# PROJECT REPORT

**PROJECT NAME: EFFICIENT WATER QUALITY ANALYSIS AND PREDICTION USING MACHINE LEARNING**

**TEAM ID:PNT2022TMID11573**

**TEAM LEADER : T.N.HARIVARSHAN**
**TEAM MEMBERS : S.ARUNKUMAR**
**G.ARUNPANDI**
**S.BOGAR**

# TABLE OF CONTENTS


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 REQUIREMENTS
   - 4.2 NON-FUNCTIONAL REQUIREMENTS
5. **PROJECT DESIGN**
   - 5.1 DATA FLOW DIAGRAM
   - 5.2 SOLUTION & TECHNICAL ARCHITECTURE
   - 5.3 USER STORIES
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 SPRINT PLANNING AND ESTIMATION
   - 6.2 SPRINT DELIVERY SCHEDULE
   - 6.3 REPORTS FROM JIRA
7. **CODING & SOLUTIONING**
   - 7.1 FEATURE 1

7.2 FEATURE 2

**8. TESTING**

8.1 TEST CASES

8.2 USER ACCEPTANCE TESTING

**9. RESULT**

9.1 PERFORMANCE METRICS

**10. ADVANTAGES & DISADVANTAGES**

**11. CONCLUSION**

**12. FUTURE SCOPE**

**13. APPENDIX**

SOURCE CODE

GITHUB & PROJECT DEMO LINK

# CHAPTER 1
# INTRODUCTION

## 1.1 Project Overview:

Water quality analysis is a complex topic due to the different factors that influence it. This concept is inextricably linked to the various purposes for which water is used. Different needs necessitate different standards. There is a lot of study being done on water quality prediction. Water quality is normally determined by a set of physical and chemical parameters that are closely related to the water's intended usage. The acceptable and unacceptable values for each variable must then be established. Water that meets the predetermined parameters for a specific application is considered appropriate for that application. If the water does not fulfill these requirements, it must be treated before it may be used. Water quality can be assessed using a variety of physical and chemical properties. As a result, studying the behavior of each individual variable independently is not possible in practice to accurately describe water quality on a spatial or temporal basis.

The ecosystem and human health are directly impacted by the water quality. Water is used for many different things, including drinking, farming, and industrial uses. A crucial indicator of effective water management is the water quality index (WQI). The aim of this work was to classify a dataset of water quality in various locations across India using machine learning techniques including RF, NN, MLR, SVM, and BTM. Features including dissolved oxygen (DO), total coliform (TC), biological oxygen demand (BOD), nitrate, pH, and electric conductivity determine the quality of water (EC). These characteristics are handled in five steps: feature correlation, applied machine learning classification, model's feature significance, and data pre-processing utilizing min-max normalization and missing data management using RF.

## 1.2 Purpose:

The quality of water is a major concern for people living in urban areas. The quality of water serves as a powerful environmental determinant and a foundation for the prevention and control of waterborne diseases. However, predicting the urban water quality is a challenging task since the water quality varies in urban spaces non-linearly and depends on multiple factors, such as meteorology, water usage patterns, and land uses. The purpose of this project is to Predict Water Quality by considering all water quality standard indicators.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 EXISTING PROBLEM:

For testing the water quality we have to conduct lab tests on the water which is costly and time-consuming as well. So, in this paper, we propose an alternative approach using artificial intelligence to predict water quality. This method uses a significant and easily available water quality index which is set by the WHO(World Health Organisation). The data taken in this paper is taken from the PCPB India which includes 3277 examples of the distinct wellspring. In this paper, WQI(Water Quality Index) is calculated using AI techniques. So in future work, we can integrate this with an IoT based framework to study large datasets and to expand our study to a larger scale. By using that it can predict the water quality fast and more accurately than any other IoT framework. That IoT framework system uses some limits for the sensor to check the parameters like ph, Temperature, Turbidity, and so on. And further after reading this parameter pass these readings to the Arduino microcontroller and ZigBee handset for further prediction.

## 2.2 REFERENCES:

**Machine learning algorithms for efficient water quality prediction: A Review of literature Jamal Mabrouki, Ghizlane Fattah, Azedine Guezzaz, Faissal Aziz, 2021**

Water is an essential resource for human existence. In fact, more than 60% of the human body is made up of water. Our bodies consume water in every cell, in the different organisms and in the tissues. Therefore, to design a model that predicts water quality is nowadays very important to control water pollution, as well as to alert users in case of poor quality detection. The method we propose is based on four water parameters: temperature, pH, turbidity and coliforms. The use of the multiple regression algorithms has proven to be important and effective in predicting the water quality index. Water quality predicting is a key and primary task in the context of the environmental control strategy. Certainly, the accuracy of predictions will surely contribute significantly to more appropriate conservation of water resources. In this research paper, our goal is to suggest a model for predicting water quality based on machine learning algorithms and with minimal parameters. Machine learning is an analytical approach of data expected to make the analysis model more automatic.

**Efficient water quality prediction models based on machine learning: A Review of literature Nainital Lake, Uttarakhand**

Water quality deterioration increases day by day in hilly areas due to increasing tourism activity, unplanned construction, disposal of solid waste, improper sewage management. With this idea, the work investigates different machine learning algorithms to evaluate the water quality index (WQI) and the water

quality class (WQC). This paper utilizes Nainital Lake as a study area. The models used for testing and training comprise algorithms of machine learning for both binary and multiclass classification. In this paper, eight machine learning algorithms were employed for regression analysis, and nine machine learning algorithms were used for classification analysis. The result demonstrates that in regression analysis, the Random Forest algorithm comes out to be the most efficient Machine Learning algorithm. However, in the case of classification analysis, no single algorithm is good enough for prediction, three algorithms Stochastic Gradient Descent, Random Forest, and Support Vector Machine with the same accuracy proved to be efficient to predict water quality.

**Evaluation of E. coli in sediment for assessing irrigation water quality using machine learning**

Fresh produce irrigated with contaminated water poses a substantial risk to human health. This study evaluated the impact of incorporating sediment information on improving the performance of machine learning models to quantify E. coli level in irrigation water. Field samples were collected from irrigation canals in the Southwest U.S., for which meteorological, chemical, and physical water quality variables as well as three additional flow and sediment properties: the concentration of E. coli in sediment, sediment median size, and bed shear stress. Water quality was classified based on E. coli concentration exceeding two standard levels: 1 E. coli and 126 E. coli colony forming units (CFU) per 100 ml of irrigation water. Two series of features, including (FIS) and excluding (FES) sediment features, were selected using multi-variant filter feature selection. The correlation analysis revealed the inclusion of sediment features improves the correlation with the target standards for E. coli compared to the models excluding these features. Support vector machines, logistic regression, and ridge classifiers were tested in this study. The support vector machine model performed the best

for both targeted standards. Besides, incorporating sediment features improved all models' performance.

**machine learning in water quality evaluation : A Review of literature MengyuanZhu,JiaweiWang,XiaoYang,2022.**

With the rapid increase in the volume of data on the aquatic environment, machine learning has become an important tool for data analysis, classification, and prediction. Unlike traditional models used in water-related research, data-driven models based on machine learning can efficiently solve more complex nonlinear problems. In water environment research, models and conclusions derived from machine learning have been applied to the construction, monitoring, simulation, evaluation, and optimization of various water treatment and management systems. Additionally, machine learning can provide solutions for water pollution control, water quality improvement, and watershed ecosystem security management. In this review, we describe the cases in which machine learning algorithms have been applied to evaluate the water quality in different water environments, such as surface water, groundwater, drinking water, sewage, and seawater. Furthermore, we propose possible future applications of machine learning approaches to water environments.

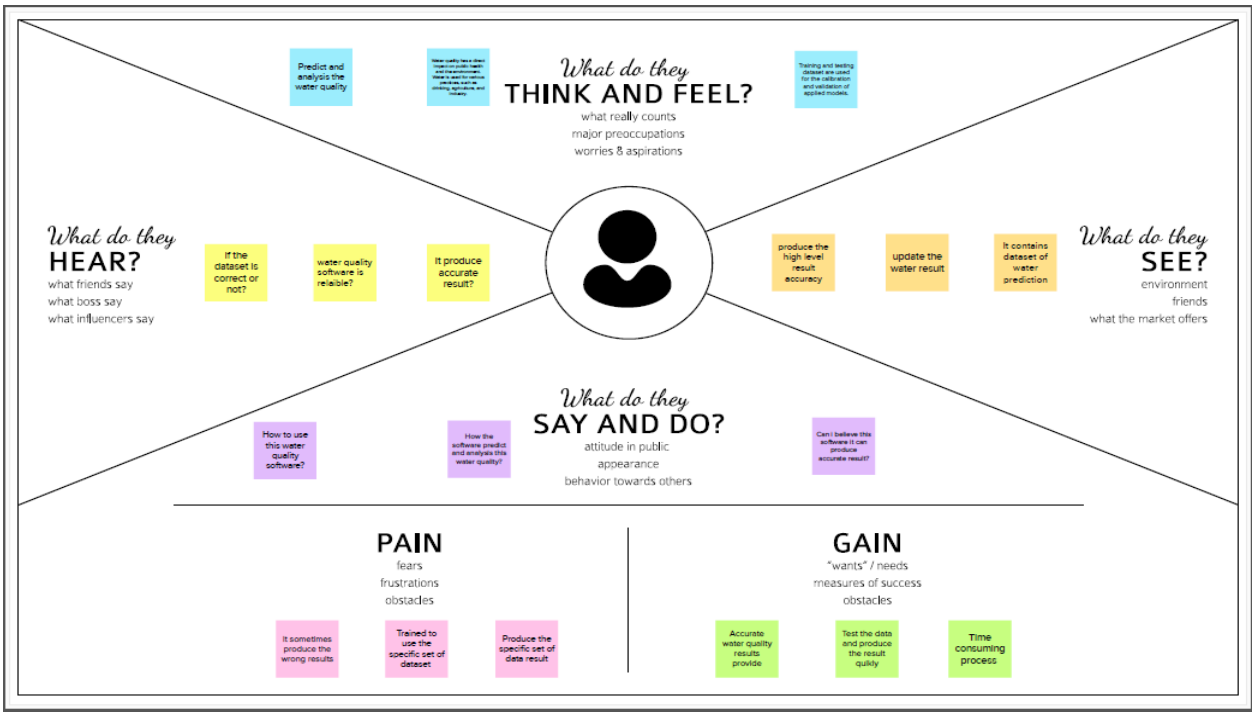
**2.3 Problem statement definition:**

Water is considered as a vital resource that affects various aspects of human health and lives. The quality of water is a major concern for people living in urban areas. The quality of water serves as a powerful environmental determinant and a foundation for the prevention and control of waterborne diseases. However predicting the urban water quality is a challenging task since the water quality varies in urban spaces non-linearly and depends on multiple factors, such as meteorology, water usage patterns, and land uses, so this project aims at building a Machine Learning (ML) model.

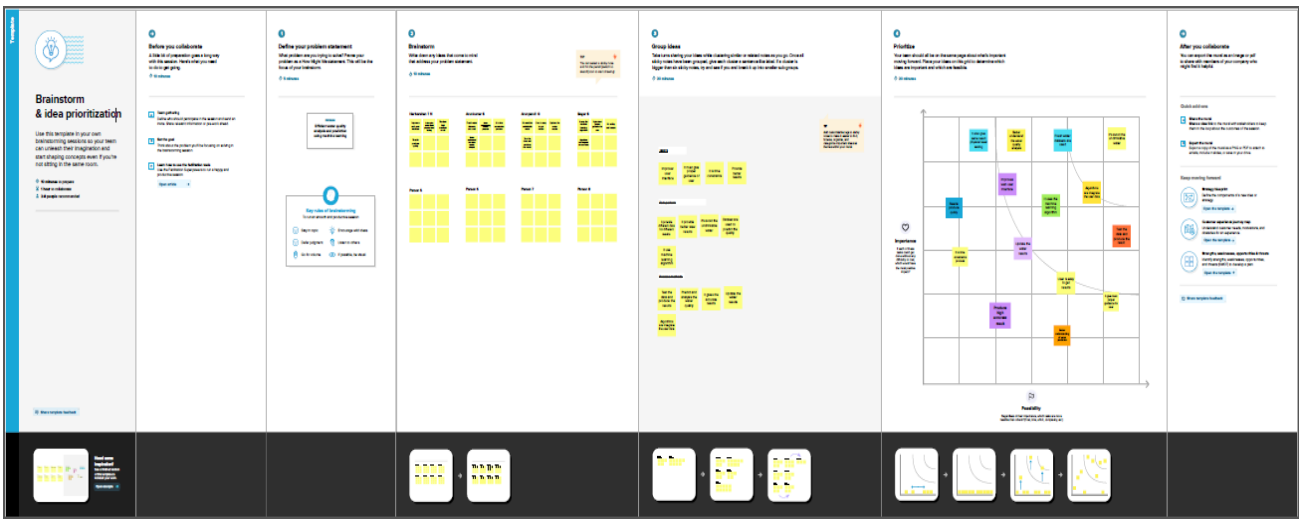
# CHAPTER 3

# IDEATION PHASE

## 3.1 EMPATHY MAP CANVAS:



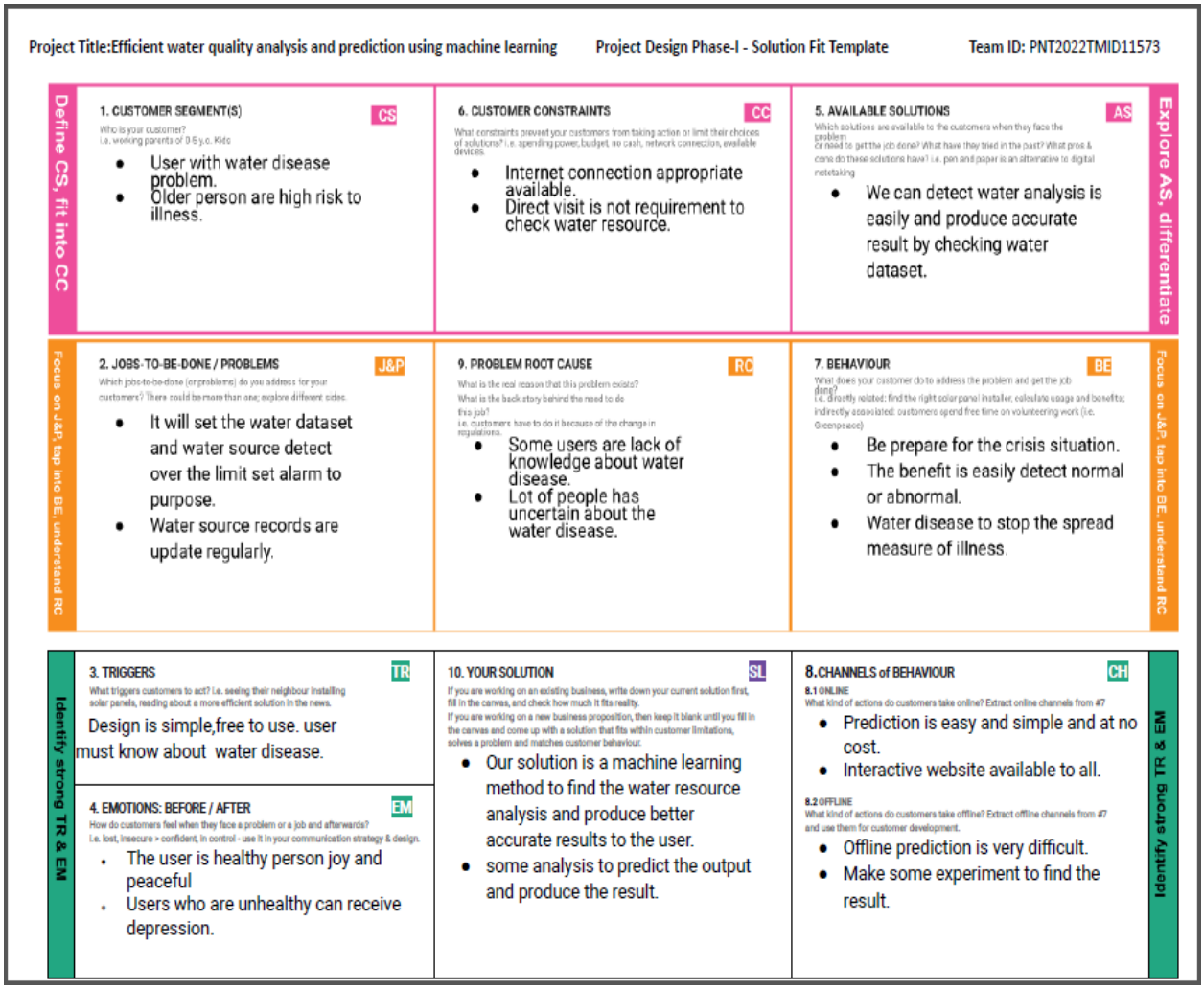

## 3.2 IDEATION & BRAINSTORMING:



## 3.3 PROPOSED SOLUTION:

| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | The software has been developed to predict and analysis the water quality and produce better result. |
| 2. | Idea / Solution description | Collect the water quality analysis dataset and create user interacts with the UI (User Interface) to enter Data. The entered data is analyzed by the model which is integrated. Once model analyses the input the prediction is showcased on the UI. |
| 3. | Novelty / Uniqueness | The water has highly polluted then show the alertness to the user. |
| 4. | Social Impact / Customer Satisfaction | To produce better water quality prediction Can drive to the vision of healthy nation. It can be identify the accurate level of water. |
| 5. | Business Model (Revenue Model) | It been sell our service/product to water purifier companies. Can collaborate with governments in analysing and providing the water quality solutions. |
| 6. | Scalability of the Solution | It provides the better accuracy of water quality. Easy to identify the ph level, temperature. |

## 3.4 PROBLEM SOLUTION FIT:




# CHAPTER 4

# REQUIREMENT ANALYSIS

## 4.1 FUNCTIONAL REQUIREMENTS:

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User Registration | Registration through Form<br>Registration through Gmail<br>Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email<br>Confirmation via OTP |
| FR-3 | User input | Users are required to give chemical components of their water which they need tested.<br>The chemical components such as Temperature,pH,DissolvedOxygen,Fecal Coliform,Biochemical oxygen demand,conductivity and Nitratenan details are must. |
| FR-4 | Output Display | Based on the range of water quality index available,given water samples are analyzed and predicted the final results. |
| FR-5 | Model prediction | Confirming based on water quality index and shows the ML prediction with percent of various parameter. |
| FR-6 | Data handling | File contains water quality metrics for different water bodies. |
| FR-7 | Quality analysis | Analyze with acquired information of water across various water quality indication using different models. |

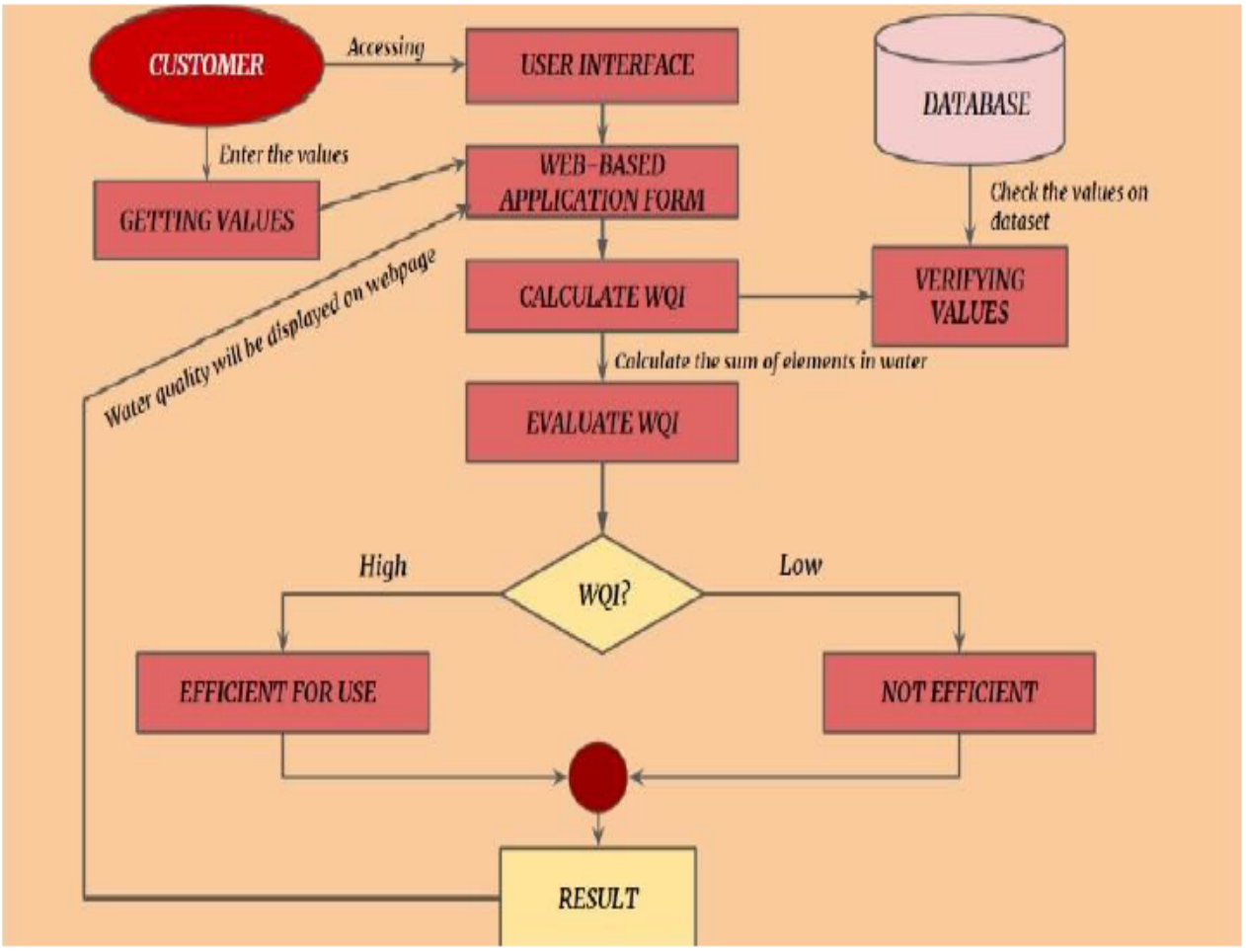
## 4.2 NON FUNCTIONAL REQUIREMENTS:

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | System is stand up to the customer's expectations. When an application is, users can easily navigate its interface. The user can determine what feature and what it can do. |
| NFR-2 | **Security** | Various forms of question for calculating the water quality index(wqi) and securely stored in database. |
| NFR-3 | **Reliability** | If the number of failures is low, it means that the system operates properly. reliability of Track the |
| | | time between critical feature can help you understand |
| NFR-4 | **Performance** | Our system should run on a 32 bit(x86) or 64 bit(x64) dual-core 2.66-GHZ or faster processor. It should not exceed 2 GB ram. |
| NFR-5 | **Availability** | The system should be available for the duration of the user access to the system until the user terminates the access. |
| NFR-6 | **Scalability** | It provides an efficient outcome and ability to increase or decrease the performance of a system based on datasets. |

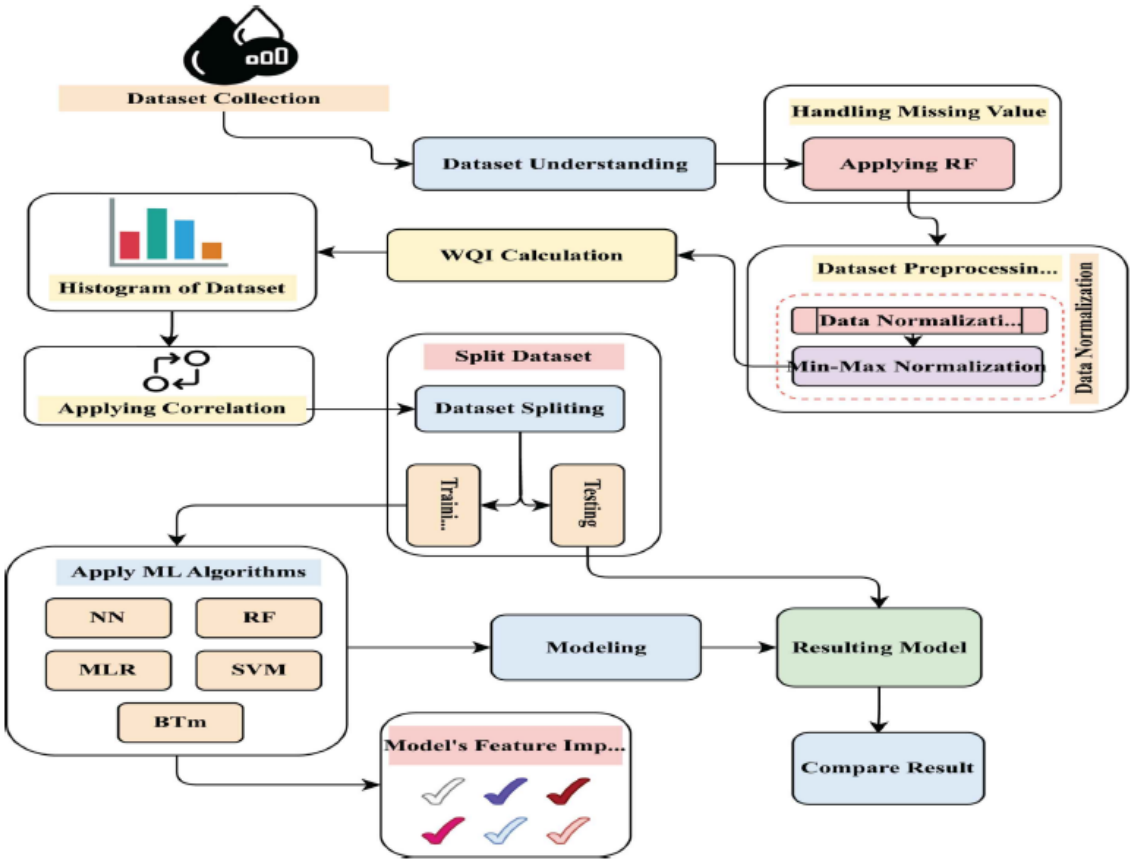
# CHAPTER 5

# PROJECT DESIGN

## 5.1 DATA FLOW DIAGRAM:




## 5.2 SOLUTION & TECHNICAL ARCHITECTURE:




## 5.3 USER STORIES:

**User Stories**

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

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer (Web user) | Access web page | USN-1 | As a user, anyone can access the web page to check the water quality. | I can access my webpage online at any time. | High | Sprint-1 |
| Customer | Usage of water | USN-2 | As per the usage of the user the quality of water should be predicted in an easy way. | Prediction can be done in an easy way. | High | Sprint-2 |
| Customer | Accuracy of water | USN-3 | Using a prediction model the user will know the quality of water on a daily basis. | The quality analysis of water will be accurate. | High | Sprint-3 |
| Administrator | Manage the web page | USN-4 | As admin, I can manage user details and update parameters essential for prediction. | Make changes on user interface(UI). | High | Sprint-3 |
| Administrator | Calculation of WQI | USN-5 | An admin can update the calculations for water quality index calculation. | Improves the accuracy of the calculation. | High | Sprint-3 |

# CHAPTER 6

# PROJECT PLANNING AND SCHEDULING

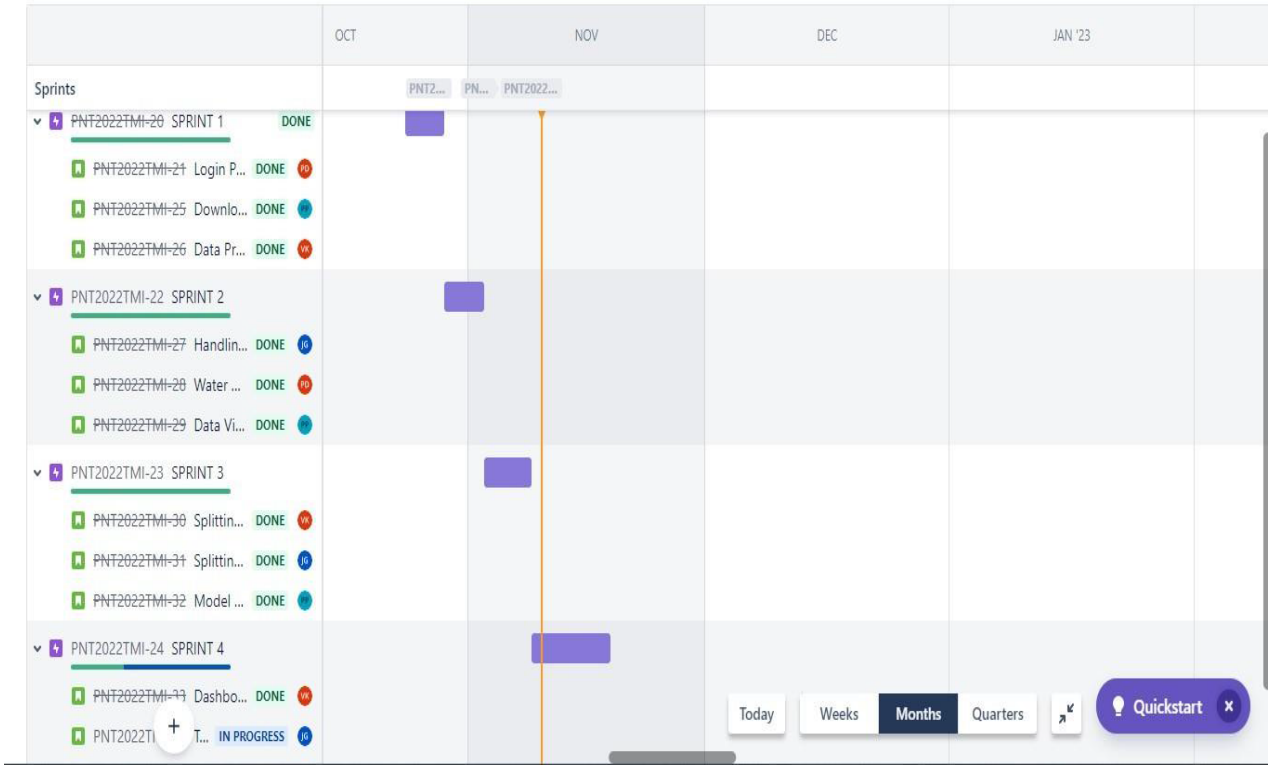
## 6.1 SPRINT PLANNING AND ESTIMATION:

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Data Preparation | USN-1 | Collecting water dataset and pre-processing it | 10 | High | S.Arunkumar G.Arunpandi |
| Sprint-1 | Model Building and Model Evaluation | USN-2 | Create an ML model to predict water quality.Calculate the performance, error rate, and complexity of the ML model and evaluate the dataset based on the parameter that the dataset consists. | 10 | High | S.Arunkumar G.Arunpandi |
| Sprint-2 | Model Deployment | USN-3 | As a user, I need to deploy the model and need to find the results. | 20 | High | T.N.Harivarshan S.Arunkumar |
| Sprint-3 | Web page (Form) | USN-4 | As a user, I can use the application by entering the water dataset to analyze or predict the results. | 20 | High | T.N.Harivarshan S.Arunkumar G.Arunpandi S.Bogar |
| Sprint-4 | Dashboard | USN-5 | As a user, I can predict the water quality by clicking the submit button and the application will show whether the water is efficient for use or not. | 20 | High | T.N.Harivarshan S.Arunkumar G.Arunpandi S.Bogar |

## 6.2 SPRINT DELIVERY SCHEDULE:

| 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 | 5 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 |

## 6.3 REPORTS FROM JIRA:




# CHAPTER 7

# CODING & SOLUTIONING

## 7.1 FEATURE 1:

The proposed system is the machine learning model where we could predict the quality of the water from giving the necessary details regarding the water body. This part deals with creating a model from the random forest algorithm. With the dataset we will be finding out the water quality index and using that we split the data into the training and testing set. Then the model will be created using the splitted data. After the model is created the accuracy of the

model will be determined and the model is deployed in the pickle. There is also another method to deploy a model using the IBM cloud.

The flask app is created to act as an interface to predict the quality from the details the user is giving,

```
import numpy as np
from flask import Flask,render_template,request
import pickle
app= Flask(__name__)
model = pickle.load(open('wqi.pkl','rb'))
@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/web')
def Home():
  return render_template("web.html")
@app.route('/quality_test')
def QualityTest():
  return render_template("quality_test.html")
@app.route('/about')
def About():
  return render_template("about.html")

@app.route('/login',methods = ['POST'])
def login() :
  year = request.form["year"]
  do = request.form["do"]
  ph = request.form["ph"]
  co = request.form["co"]
```

```
  bod = request.form["bod"]
  tc = request.form["tc"]
  na = request.form["na"]
  total = [[float(do),float(ph),float(co),float(bod),float(na),float(tc)]]
  y_pred = model.predict(total)
  y_pred = y_pred[[0]]
  if(y_pred >= 95 and y_pred<=100):
          return render_template("quality_test.html",showcase = 'Excellent, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 89 and y_pred<=94):
         return render_template("quality_test.html",showcase = 'Very Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 80 and y_pred<=88):
      return render_template("quality_test.html",showcase = 'Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 65 and y_pred<=79):
       return render_template("quality_test.html",showcase = 'Fair, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 45 and y_pred<=64):
          return render_template("quality_test.html",showcase = 'Marginal, The Predicted Value Is'+ str(y_pred))
  else:
      return render_template("quality_test.html",showcase = 'Poor, The Predicted Value Is'+ str(y_pred))



if __name__ == '__main__':
    app.run(debug = True,port=5000)
```

## 7.2 FEATURE 2:

The model is deployed in the IBM cloud with the following code:

```
import requests

import json

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

API_KEY = "tkJeJFVitIoTIn9x13pYMVA7AtDG4btmwjuOjVblFO_z"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
print("mltoken",mltoken)
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}


# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [["year","do","ph","co","bod","tc","na"]], "values": [[2014,6.7, 7.5, 203, 2, 0.1, 27.0]]}]}
```

```
response_scoring =
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/033cf98b-23
59-4eb1-b4bf-1c1a494432bd/predictions?version=2022-11-18',
json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

predict = response_scoring.json()

pred = (predict['predictions'][0]['values'][0][0])
print(pred)
```

# CHAPTER 8

# TESTING

## 8.1 TEST CASES:

- Verify that the user was able to use that web page.
- Verify that the user was able to enter the value.
- Verify that the values entered by the user are computed.
- Verify that the user was able to see the predicted value.

## 8.2.USER ACCEPTANCE TESTING:

## 8.2.1 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 | 9 | 3 | 2 | 1 | 15 |
| Duplicate | 1 | 0 | 0 | 0 | 1 |
| External | 2 | 1 | 0 | 1 | 4 |
| Fixed | 6 | 5 | 4 | 25 | 40 |
| Not Reproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 3 | 2 | 1 | 6 |
| Totals | 18 | 12 | 10 | 29 | 69 |

## 8.2.2 TEST CASE ANALYSIS:

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

| Section | Total Cases | Not Tested | Fail | Pass |
|---|---|---|---|---|
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 48 | 0 | 0 | 48 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 3 | 0 | 0 | 3 |
| Exception Reporting | 5 | 0 | 0 | 5 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

# CHAPTER 9
# RESULTS

## 9.1. PERFORMANCE METRICS:

The performance metrics are the accuracy of the model and the errors that the model is predicting.The MAE(Mean Absolute Error), MSE(Mean Squared Error) and RMSE( Root Mean Squared Error) are 0.24115288220552444,1.183839899749373 and 1.088044070683432 respectively. The accuracy of the model is 0.9935468906027789. These are the factors that determine the performance of the model.

# CHAPTER 10
# ADVANTAGES & DISADVANTAGES

## ADVANTAGES:

- water quality prediction helps in controlling Water Pollution
- To predict whether the water is safe or not.
- Predicting potable water quality for water management and water pollution prevention.
- Water quality prediction conveys the health of ecosystems, safety of human contact, extent of water pollution and condition of drinking water.

## DISADVANTAGES:

- Training necessary Somewhat difficult to manage over time and with large data sets.
- Requires manual operation to submit data, some configuration required.
- Costly, usually only feasible under Exchange Network grants Technical expertise and network server required.
- Requires manual operation to submit data Cannot respond to data queries from other nodes, and therefore cannot interact with the Exchange Network Technical expertise and network server required.

# CHAPTER 11

## CONCLUSION:

The water quality is monitored and managed effectively because of the importance of drinking water. Water has a direct effect on our health. This adds more reason to test the quality of drinking water. Several boards of committees and protocols are established to check the quality of water. The assessment of water quality differs from origin to origin. Using machine learning techniques the water quality is tested without any regular laboratory tests. By using Random Forest algorithm, we can evaluate the quality of water based on the attributes such as pH, BOD, DO, minerals and coliform in the water.This model can be used for predicting the quality of water and can monitor the potability of the water. This model acts as a prototype for the IoT sensors and can make the model even more efficient to predict the quality of water and potability of water. Data cleaning and processing, missing value analysis, exploratory analysis, and model creation and evaluation were all part of the analytical process. The best accuracy on a public test set will be discovered, as will the highest accuracy score. This application can assist in determining the current state of water quality.

# CHAPTER 12

## FUTURE SCOPE:

In future works, we propose integrating the findings of this research in a large-scale IoT-based online monitoring system using only the sensors of the required parameters. The tested algorithms would predict the water quality immediately based on the real-time data fed from the IoT system.The proposed IoT system would employ the parameter sensors of pH, turbidity, temperature and TDS for parameter readings and communicate those readings using an Arduino microcontroller. It would identify poor quality water before it is released for consumption and alert concerned authorities. It will hopefully result in curtailment of people consuming poor quality water and consequently de-escalate harrowing diseases like typhoid and diarrhea. In this regard, the application of a prescriptive analysis from the expected values would lead to future facilities to support decision and policy makers.

# CHAPTER 13

# APPENDIX:

# SOURCE CODE:

## Importing the libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

## Reading Dataset

In [2]:
```
data = pd.read_csv('water_dataX.csv',encoding='ISO-8859-1',low_memory=False)
```

## Analyse the data

In [3]:
```
data.head()
```

Out[3]:

| | STATION CODE | LOCATIONS | STATE | Temp | D.O. (mg/l) | PH | CONDUCTIVITY (µmhos/cm) | B.O.D. (mg/l) | NITRATENAN N+ NITRITENANN (mg/l) | FECAL COLIFORM (MPN/100ml) | TOTAL COLIFORM (MPN/100ml)Mean | year |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1393 | DAMANGANGA AT D/S OF MADHUBAN, DAMAN | DAMAN & DIU | 30.6 | 6.7 | 7.5 | 203 | NAN | 0.1 | 11 | 27 | 2014 |

In [4]:
```
data.describe()
```

Out[4]:

| | year |
|---|---|
| count | 1991.000000 |
| mean | 2010.038172 |
| std | 3.057333 |
| min | 2003.000000 |
| 25% | 2008.000000 |
| 50% | 2011.000000 |
| 75% | 2013.000000 |
| max | 2014.000000 |

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1991 entries, 0 to 1990
Data columns (total 12 columns):
 #   Column                            Non-Null Count  Dtype 
---  ------                            --------------  ----- 
 0   STATION CODE                      1991 non-null   object
 1   LOCATIONS                         1991 non-null   object
 2   STATE                             1991 non-null   object
 3   Temp                              1991 non-null   object
 4   D.O. (mg/l)                       1991 non-null   object
 5   PH                                1991 non-null   object
 6   CONDUCTIVITY (µmhos/cm)           1991 non-null   object
 7   B.O.D. (mg/l)                     1991 non-null   object
 8   NITRATENAN N+ NITRITENANN (mg/l)  1991 non-null   object
 9   FECAL COLIFORM (MPN/100ml)        1991 non-null   object
 10  TOTAL COLIFORM (MPN/100ml)Mean    1991 non-null   object
 11  year                              1991 non-null   int64 
dtypes: int64(1), object(11)
```

In [6]:
```
data.shape
```

Out[6]:
```
(1991, 12)
```

## Handling Missing Values 1

In [7]:
```
data.isnull().any()
```

Out[7]:
```
STATION CODE                      False
LOCATIONS                         False
STATE                             False
Temp                              False
D.O. (mg/l)                       False
PH                                False
CONDUCTIVITY (µmhos/cm)           False
B.O.D. (mg/l)                     False
NITRATENAN N+ NITRITENANN (mg/l)  False
FECAL COLIFORM (MPN/100ml)        False
TOTAL COLIFORM (MPN/100ml)Mean    False
year                              False
dtype: bool
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
STATION CODE                      0
LOCATIONS                         0
STATE                             0
Temp                              0
D.O. (mg/l)                       0
PH                                0
CONDUCTIVITY (µmhos/cm)           0
B.O.D. (mg/l)                     0
NITRATENAN N+ NITRITENANN (mg/l)  0
FECAL COLIFORM (MPN/100ml)        0
TOTAL COLIFORM (MPN/100ml)Mean    0
year                              0
```

In [9]:
```
data.dtypes
```

Out[9]:
```
STATION CODE                        object
LOCATIONS                           object
STATE                               object
Temp                                object
D.O. (mg/l)                         object
PH                                  object
CONDUCTIVITY (µmhos/cm)             object
B.O.D. (mg/l)                       object
NITRATENAN N+ NITRITENANN (mg/l)    object
FECAL COLIFORM (MPN/100ml)          object
TOTAL COLIFORM (MPN/100ml)Mean      object
year                                 int64
dtype: object
```

## Handling missing values 2

In [10]:
```
data['Temp']=pd.to_numeric(data['Temp'],errors='coerce')
data['D.O. (mg/l)']=pd.to_numeric(data['D.O. (mg/l)'],errors='coerce')
data['PH']=pd.to_numeric(data['PH'],errors='coerce')
data['B.O.D. (mg/l)']=pd.to_numeric(data['B.O.D. (mg/l)'],errors='coerce')
data['CONDUCTIVITY (µmhos/cm)']=pd.to_numeric(data['CONDUCTIVITY (µmhos/cm)'],errors='coerce')
data['NITRATENAN N+ NITRITENANN (mg/l)']=pd.to_numeric(data['NITRATENAN N+ NITRITENANN (mg/l)'],errors='coerce')
data['TOTAL COLIFORM (MPN/100ml)Mean']=pd.to_numeric(data['TOTAL COLIFORM (MPN/100ml)Mean'],errors='coerce')
data.dtypes
```

Out[10]:
```
STATION CODE                         object
LOCATIONS                            object
STATE                                object
Temp                                float64
D.O. (mg/l)                         float64
PH                                  float64
CONDUCTIVITY (µmhos/cm)             float64
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
STATION CODE                          0
LOCATIONS                             0
STATE                                 0
Temp                                 92
D.O. (mg/l)                          31
PH                                    8
CONDUCTIVITY (µmhos/cm)              25
B.O.D. (mg/l)                        43
NITRATENAN N+ NITRITENANN (mg/l)    225
FECAL COLIFORM (MPN/100ml)            0
TOTAL COLIFORM (MPN/100ml)Mean      132
year                                  0
dtype: int64
```

## Handling missing values 3

In [12]:
```
data['Temp'].fillna(data['Temp'].mean(),inplace=True)
data['D.O. (mg/l)'].fillna(data['D.O. (mg/l)'].mean(),inplace=True)
data['PH'].fillna(data['PH'].mean(),inplace=True)
data['CONDUCTIVITY (µmhos/cm)'].fillna(data['CONDUCTIVITY (µmhos/cm)'].mean(),inplace=True)
data['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].mean(),inplace=True)
data['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(data['NITRATENAN N+ NITRITENANN (mg/l)'].mean(),inplace=True)
data['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(data['TOTAL COLIFORM (MPN/100ml)Mean'].mean(),inplace=True)
```

In [13]:
```
data.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
```

In [14]:
```
data=data.rename(columns = {'D.O. (mg/l)': 'do'})
data=data.rename(columns = {'CONDUCTIVITY (µmhos/cm)': 'co'})
data=data.rename(columns = {'B.O.D. (mg/l)': 'bod'})
data=data.rename(columns = {'NITRATENAN N+ NITRITENANN (mg/l)': 'na'})
data=data.rename(columns = {'TOTAL COLIFORM (MPN/100ml)Mean': 'tc'})
data=data.rename(columns = {'STATION CODE': 'station'})
data=data.rename(columns = {'LOCATIONS': 'location'})
data=data.rename(columns = {'STATE': 'state'})
data=data.rename(columns = {'PH': 'ph'})
```

## Water Quality Index (WQI) Calculation-1

In [15]:
```
#calculation of pH
data['npH']=data.ph.apply(lambda x: (100 if(8.5>=x>=7)
                                  else(80 if(8.6>=x>=8.5) or (6.9>=x>=6.8)
                                    else (60 if(8.8>=x>=8.6) or (6.8>=x>=6.7)
                                        else(40 if(9>=x>=8.8) or (6.7>=x>=6.5)
                                            else 0)))))
```

In [16]:
```
#calculation of dissolved oxygen
data['ndo']=data.do.apply(lambda x: (100 if(x>=6)
                                  else(80 if(6>=x>=5.1)
                                    else (60 if(5>=x>=4.1)
                                        else(40 if(4>=x>=3)
                                            else 0)))))
```

## Water Quality Index (WQI) Calculation-2

In [17]:
```
#calculation of total coliform
data['nco']=data.tc.apply(lambda x: (100 if(5>=x>=0)
                                  else(80 if(50>=x>=5)
                                    else (60 if(500>=x>=50)
                                        else(40 if(10000>=x>=500)
                                            else 0)))))
```

In [18]:
```
#calculation of B.D.O
data['nbdo']=data.bod.apply(lambda x:(100 if(3>=x>=0)
                                  else(80 if(6>=x>=3)
                                    else (60 if(80>=x>=6)
                                        else(40 if(125>=x>=80)
                                            else 0)))))
```

## Water Quality Index (WQI) Calculation-3

In [19]:
```
#calculation of electric conductivity
data['nec']=data.co.apply(lambda x:(100 if(75>=x>=0)
                                  else(80 if(150>=x>=75)
                                    else (60 if(225>=x>=150)
                                        else(40 if(300>=x>=225)
                                            else 0)))))
```

In [20]:
```
#calculation of nitrate
data['nna']=data.na.apply(lambda x:(100 if(20>=x>=0)
                                  else(80 if(50>=x>=20)
                                    else (60 if(100>=x>=50)
                                        else(40 if(200>=x>=100)
                                            else 0)))))
```

In [21]:
```
#Calculation of Water Quality Index WQI
data['wph']=data.npH*0.165
data['wdo']=data.ndo*0.281
data['wbdo']=data.nbdo*0.234
data['wec']=data.nec*0.009
data['wna']=data.nna*0.028
data['wco']=data.nco*0.281
data['wqi']=data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
data
```

Out[21]:

| | station | location | state | Temp | do | ph | co | bod | na | tc | ... | nbdo | nec | nna | wph | wdo | wbdo | wec | wna | wco |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1393 | DAMANGANGA AT D/S OF MADHUBAN, DAMAN | DAMAN & DIU | 30.600000 | 6.7 | 7.5 | 203.0 | 6.940049 | 0.100000 | 27.0 | ... | 60 | 60 | 100 | 16.5 | 28.10 | 14.04 | 0.54 | 2.8 | 22.48 |
| **1** | 1399 | ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI... | GOA | 29.800000 | 5.7 | 7.2 | 189.0 | 2.000000 | 0.200000 | 8391.0 | ... | 100 | 60 | 100 | 16.5 | 22.48 | 23.40 | 0.54 | 2.8 | 11.24 |
| **2** | 1475 | ZUARI AT PANCHAWADI | GOA | 29.500000 | 6.3 | 6.9 | 179.0 | 1.700000 | 0.100000 | 5330.0 | ... | 100 | 60 | 100 | 13.2 | 28.10 | 23.40 | 0.54 | 2.8 | 11.24 |
| **3** | 3181 | RIVER ZUARI AT BORIM BRIDGE | GOA | 29.700000 | 5.8 | 6.9 | 64.0 | 3.800000 | 0.500000 | 8443.0 | ... | 80 | 100 | 100 | 13.2 | 22.48 | 18.72 | 0.90 | 2.8 | 11.24 |
| **4** | 3182 | RIVER ZUARI AT MARCAIM JETTY | GOA | 29.500000 | 5.8 | 7.3 | 83.0 | 1.900000 | 0.400000 | 5500.0 | ... | 100 | 80 | 100 | 16.5 | 22.48 | 23.40 | 0.72 | 2.8 | 11.24 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1986** | 1330 | TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU | NAN | 26.209814 | 7.9 | 738.0 | 7.2 | 2.700000 | 0.518000 | 202.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 16.86 |
| **1987** | 1450 | PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T... | NAN | 29.000000 | 7.5 | 585.0 | 6.3 | 2.600000 | 0.155000 | 315.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 16.86 |

In [22]:
```
#Calculation of overall WQI for each year
average = data.groupby('year')['wqi'].mean()
average.head()
```

Out[22]:
```
year
2003    66.239545
2004    61.290000
2005    73.762689
2006    72.909714
2007    74.233000
Name: wqi, dtype: float64
```

## Splitting Dependent and Independent Columns

In [23]:
```
data.head()
data.drop(['location','station','state'],axis =1,inplace=True)
```

In [24]:
```
data.head()
```

Out[24]:

| | Temp | do | ph | co | bod | na | tc | year | npH | ndo | ... | nbdo | nec | nna | wph | wdo | wbdo | wec | wna | wco | wqi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30.6 | 6.7 | 7.5 | 203.0 | 6.940049 | 0.1 | 27.0 | 2014 | 100 | 100 | ... | 60 | 60 | 100 | 16.5 | 28.10 | 14.04 | 0.54 | 2.8 | 22.48 | 84.46 |
| **1** | 29.8 | 5.7 | 7.2 | 189.0 | 2.000000 | 0.2 | 8391.0 | 2014 | 100 | 80 | ... | 100 | 60 | 100 | 16.5 | 22.48 | 23.40 | 0.54 | 2.8 | 11.24 | 76.96 |
| **2** | 29.5 | 6.3 | 6.9 | 179.0 | 1.700000 | 0.1 | 5330.0 | 2014 | 80 | 100 | ... | 100 | 60 | 100 | 13.2 | 28.10 | 23.40 | 0.54 | 2.8 | 11.24 | 79.28 |
| **3** | 29.7 | 5.8 | 6.9 | 64.0 | 3.800000 | 0.5 | 8443.0 | 2014 | 80 | 80 | ... | 80 | 100 | 100 | 13.2 | 22.48 | 18.72 | 0.90 | 2.8 | 11.24 | 69.34 |
| **4** | 29.5 | 5.8 | 7.3 | 83.0 | 1.900000 | 0.4 | 5500.0 | 2014 | 100 | 80 | ... | 100 | 80 | 100 | 16.5 | 22.48 | 23.40 | 0.72 | 2.8 | 11.24 | 77.14 |

5 rows × 21 columns

In [41]:
```
x=data.iloc[:,1:7].values
```

In [42]:
```
x.shape
```

Out[42]: (1991, 6)

In [64]:
```
y=data.iloc[:,20:].values
y.shape
```

Out[64]: (1991, 1)

In [65]:
```
print(x)
```

```
[[16.5  28.1  14.04  0.54  2.8  22.48]
 [16.5  22.48 23.4   0.54  2.8  11.24]
 [13.2  28.1  23.4   0.54  2.8  11.24]
 ...
 [ 0.   28.1  23.4   0.9   2.8  11.24]
 [ 0.   28.1  23.4   0.9   2.8  11.24]
 [ 0.   28.1  23.4   0.9   2.8  11.24]]
```

In [66]:
```
print(y)
```

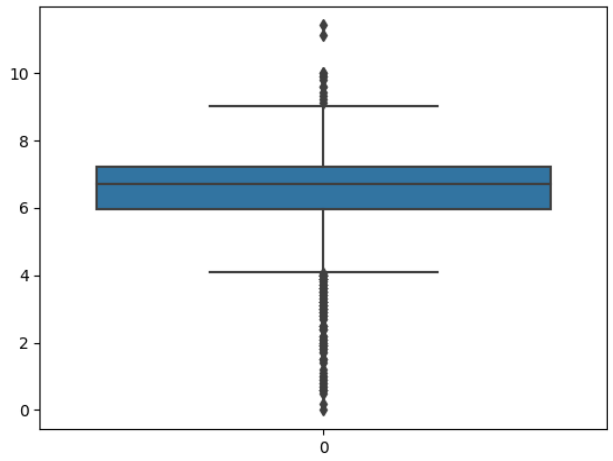
```
[[84.46]
 [76.96]
 [79.28]
 ...
 [66.44]
 [66.44]
 [66.44]]
```

## Data Visualization

In [30]:
```
import seaborn as sns
```

In [32]:
```
sns.boxplot(data["do"])
```

Out[32]: <AxesSubplot:>

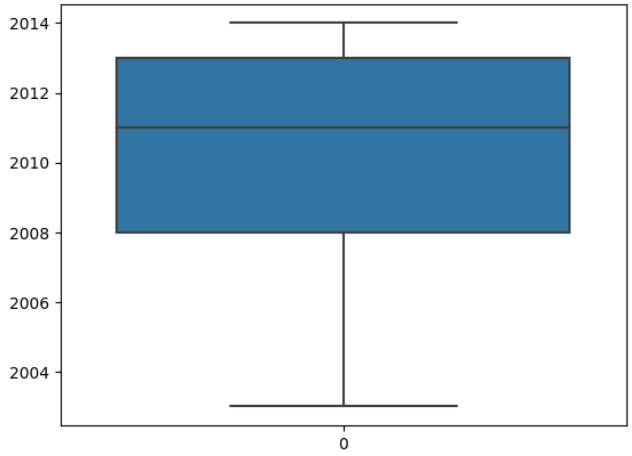



In [33]:
```
sns.boxplot(data["year"])
```

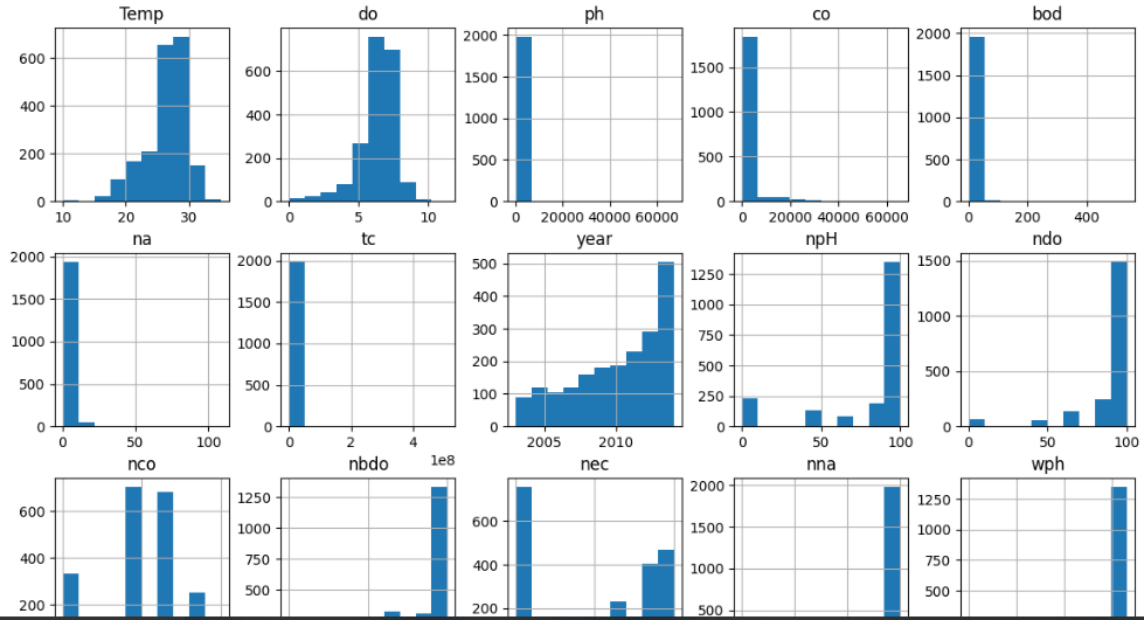
Out[33]: <AxesSubplot:>

In [33]:
```
sns.boxplot(data["year"])
```

Out[33]: <AxesSubplot:>




In [34]:
```
data.hist(figsize=(14,14))
plt.show()
```



## Splitting the Data Into Train and Test

In [67]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=10)
```

## Random_Forest_Regression

In [68]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [73]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train.ravel())
y_pred = regressor.predict(x_test)
```

## Model Evaluation

In [74]:
```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
MAE: 0.24115288220552444
MSE: 1.183839899749373
RMSE: 1.088044070683432
```

In [75]:
```
#accuracy of the model
metrics.r2_score(y_test, y_pred)
```

Out[75]:
```
0.9935468906027789
```

## Save the model

In [72]:
```
import pickle
pickle.dump(regressor,open('wqi.pkl', 'wb'))
model=pickle.load(open('wqi.pkl','rb'))
```

# APP.PY:

```python

import numpy as np
from flask import Flask,render_template,request
import pickle
app= Flask(__name__)
model = pickle.load(open('wqi.pkl','rb'))
@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/web')
def Home():
  return render_template("web.html")
@app.route('/quality_test')
def QualityTest():
  return render_template("quality_test.html")
@app.route('/about')
def About():
  return render_template("about.html")

@app.route('/login',methods = ['POST'])
def login() :
  year = request.form["year"]
  do = request.form["do"]
  ph = request.form["ph"]
  co = request.form["co"]
  bod = request.form["bod"]
  tc = request.form["tc"]
  na = request.form["na"]
  total = [[float(do),float(ph),float(co),float(bod),float(na),float(tc)]]
  y_pred = model.predict(total)
  y_pred = y_pred[[0]]
  if(y_pred >= 95 and y_pred<=100):
    return render_template("quality_test.html",showcase = 'Excellent, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 89 and y_pred<=94):
    return render_template("quality_test.html",showcase = 'Very Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 80 and y_pred<=88):
    return render_template("quality_test.html",showcase = 'Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 65 and y_pred<=79):
    return render_template("quality_test.html",showcase = 'Fair, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 45 and y_pred<=64):
    return render_template("quality_test.html",showcase = 'Marginal, The Predicted Value Is'+ str(y_pred))
  else:
    return render_template("quality_test.html",showcase = 'Poor, The Predicted Value Is'+ str(y_pred))


if __name__ == '__main__':
    app.run(debug = True,port=5000)
```

## ABOUT.HTML:

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1">
<link rel="stylesheet" href="static\css\about.css">
</head>

<body>

<ul>
  <li><a href="web">Home</a></li>
  <li><a href="quality_test">Quality Test</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="#about">About</a></li>
  <center><h1 style="color:Black;font-size:20px">Efficient Of Water Quality Analysis & Prediction</h1></center>
</ul><br><br>
<div class="container">

<p>Dissolved oxygen (DO) :</p>
<p>     Dissolved oxygen (DO) is a measure of how much oxygen is dissolved in the water - the amount of oxygen available to living aquatic organisms. The amount o
<p>potential hydrogen (ph):</p>
<p>     pH is a measure of how acidic/basic water is. The range goes from 0 to 14, with 7 being neutral. pHs of less than 7 indicate acidity, whereas a pH of grea
<p>Conductivity:</p>
<p>     Conductivity is a measure of the ability of water to pass an electrical current. Because dissolved salts and other inorganic chemicals conduct electrical
<p>Biochemical Oxygen Demand:</p>
<p>     BOD is a measure of the amount of oxygen required to remove waste organic matter from water in the process of decomposition by aerobic bacteria (those bac
<p>Nitrate-N :</p>
<p>     Nitrate-N is found naturally in soil and water, but usually at relatively low concentrations (less than 4 mg/L in water). However, nitrate is highly solub
<p>Total coliform :</p>
<p>     Total coliform counts give a general indication of the sanitary condition of a water supply. Total coliforms include bacteria that are found in the soil,
</div>
</body>

</html>
```

## QUALITY_TEST.HTML:

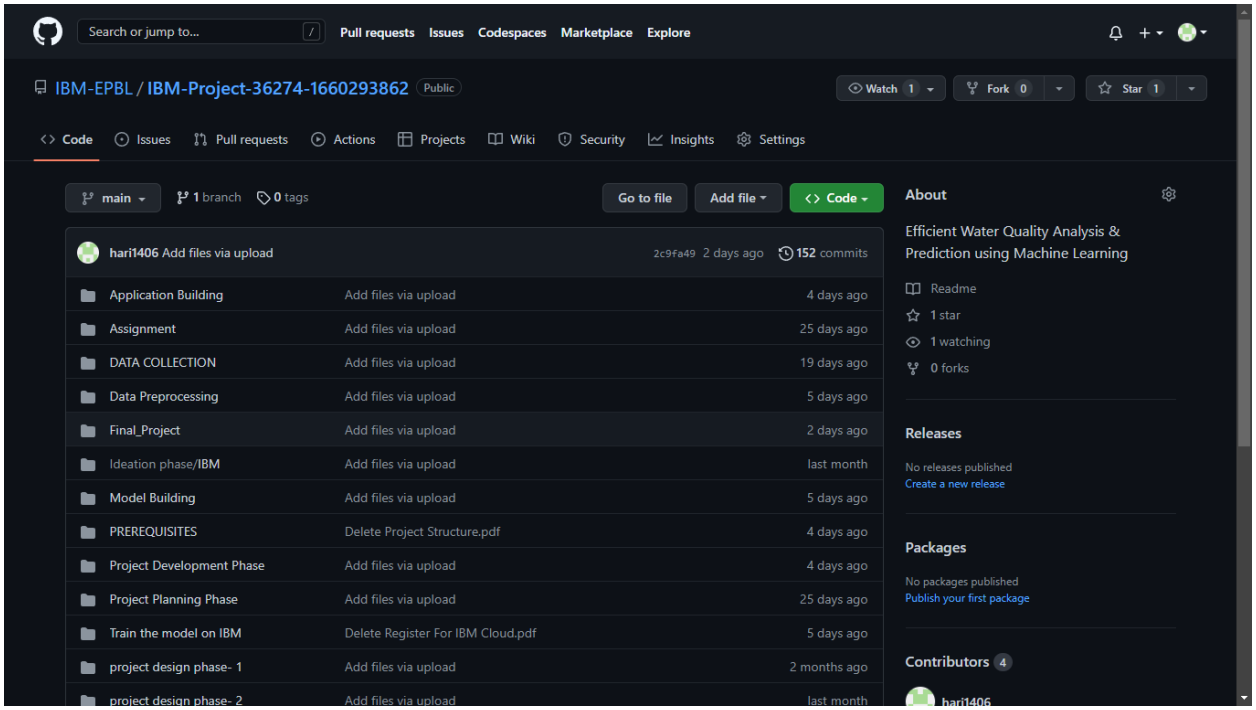
```html
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1">
<link rel="stylesheet" href="static\css\quality_test.css">
</head>

<body>

<ul>
  <li><a href="web">Home</a></li>
  <li><a href="#quality">Quality Test</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="about">About</a></li>
  <center><h1 style="color:Black;font-size:20px">Efficient Of Water Quality Analysis & Prediction</h1></center>
</ul><br>


 <form action="/login" method="POST">

   <div class="container">

    <div class="col-25">
      <label for="enter the year">Enter The Year:</label>
    </div>
    <select id="Year" name="year">
    <option value="2003">2003</option>
    <option value="2004">2004</option>
    <option value="2005">2005</option>
    <option value="2006">2006</option>
    <option value="2007">2007</option>
    <option value="2008">2008</option>
    <option value="2009">2009</option>
    <option value="2010">2010</option>
    <option value="2011">2011</option>
    <option value="2012">2012</option>
    <option value="2013">2013</option>
```

```html
  </select><br>

  <input type="text" id="Dissolved Oxygen" name="do" placeholder="Enter D.O" pattern="[0-9]+([\.,][0-9]+)?" required><br>


  <input type="text" id="Potential of Hydrogen" name="ph" placeholder="Enter PH" pattern="[0-9]+([\.,][0-9]+)?" required><br>


  <input type="text" id="Conductivity" name="co" placeholder="Enter Conductivity" pattern="[0-9]+([\.,][0-9]+)?" required><br>


  <input type="text" id="Biochemical  Oxygen Demand" name="bod" placeholder="Enter B.O.D" pattern="[0-9]+([\.,][0-9]+)?" required><br>


  <input type="text" id="Nitratenan" name="na" placeholder="Enter Nitratenan" pattern="[0-9]+([\.,][0-9]+)?" required><br>


  <input type="text" id="Total Coliform" name="tc" placeholder="Enter Total Coliform" pattern="[0-9]+([\.,][0-9]+)?" required><br>


  <input type="submit"  value="Submit"><br><br>


  <div class="result" id="result">
      {{showcase}}
  </div>




 </div>
</form>

</body>
```

## WEB.HTML:

```html
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1">
<link rel="stylesheet" href="static\css\web.css">
</head>

<body>



<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="quality_test">Quality Test</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="about">About</a></li>
  <center><h1 style="color:Black;font-size:20px">Efficient Of Water Quality Analysis & Prediction</h1></center>
</ul><br>
<div class="container">

<p1>"The amount of water on the planet does not change,only its quality."</p1><br>
<p2> Making testing of water sample simple,faster and easier.</p2>


<button class="btn" id="myButton">Let's predict water quality</button>
<script type="text/javascript">
    document.getElementById("myButton").onclick = function () {
        location.href = "quality_test";
 };
</script>
</div>
</body>
</html>
```

## GITHUB:




**PROJECT DEMO LINK:**

https://github.com/IBM-EPBL/IBM-Project-36274-1660293862/blob/main/Final_Project/project%20video%20record.mp4