    r
```

```
    v
```

```
    a
```

```
    l
```

```
    =
```

```
cmd.data['interval'] elif
```

```
cmd.command ==
```

```
"print": if 'message' not in cmd.data: print("Error - command  
is missing required information: 'message'")
```

```

else:
    output =
    cmd.data['message
    ']' print(output)

```

```

t
r
y
:
    deviceOptions = {"org": organization, "type": deviceType,
    "id": deviceId, "auth- method": authMethod,
        "auth-token": authToken}
    deviceCli =
    ibmiotf.device.Client(deviceOptions) #
    .....
except
    Exception
    as e:
        print("Caught exception connecting
device: %s" % str(e)) sys.exit()
    # Connect and send a datapoint "hello" with value "world" into
    the cloud as an event of type "greeting" 10 times
    deviceCli.connect()

```

```
while True:

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application
from the cloud deviceCli.disconnect()
```

SENSOR.PY

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
```

```
#Provide your IBM Watson Device
Credentials organization = "63004g"
deviceType = "MainDevice"
deviceId =
    "9344022806"
authMethod =
    "token"
authToken =
    "9944611970"
```

```
# Initialize GPIO def myCommandCallback(cmd):
print("Command received: %s" %
cmd.data['command'])
```

```

status=cmd.data['command'] if status ==
"motoron": print ("motor is on") elif status ==
"motoroff": print ("motor is off")
else : print ("please send
proper command")

```

```

try:
    deviceOptions = {"org": organization, "type": deviceType, "id":
                                                              deviceId,
    "auth- method":
authMethod, "auth-
token": authToken}
    deviceCli =
ibmiotf.device.Client(devi
ceOptions)
    #.....
except Exception as e: print("Caught
exception connecting device: %s" % str(e))
sys.exit()

```

```
# Connect and send a datapoint "hello" with value "world" into  
the cloud as an event of type "greeting" 10 times  
deviceCli.connect()
```

```
while True:
```

```
    #Get Sensor Data
```

```
    from DHT11
```

```
    animal=random.uni
```

```
    form(0.
```

```
    1, 0.99)
```

```
    moisture=random.randi
```

```
    nt(0
```

```
    ,110)
```

```
    temperature=random.
```

```
    randin t(-20,125)
```

```
    Humid=random.randint(0,1
```

```
    00)
```

```
    data = {'animal':animal,'moisture': moisture, 'temperature' :
```

```
    temperature, 'Humid': Humid } #print data
```

```
    def myOnPublishCallback(): print ("Published Soil Moisture =
```

```
        %s %" %moisture,"Temperature =
```

```
%s C" % temperature, "Humidity = %s %" % Humid,'animal = %s'%animal,
```

```

"to IBM Watson")
    if
        animal>
        0
        .
        9
        8
        :
        p
        ri
        n
        t(
        "

        Alert") success    =
        deviceCli.publishEvent("IoTSensor",
"json",                                data,
qos=0, on_publish=myOnPublishCallback)
        if      not
        success:
        print("Not
        connected
        to    IoTF")
        time.sleep(1
        0)

```

```
deviceCli.commandCallback = myCommandCallback
```

```
# Disconnect the device and application from the cloud
```

```
deviceCli.disconnect()
```

GitHub Link

<https://github.com/IBM-EPBL/IBM-Project-48646-1660810853>

PROJECT DEMO VIDEO LINK

<https://drive.google.com/file/d/1XIGaeroZGxREgXgwwGk-Z0RZOxDHxpmH/view?usp=drivesdk>