**IOT BASED REAL TIME RIVER WATER QUALITY MONITORING**

**AND CONTROL SYSTEM**

**IBM PROJECT REPORT**

SUBMITTED BY

                  T.JOTHI                                                          951219106023

                  N.ALISH FLORA                                            951219106004

                  G.MUTHUSELVI                                             951219106042

                  S.ANANTHA SATHIYA                                  951219106006

**BACHELOR OF  ENGINEERING**

**IN**

**ELECTRONICS  AND  COMMUNICATION**

**ENGINEERING**

**INTRODUCTION**

**1.1 PROJECT OVERVIEW:**

pollution, global warming and so on are being formed,because of this is no safe drinking water for the world pollution. nowdays, maintaining pure supply of water  of the people is getting more challenging day by day.for driking purpose it should be 6.5-8.5ph. higher the turbidity higher the risk of diarrhea, cholera. lower the turbidity then the water is clean. Temperature sensor measures how the water is, hot or cold.

**1.2 PURPOSE:**

To measure various chemical and physical properties of water like pH.temperature and particle density of water using sensor.send the data collected in a raspherry pi,show the data in display and sent it to a cloud based database using wired/wireless channel, Trigger alarn when any discrepancies are found in the water quality. data visualization and analysis using cloud based visualization tools.

**LITERATURE SURVEY**

**2.1 EXISTING PROBLEM:**

To design a good quality model ,we reviewed out different exisiting system developed by researcher . different  authors have proposed distinguished model to check water quality by analyzing the parameter such as temperature ,pH  and conductivity. and son on . by considering  all these points , we designed a smart water quality monitoring system which  can perform all these monitoring function.

**2.2 REFERENCES:**

                [1]  Sridharan,S.(2014) water quality monitoring system using wireless sensornetwork.Internetwork journal of electronic communication engineering advanced research,3,399-402

                [2]  Losilla,F.,Garcia-sanchez.,F.,Garcia-Haro,J.and Haas,Z.J.[2011] A comprehensive Approach to WSN-Based ITS Application.sensor,10,10220-10265.http://dx.doi.org/10.3390/s111110220

                [3] Mo Deqing,Zhao [3:07 PM, 11/18/2022] Over Seen: 3] Mo Deqing, Zhao Ying, Chen Shangsong,  “Automatic Measurement and Reporting System of Water Quality Based on GSM,” 2012 International Conference on Intelligent System design and engineering application.

               [4] Nikhil Kedia, Water Quality Monitoring for Rural Areas- A Sensor Cloud Based Economical Project, in 1st International Conference on Next Generation Computing Technologies (NGCT-2015) Dehradun, India, 4-5 September 2015. 978-

1-4673-6809-4/15/$31.00 ©2015 IEEE

               [5] Jayti Bhatt, Jignesh Patoliya, Iot Based Water Quality Monitoring System, IRFIC, 21 feb, 2016.

               [6] Michal lom, ondrej priby & miroslav svitek, Internet 4.0 as a part of smart cities, 978-1-5090-1116-2/16/$31.00 ©2016 IEEE

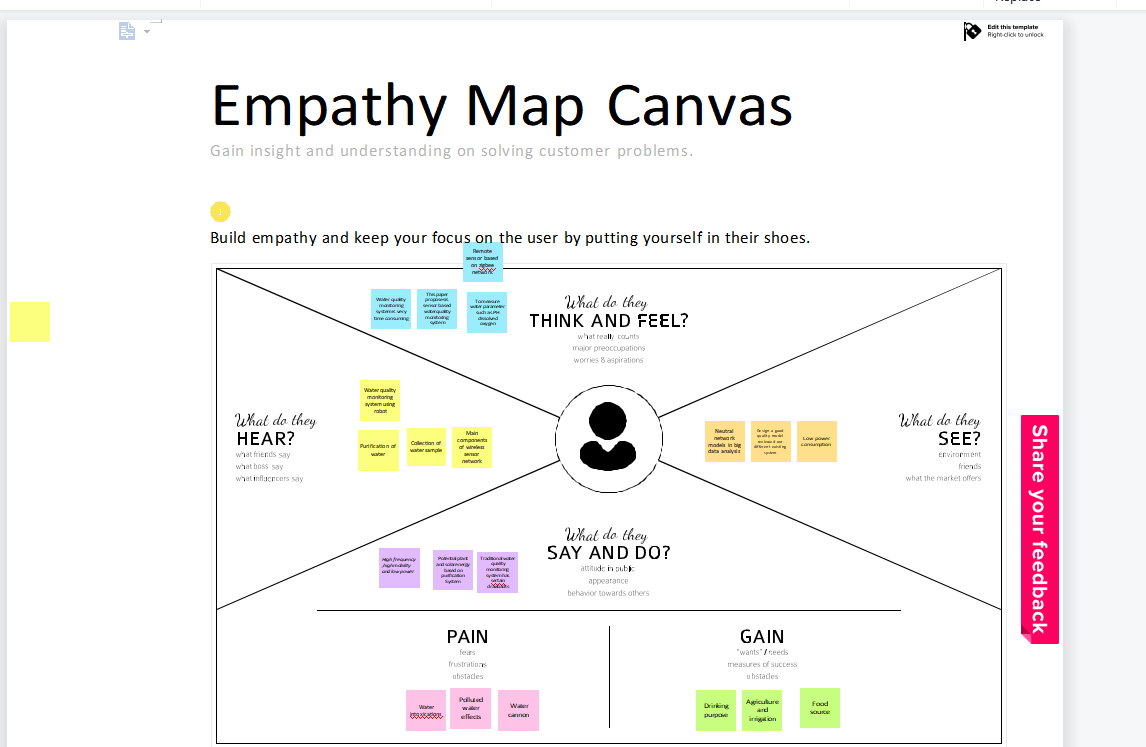
               [7] Zhanwei Sun, Chi Harold Liu, Chatschik Bisdikia\_, Joel W. Branch and Bo Yang, 2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and network.

**2.3 PROBLEM STATEMENT DEFINITION STATEMENT:**

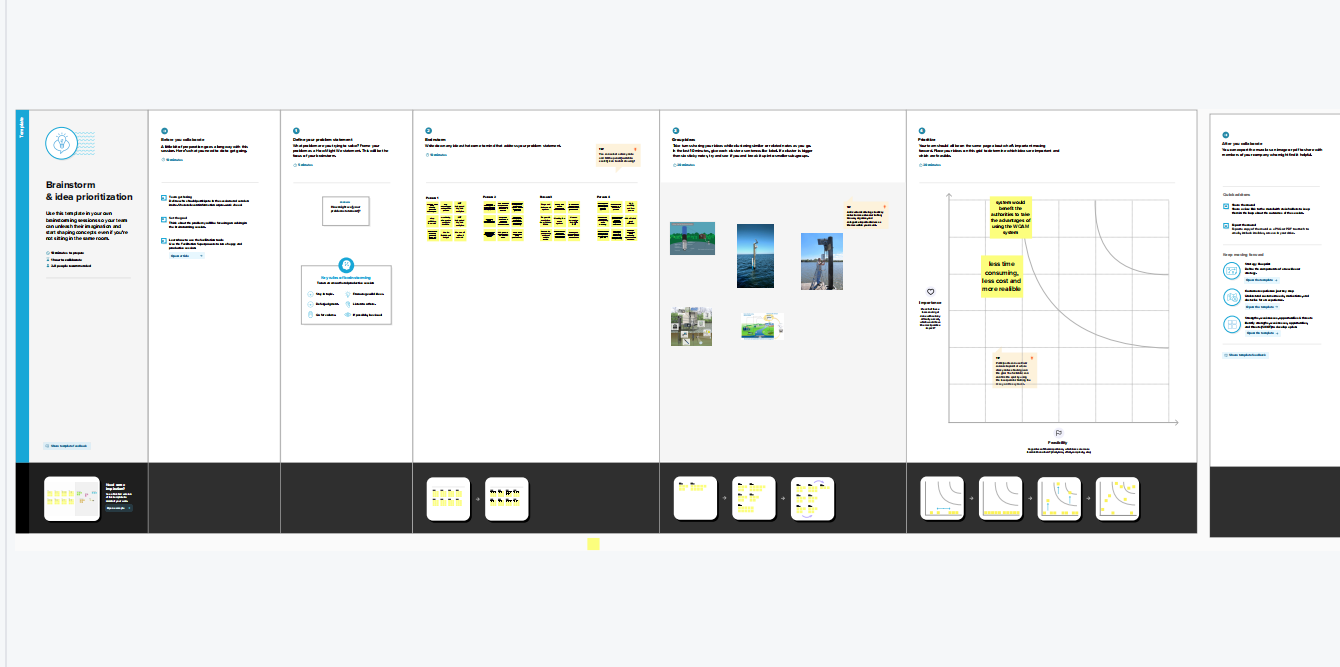
Due to the fast growing urbanization supply of safe drinking water is a challenge for the every city authority. Water can be polluted any time. So the water we reserved in the water tank at our roof top or basement in our society or apartment may not be safe. Still in India most of the people use simple water purifier that is not enough to get surety of pure water. Sometimes the water has dangerous particles or chemical mixed and general purpose water purifier cannot purify that. And it’s impossible to check the quality of water manually in every time.

**IDEATION AND PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS:**



**3.2 IDEATION AND BRAINSTROMING:**

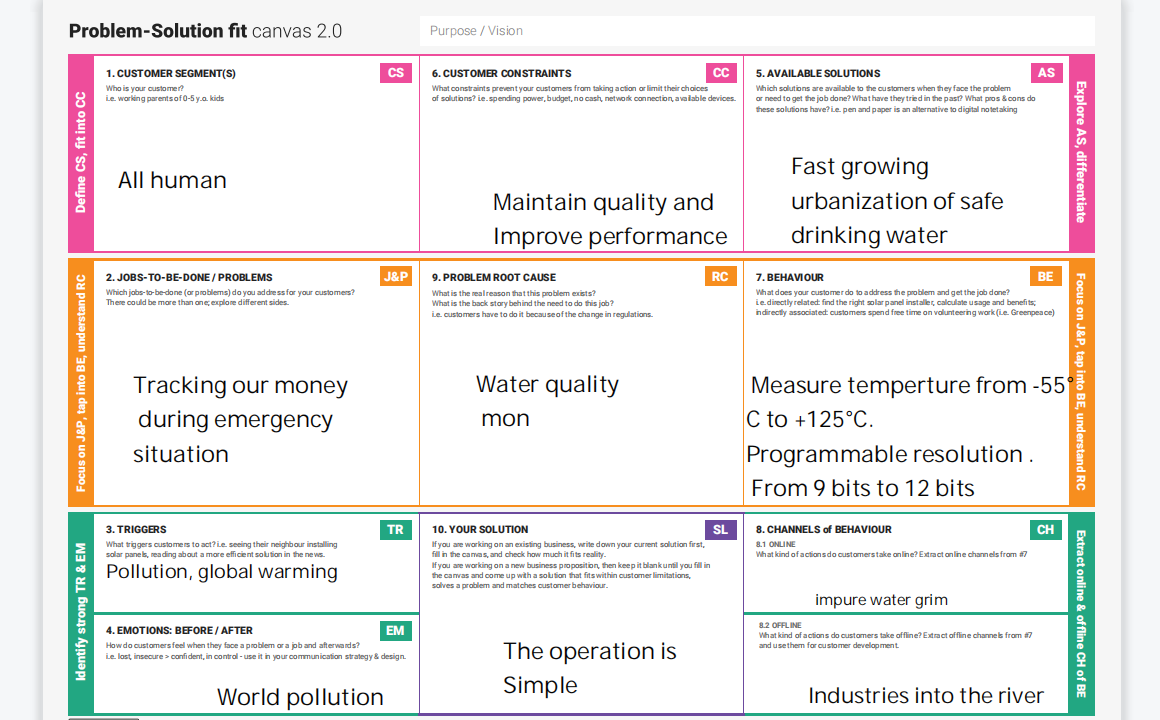


**3.3 PROPOSED SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **parameter** | **Description** |
| 1. | Ammonia, Ammonium | Ammonia in water is either un-ionized ammonium or the ammonium ion. Typically the value reported is the sum of both forms and is reported as total ammonia or simply ammonia |
| 2. | Biochemical Oxygen Demand-BOD | Biological oxygen demand directly affects the amount of dissolved oxygen in rivers and streams and the type of organic and inorganic material in the water |
| 3. | Blue Green algae -BGA | BGA can reduce nitrogen and carbon in water, but can also deplete dissolved oxygen when overabundant. monitoring BGA is important because that pose a serious threat to water quality |
| 4. | CDOM-FDOM | Chromo dissolved organic matter or colored dissolved organic matter (CDOM): both refer to organic matter in water that absorb in UV spectrum |
| 5. | Chlorophyll in water | Chlorophyll in various forms in bound within living cells or photosynthesis organism such as phytoplankton and bacteria |
| 6. | Colorimetry & photometry | Colorimetry allow you to easily take reading directly in the multiple parameters power pack reagents are also 37% larger than other brands and the easy to open pouches have a pre-tear |
|  |  |  |

**3.4 PROBLEM SOLUTION FIT:**

 **3.4 PROBLEM SOLUTION FIT:**

****

**REQUIREMENT ANALYSIS**

**4.1 FUNCTIONAL REQUIREMENT:**

**Project Design Phase-II**

**Solution Requirements (Functional & Non-functional)**

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through email id and mobile number |
| FR-2 | User Confirmation | Confirmation via Email  Confirmation via OTP |
| FR-3 | Web application | Node –red-service |
| FR-4 | Configuration to device | IBM Watson IOT platform |
| FR-5 | Database | Cloudant database |
| FR-6 | python | IBM IOT  platform |

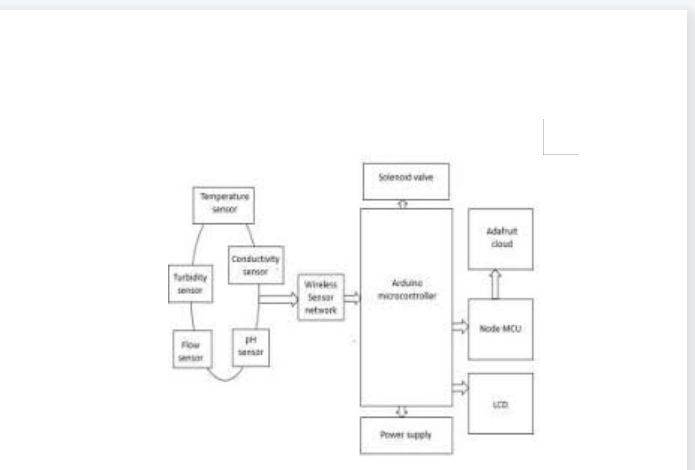
**Non-functional Requirements:**

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

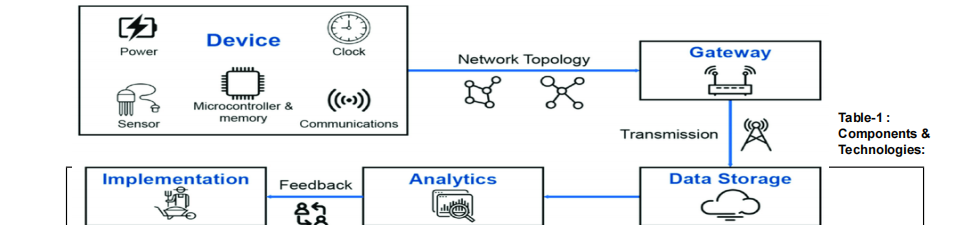
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | Sensor and changing the relevant software program |
| NFR-2 | **Security** | Due to the fast growing urbanization supply of safe drinking water |
| NFR-3 | **Reliability** | The estimation of water parameter like turbidity, PH, dissolved oxygen. |
| NFR-4 | **Performance** | Water quality monitoring system using robot |
| NFR-5 | **Availability** | There is need of developing better methodologies to motor |
| NFR-6 | **Scalability** | The impure water from the industry can be purified |

**PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAM:**

****

**5.2 SOLUTION AND TECHANICAL ARCHITECTURE:**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1. | User Interface | User interface through the mobile application. | HTML, CSS, JavaScript / Angular Js / React Js etc. |
| 2. | Registration | User register in the application to connect with  the sensor. | Java / Python |
| 3. | Registration | Verification of user sensor detail. | Ipython flask, HTml, css. |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | Notification of registration | Send notification to the user | Python flask, Html css. |
| 5. | Monitoring devices | User start monitoring by the application. | Python flask |
| 6. | Cloud Database | Database Service on Cloud | IBM DB2, IBM Cloudant etc. |
| 7. | File Storage | Sensor collect all the receiving and transmitting  information are stored | IbM DB2. |
| 8. | External API-1 | Shows tha data. | IBM Weather API, etc. |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | Iot is a efficient open source application. | CoAP. |
| 2. | Security Implementations | secure password practices | Encryption. |
| 3. | Scalable Architecture | it connected with scalable architecture. | python flask. |
| 4. | Availability | This is available at anytime anywhere. | IBM DB2. |
| 5. | Performance | Monitor the area and send the information. | IBM DB2. |

**5.3 USER STOREIS**

Use the  below template to list all the user stories for the product.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Funtional**  **Requirment(epic)** | **User Stories Number** | **User story**  **/task** | **Acceptance creteria** | **priority** | **Release** |
| Hardware Design | Control Surface | USN-1 | AS a user ,utilized core person | This system several  sensor very time consuming | Reduce the air polution | System with a monotous process. |
| PH sensor | Thing useful constant to diplay effect on the author. | USN-2 | Hydrongen ion density in a bleach. | Ph value process the solubility of element and componends | pH=-log{H=} | It consist of value 0-14pH. |
| Parameter monitring | Us environmental  production agency | USN-3 | AS a user ,water is a one of the most important facter | Monitoring and detecting change in the water monitoring  including turbidity | Feasible to infer the water quality | Amount of the impurityin the water. |
| Application | Domestic water is indeted  drinking and cooking purpose | USN-4 | As a user,acceptable limits of substance  such as aluminium. | Efficience to availability real time information | Data collect at the apart site can be displayed in visual formar. | System will immense help  high frequency . |
| Software Design | WSN used for a sysytem | USN-5 | IOT platfrom | Natural network models in big data | Big data analytic |  |

**PROJECT PLANNING AND SCHEDULING**

**6.1 SPRINT PLANNING AND ESTIMATION:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Login | USN-1,USN-2 | The uniqueness of our proposed paper is to obtain the water  monitoring system with high frequency, high mobility. | 20 | High | JOTHI |
| Sprint-2 | Dashboard | USN-3 | Real-time data access can be done by using remote monitoring and Internet of Things (IoT) technology. | 20 | Low | ALISH FLORA |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-3 | Dashboard | USN-4 | Current water quality monitoring system is a manual system with a monotonous process and is very time-consuming | 20 | medium | MUTHU SELVI |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-4 | Dashboard | USN-5 | Deep learning neural network  models, Belief Rule Based (BRB) system and is also compared with standard values | 20 | High | ANANTHA SATHYA |

**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=SPRINT DURATION           20***

***---------------------- =    -------=2***

**VELOCITY**                       10

**CODING AND SOLUTIONING**

**7.1 FEATURE -1**

CODE:-

#include "DHTesp.h" #include <cstdlib> #include <time.h> #include <WiFi.h>

#include <PubSubClient.h>

#define ORG "pfrrli"

#define DEVICE\_TYPE "Rasp" #define DEVICE\_ID "12345"

#define TOKEN "12345678"

#define speed 0.034

char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; char publishTopic[] = "iot-2/evt/data/fmt/json";

char authMethod[] = "use-token-auth"; char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

PubSubClient client(server, 1883, wifiClient); float temperature = 0;

int pH = 0;

String quality\_status = ""; String temperture\_status = "";

void setup() {

**Serial**.begin(99900);

wifiConnect(); mqttConnect();

}

void loop() { srand(time(0));

//initial variable int p;

temperature = random(-20,40); pH = random(0,14);

if(pH > 6.5 && pH < 8.5){ p = 0;

}

else{

p = 1;

}

//set a quality status switch (p) {

case 0:

quality\_status = "Drinkable"; break;

case 1:

quality\_status = "Not Drinkable"; break;

}

//Obivously the output.It is like json format 'cause it will help us for future sprints String payload = "{";

payload+="\"pH level is \":"; payload+=pH; payload+=",";

payload+="\"Temperature of Water\":"; payload+=(int)temperature; payload+=",";

payload+="\"Alert\":\""+quality\_status+"\"}";

**Serial**.println(payload);

if(client.publish(publishTopic, (char\*) payload.c\_str()))

{

**Serial**.println("Publish OK");

}

else{

**Serial**.println("Publish failed");

}

delay(1000);

if (!client.loop())

{

mqttConnect();

}

}

void wifiConnect()

{

**Serial**.print("Connecting to "); **Serial**.print("Wifi"); WiFi.begin("Wokwi-GUEST", "", 6);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

**Serial**.print(".");

}

**Serial**.print("WiFi connected, IP address: ");

**Serial**.println(WiFi.localIP());

}

void mqttConnect()

{

if (!client.connected())

{

**Serial**.print("Reconnecting MQTT client to ");

**Serial**.println(server);

while (!client.connect(clientId, authMethod, token))

{

**Serial**.print("."); delay(500);

}

**Serial**.println();

}

}

**DIAGRAM.JSON:-**

{

"version": 1,

"author": "PNT2022TMID51903",

"editor": "wokwi", "parts": [

{ "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -16.32, "left": -0.82, "attrs": {} },

{

"type": "wokwi-dht22",

"id": "dht1",

"top": -30.22,

"left": 165.89,

"attrs": { "temperature": "59.3" }

}

],

"connections": [

[ "esp:TX0", "$serialMonitor:RX", "", [] ],

[ "esp:RX0", "$serialMonitor:TX", "", [] ],

[ "dht1:SDA", "esp:D15", "green", [ "v0" ] ],

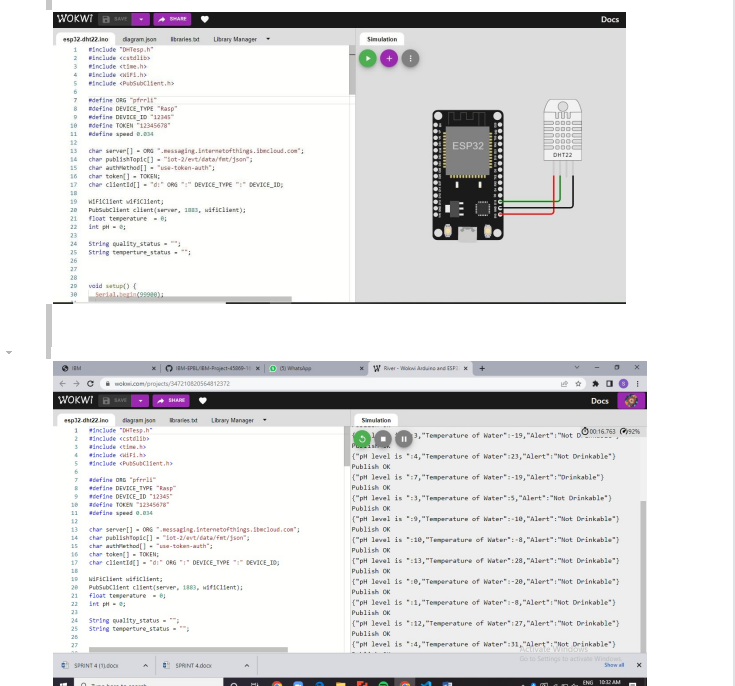
[ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],

[ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ]

]

}

OUTPUT:-

****

**TESTING**

**8.1 TEST CASES:**

**User Acceptance Testing**

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open

issues of the web page for real time river water quality monitoring and control

system project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level and how ther were  resolved.

 Test Case Analysis

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SECTION** | **TOTAL CASES** | **NOT TESTED** | **FAIL** | **PASS** |
| client application | 40 | 0 | 2 | 36 |
| security | 2 | 0 | 0 | 2 |
| application login | 3 | 0 | 0 | 3 |
| exception reporting | 9 | 0 | 0 | 9 |
| final report output | 3 | 0 | 0 | 3 |

**TESTING TOOLS:**

User Acceptance Testing

User acceptance testing (UAT) is necessary when implementing changes

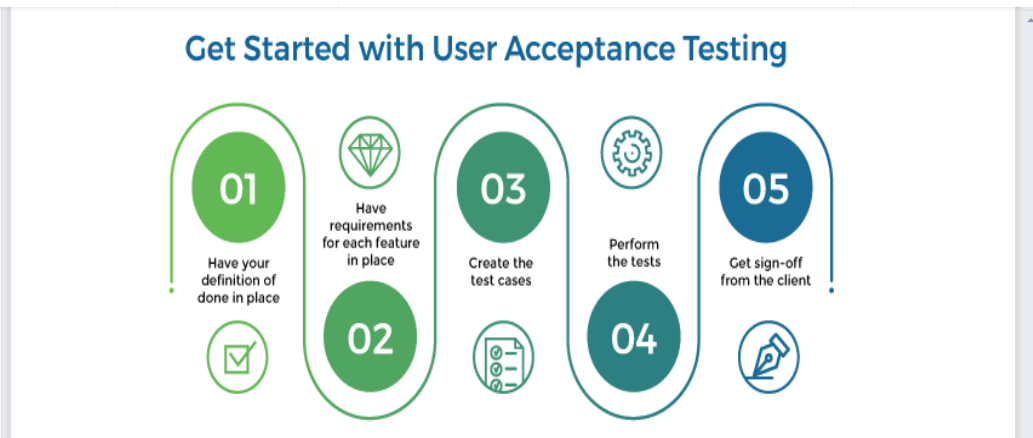
to an IT landscape due to the ever increasing complexity of software which can cause bugs to slip

through even under the most perfect development conditions. Commonly, acceptance testing is

performed as the last step before the release of a software, after all other testing phases have exited.

As implied by its name, user acceptance testing is typically performed by the end users in a real setting

during the unit-, integration- and system testing phases.

****

**TEST CASES:**

In the context of this thesis it is important to note the use of subtle differences in the

definition of automated testing. In literature we often find mentions of how automated test cases can

require a lot of manual effort coding cases.

****

**Automated Acceptance Testing**

 Automated acceptance testing (AAT) is the practice of executing

business logic test cases automatically and can serve as an addition to user acceptance testing . Some

sources also discuss the automation of user acceptance testing, in which case the automated acceptance

tests are UAT tests as described above. It mostly revolves around the idea that businesses which

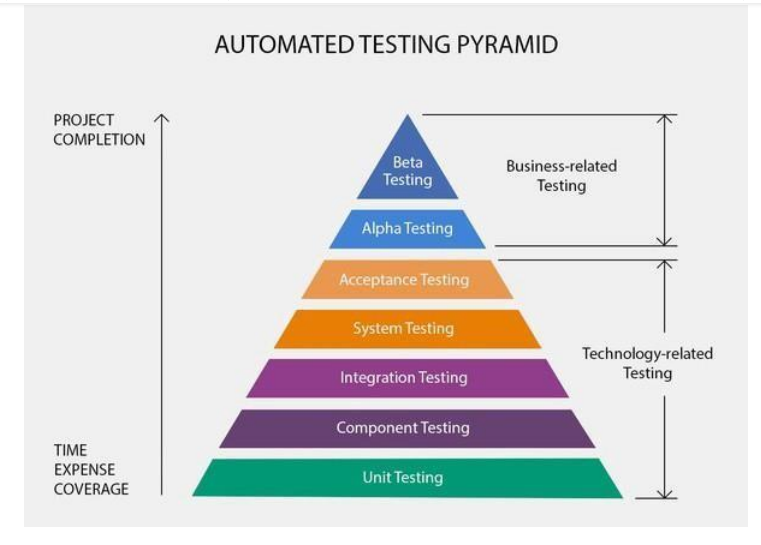
procure software define a set of rigorous requirements with clearly defined input and output data.

When this is the case, test cases can be automated using a variety of tools making use of UI selectors

or visual user interface automation . This means that scripts could be written to automate user

acceptance test cases with the intent to reduce the manual effort emerging from the repeated execution

of acceptance testing

****

**RESULTS**

**9.1 PERFORMANCE METRICS:**

COLLECTION OF PERFORMANCE MEASUREMENTS

Managing application performance requires the continuous collection of data

about all relevant parts of the system starting from the end user all the way through the system. This

collected data is the basis for getting a holistic end-to-end and up-to-date view of the application state

including the end-user experience. In this chapter, we will discuss what data to collect, and from

where and how to collect the data in order to achieve this view Most application systems are

implemented in a way that, in addition to the application logic executed at the provider’s site (referred

to as the back-end), parts of the application are executed at client’s site. The client site usually

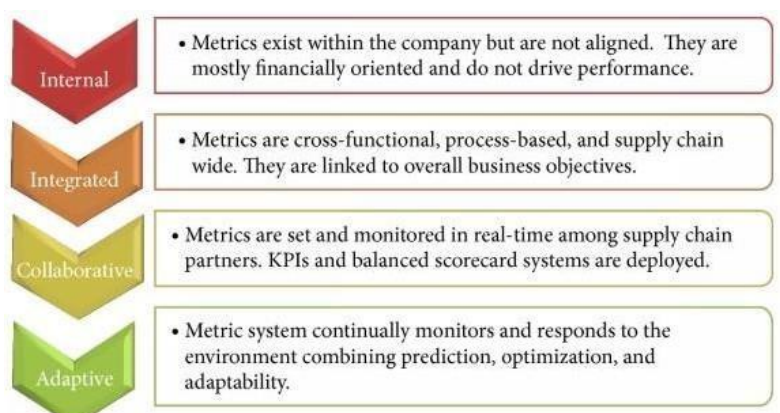
constitutes a system tier accessing the back-end

EXTRACTION OF PERFORMANCE-RELEVANT SYSTEM INFORMATION

summary statistics (e.g., counts, percentile, etc.) over time, execution traces provide a

detailed representation of the application-internal control flow that results from individual system

requests.

****

          EXTRACTION OF PERFORMANCE-RELEVANT SYSTEM INFORMATION

collection of performance measurements from the relevant locations of the

application system. This chapter focuses on the representation of higher the application system.

While time series represent summary statistics (e.g., counts, percentile, etc.) over time, execution

traces provide a detailed representation of the application-internal control flow that results from

individual system requests.

From this data, architectural information, including logical and physical deployments

and interactions (topology), can be extracted. For all cases, we will highlight examples and use cases

in the context of APM level performance-relevant information about the system and their end-users

that can be extracted from this data and that is used for APM visualization and reasoning, as detailed

in the next chapters. Notably, we will focus on three commonly used representations, namely time

series, execution traces, and augmented information about the architecture.

When depicting the number of users accessing a system, time series usually show a

periodic pattern, e.g., based on the weekdays and the hours of the day. Other interesting patterns are

spikes, for instance, indicating peaks in workload or hiccups.

EXECUTION TRACES

A data structure commonly used in APM for this purpose is an execution trace.

Informally, an execution trace is a representation of the execution flow of a request through the

system–ideally starting from the end user..

****

The execution trace starts with an operation called do Filter

that is commonly found as an entry point in web-based applications. It can be observed that the

execution of the do Filter operation includes a sequence of additional nested operation executions,

until the list operation performs a sequence of calls to a database.

In addition to the execution flow, capturing components (e.g.,

Java classes or microservices) and operations, and locations (e.g., application server, IP address),

execution traces usually include further measurements. One type of performance measurement

commonly found in execution traces is the response time (or duration) of each operation execution.

In the example, the response time for each operation execution is

included in the second column. Moreover, execution traces may include information such as the

parameters of the operation executions.

ARCHITECTURAL INFORMATION

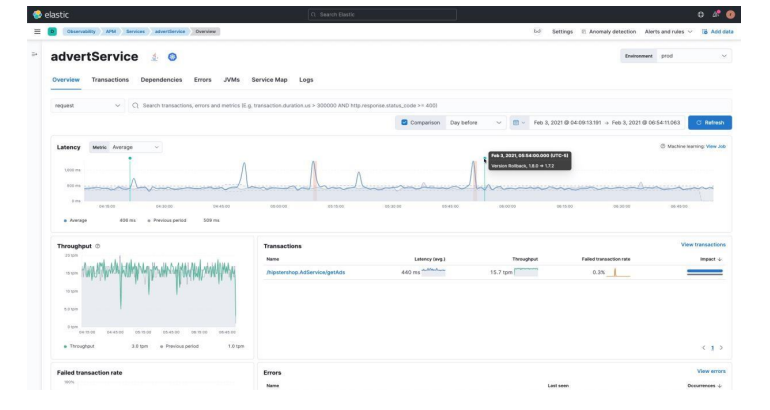
Time series and execution traces allow to analyze the chronological order of

performance measurements and of individual requests respectively. This information is commonly

used to derive and represent performance-relevant architectural information of a system. The

architecture of a system includes structural and dynamic information. Examples for structural

information are the existence and deployment of software and hardware components.

****

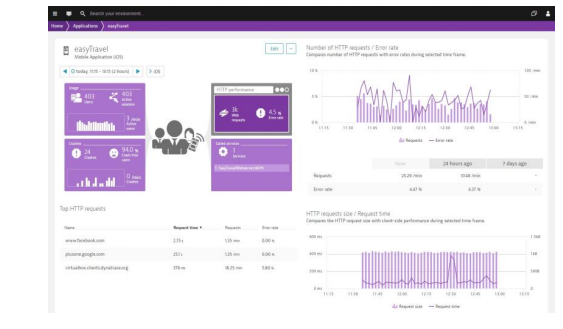
The dynamic information includes interactions (e.g., number of calls, average

response times) between components and associated information about the runtime behavior, e.g., a

health state or time series. In Chapter 4 we include example of performance-augmented architectural

information. This representation is useful to have an overall state of the system and it provides a basis

for a detailed manual or automated.

****

**10. ADVANTAGES & DISADVANTAGES**

**Benefits or Advantages of IoT based Water Quality Monitoring**

**System**

**Following are the benefits or advantages of IoT based Water Quality Monitoring System are**

**as follows.**

**➤The boat is mobile in nature and hence large number of samples are easily collected from**

**different locations in less time.**

**➤It is very easy to maintain the IoT based water quality monitoring system as all the**

**electronic boards are available in the boat itself.**

**➤The system is very cheap as the hardware and software does not cost much.**

**➤Machine learning techniques have made it very easy to plot the data collected in various**

**formats for proper analysis.**

**➤Cloud storage platforms such as IBM CLOUD,azure helps in storing the sensor data**

**immediately and wirelessly to the robust servers.**39

**Disadvantages of IoT based Water Quality Monitoring System**

**Following are the disadvantages of IoT based Water Quality Monitoring System are as**

**follows.**

**➤**the system cannot provide real time monitoring of water parameters

**➤**For trouble shooting the system technicians is required and this process might take some

time

**11. CONCLUSION**

Water pollution is one of the biggest threats to all living beings. Polluted water causes

various diseases in humans, plants, animals, which, in turn, negatively impact the life cycle

of the ecosystem. If the contamination is detected early on, suitable measures can be

taken to preserve water quality or even upgrade it.

Therefore, Smart Water Quality Monitoring using IoT is paramount to supply pure water

in real-time. Thanks to innovation in sensors, wireless modules, and communication

devices, the activity is easy.

**12.FUTURE SCOPE**

We use water detection sensor has unique advantage. It consumes less time to

monitor than a manual method for checking polluted levels, and notifies

immediately to reduce affected rate of pollution in water. People who are living in

rural areas near to the river will bevery satisfied with our idea. It will be useful to monitor water pollution in specific area.

So this system prevent people from water pollution. It will be used for farming

purpose to check quality water temperature and PH level. Our Impact of this

project is also create a social satisfaction for farmers too. The scalabilty of this

project gives the addition of more different type of sensors. By interfacing the

relay we can control the supply of water. We can also implement as a revenue

model. Thissystem could also be implemented in variousindustrial processes. The

system can be modified according to the needs of the user and can be implemented

along with lab view to monitor data on computers

**13. APPENDIX**

**13.1 SOURCE CODE:**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide your IBM Watson Device

Credentials

organization = "uo60re"

deviceType = "AKASH"

deviceId = "1234"

authMethod = "token"

authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" %

cmd.data['command'])

status=cmd.data['command']

if status=="lighton":

print ("led is on")

else:

print ("led is off")

#print(cmd)deviceType = "AKASH"

deviceId = "1234"

authMethod = "token"

authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" %

cmd.data['command'])

status=cmd.data['command']

if status=="lighton":

print ("led is on")

else:

print ("led is off")

#print(cmd)

try:

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:

#Get Sensor Data from DHT11

temp=random.randint(60,100)

Turbidity=random.randint(0,100)

phvalue=random.randint(2,14)

data = { 'temp' : temp,

'Turbidity': Turbidity,'phvalue': phvalue}

#print data

def myOnPublishCallback():

print ("Published temp = %s

'C" % temp, "Turbidity = %s %%" %

Turbidity,"phvalue = %s %%" % phvalue,

"to IBM Watson")

success =

deviceCli.publishEvent("IoTSensor",

"json", data, qos=0,

on\_publish=myOnPublishCallback)

if not success:

print("Not connected to

IoTF")

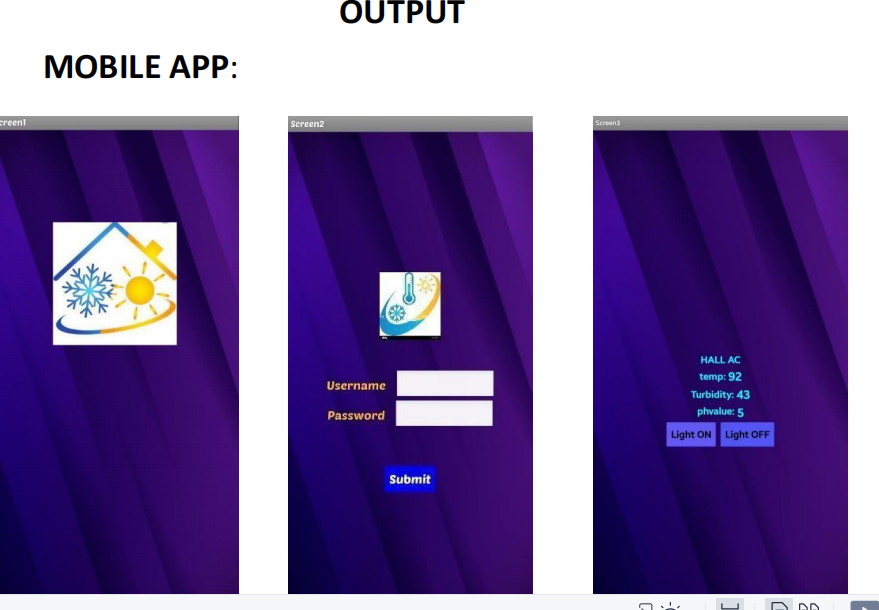
time.sleep(10)

deviceCli.commandCallback =

myCommandCallback# Disconnect the device and

application from the cloud

deviceCli.disconnecT()

****

**PROJECT DEMO LINLK:**

<https://drive.google.com/file/d/1RT0ifRhLdqgYj3hmwrtlW_Y2mhwDZCdY/view?usp=share_link>

}