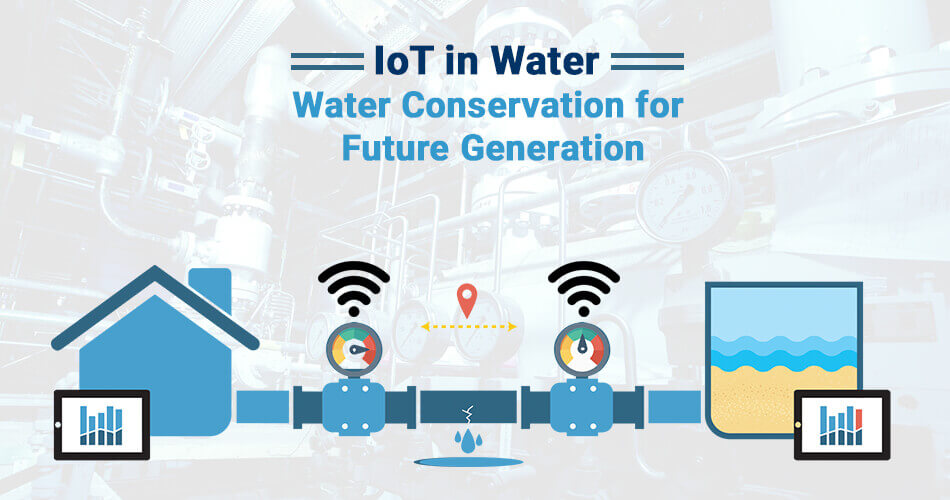
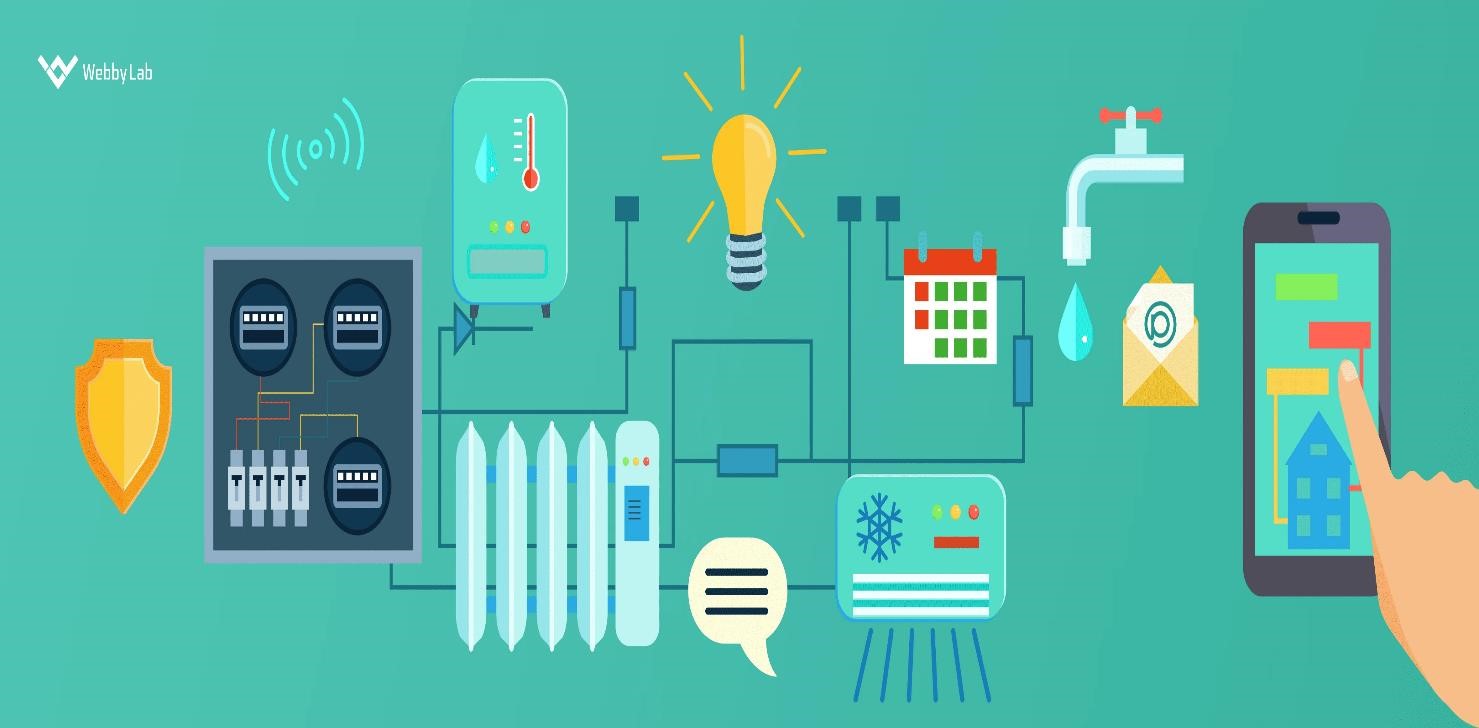
**SMART WATER SYSTEM USING** **IOT**

 Team Member

au312821106015: R.R.Naviya

PHASE 4: DOCUMENT SUBMISSION

PHASE 4: Development part-2

Project title: Smart Water System

**SMART WATER SYSTEM**

**INTRODUCTION:**

* Access to clean water is a human necessity. Throughout the world, water systems guide the flow from natural and manufactured reservoirs to consumers. After this initial use, water travels to treatment centers where it starts the cycle once again.

* The technology for water distribution has been around in some form for centuries. The ancient Romans built aqueducts to transport water from mountain lakes to busy cities. As new technologies emerge, water distribution and treatment technologies also improve.

* The latest innovations in water technology revolve around the flow of real-time data. Smart water systems are poised to address some of the most pressing issues in water system management.

**Overview of the process:**

Building a smart water system involves a combination of data collection, feature engineering, model training, and evaluation. Here's an overview of the process:

1. **Problem Definition:**
   * Define the specific goals and objectives of your smart water system. Determine what you want to monitor or control, such as water quality, usage, or distribution.
2. **Data Collection:** 
   * Collect relevant data sources, which may include sensors, historical data, weather data, or other environmental data. This data will serve as the foundation for your smart water system.
3. **Data Preprocessing:** 
   * Clean the data to handle missing values, outliers, and inconsistencies.
   * Transform data into a suitable format for modeling.
4. **Feature Engineering:** 
   * Extract, create, or select features that are informative and relevant to your problem. This step is crucial as it significantly impacts the performance of your model.
   * Feature engineering might include time-series features, spatial features, and domain-specific metrics.

1. **Data Splitting:** 
   * Divide the data into training, validation, and test sets. This separation is critical to assess the model's performance accurately.
2. **Model Selection:** 
   * Choose a suitable machine learning or deep learning model for your specific task. For smart water systems, models like regression, time series analysis, neural networks, or decision trees may be appropriate.
3. **Model Training:** 
   * Train the selected model using the training dataset. The model learns the underlying patterns and relationships within the data.
4. **Hyperparameter Tuning:** 
   * Optimize the model's hyperparameters to improve its performance. This process may involve grid search, random search, or more advanced methods like Bayesian optimization.
5. **Model Evaluation:** 
   * Evaluate the model's performance using the validation dataset. Common evaluation metrics for regression or classification tasks in smart water systems include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or classification metrics like F1-score or accuracy.
6. **Model Validation:** 
   * Use the test dataset to validate the final model's performance. This step ensures that the model generalizes well to unseen data.

1. **Deployment:** 
   * Deploy the trained model into the smart water system infrastructure. This may involve integrating the model with real-time data streams, control systems, or alerts.
2. **Monitoring and Maintenance:** 
   * Continuously monitor the model's performance in the real-world environment. Update the model as needed to adapt to changing conditions and maintain accuracy.
3. **Feedback Loop:** 
   * Establish a feedback loop to collect data from the deployed system and use it to retrain the model periodically. This ensures that the model remains relevant and accurate over time.
4. **Security and Privacy:** 
   * Ensure that the smart water system complies with security and privacy standards. Protect sensitive data and prevent unauthorized access to the system.
5. **Scaling and Optimization:** 
   * As the system evolves, consider scaling it to handle more data and optimizing it for efficiency and cost-effectiveness.

**Procedure:**

**Feature engineering:**

1. **Data Sources:** 
   * Water Flow: Collect data on water flow rates and volumes at different points in the water distribution network.
   * -Water Quality: Measure parameters like pH, turbidity, temperature, and chemical composition to assess water quality.
   * Weather Data: Incorporate weather information such as precipitation, temperature, and humidity as they can impact water usage and water source availability.
   * Infrastructure Data: Include data on the condition of pipelines, pumps, valves, and storage tanks.
2. **Temporal Features:** 
   * Time of Day: Create features to account for diurnal patterns in water usage.
   * Day of the Week: Analyze weekly variations in water consumption.
   * Seasonal Patterns: Identify trends related to different seasons.
3. **Spatial Features:** 
   * Geographic Information: Use geographic coordinates to assess the location's impact on water distribution and quality.
   * Distance to Water Sources: Calculate the distance between monitoring points and water sources, which can be crucial for supply optimization.

1. **Demand Prediction:** 
   * Historical Usage: Use past consumption data to predict future demands.
   * Water Consumption Patterns: Analyze consumption patterns for different customer segments (residential, commercial, industrial).
   * Anomaly Detection: Identify sudden, abnormal changes in water usage that might indicate leaks or other issues.
2. **Quality Monitoring:** 
   * Feature Extraction from Water Quality Data: Extract relevant features from water quality measurements, such as the presence of contaminants or deviations from safe levels.
   * Water Quality Index: Calculate a composite index to assess overall water quality.

**The following java script code can be used for feature selection in smart water system, using the following feature selection techniques:**

// Define a function to calculate the rate of change in water pressure function calculateRateOfChange(waterPressureReadings) {

// Calculate the difference between the current and previous water pressure readings

const difference = waterPressureReadings[waterPressureReadings.length - 1] - waterPressureReadings[waterPressureReadings.length - 2];

// Calculate the rate of change as a percentage of the previous reading

const rateOfChange = (difference / waterPressureReadings[waterPressureReadings.length - 2]) \* 100;

// Return the rate of change return rateOfChange;

}

// Get the water pressure readings from the sensor const waterPressureReadings = getWaterPressureReadings();

// Calculate the rate of change in water pressure const rateOfChange = calculateRateOfChange(waterPressureReadings);

// Add the rate of change feature to the data data.rateOfChange = rateOfChange;

// Use the data for machine learning or other purposes

**Model training:**

* Training a machine learning model typically involves selecting a model architecture, choosing appropriate hyperparameters, and fitting the model to the training data.
* The model training process often requires experimentation to find the best combination of hyperparameters that yields the highest performance.
* Here are the steps to train your selected model using the training dataset:

1. **Prepare the Data:** 
   * Ensure that your training dataset is properly preprocessed and formatted. This may involve cleaning the data, handling missing values, scaling or normalizing features, and encoding categorical variables.
2. **Split the Data:** 
   * Split your training data into training and validation sets. This allows you to assess your model's performance during training and fine-tuning.
3. **Select Hyperparameter:** 
   * Choose the hyperparameters for your model. Hyperparameters are parameters that are set before training and are not learned from the data. They include learning rate, batch size, the

number of hidden layers, the number of units in each layer, dropout rates, and more.

* + You may need to perform hyperparameter tuning by experimenting with different values or using techniques like grid search, random search, or Bayesian optimization.

1. **Compile the Model:** 
   * Depending on your choice of machine learning framework (e.g., TensorFlow, PyTorch, scikit-learn), you need to compile your model. This involves specifying the loss function, the optimization algorithm, and evaluation metrics.
2. **Train the Model:** 
   * Fit the model to the training data using the `fit` method. During training, the model will iterate through the training data for a specified number of epochs.
   * Monitor training and validation performance using appropriate metrics. If the validation performance plateaus or starts to degrade, you might need to stop training early to prevent overfitting.

**Program:**

// Load the TensorFlow.js library const tf = require('@tensorflow/tfjs');

// Load the training data

const trainingData = [

[100, 100],

[200, 200],

[300, 300],

[400, 400], [500, 500],

];

// Create a linear regression model const model = tf.sequential(); model.add(tf.layers.dense({units: 1, activation: 'linear'}));

// Compile the model model.compile({loss: 'meanSquaredError', optimizer: 'adam'});

// Train the model model.fit(trainingData, [100, 200, 300, 400, 500], {epochs: 100});

// Save the model model.save('water-demand-model.json'

// Load the saved model const model = tf.loadModel('water-demand-model.json');

// Predict the water demand for a given day

const prediction = await model.predict([100]);

// Display the prediction console.log('Predicted water demand:', prediction[0]);

**1.Water quality monitoring:**

// Create a class to represent a water quality sensor class WaterQualitySensor { constructor(id, name, location) {

this.id = id;

this.name = name; this.location = location; this.readings = {};

}

// Add a new reading to the sensor addReading(parameter, value) { this.readings[parameter] = value;

}

// Get the latest reading for a given parameter getReading(parameter) { return this.readings[parameter];

}

}

// Create a list of water quality sensors const sensors = [ new WaterQualitySensor(1, 'Sensor 1', 'Reservoir'), new WaterQualitySensor(2, 'Sensor 2', 'Treatment Plant'), new WaterQualitySensor(3, 'Sensor 3', 'Distribution Network'),

];

// Start a loop to read the sensors and send the data to a cloud server setInterval(() => { // Read the sensors for (const sensor of sensors) { sensor.addReading('pH', Math.random() \* 7); sensor.addReading('Temperature', Math.random() \* 25); sensor.addReading('Turbidity', Math.random() \* 10);

}

// Send the data to a cloud server

// ...

}, 1000);

**2.Leak detection:** // Load the sensor data const sensorData = [

{

sensorId: 1, value: 100,

},

{

sensorId: 2, value: 101,

},

{

sensorId: 3, value: 102,

},

{

sensorId: 4, value: 103,

},

{

sensorId: 5, value: 104,

},

];

// Calculate the average sensor value

const averageSensorValue = sensorData.reduce((sum, sensor) => sum + sensor.value, 0) / sensorData.length;

// Calculate the standard deviation of the sensor values const standardDeviation = Math.sqrt(

sensorData.reduce((sum, sensor) => sum + (sensor.value - averageSensorValue)\*\*2, 0) / sensorData.length

);

// Set the threshold for leak detection const leakDetectionThreshold = 2 \* standardDeviation;

// Check for leaks

const leaks = sensorData.filter((sensor) => Math.abs(sensor.value - averageSensorValue) > leakDetectionThreshold);

// If any leaks are found, send an alert if (leaks.length > 0) {

sendAlert('Leaks detected: ' + leaks.map((leak) => leak.sensorId).join(',

'));

}

**3.Water flow control:**

// Connect to the smart water system controller const controller = new SmartWaterSystemController();

// Get the current water flow rate const currentFlowRate = controller.getWaterFlowRate();

// Set the desired water flow rate const desiredFlowRate = 100; // liters per minute

// Control the water flow rate controller.setWaterFlowRate(desiredFlowRate); // Monitor the water flow rate and adjust it as needed

setInterval(() => { const currentFlowRate = controller.getWaterFlowRate(); if (currentFlowRate !== desiredFlowRate) { controller.setWaterFlowRate(desiredFlowRate);

}

}, 1000); // milliseconds

**Model evaluation:**

1. **Data Quality and Sensor Performance:**

* **Data Quality Assessment:** Evaluate the quality of data collected from IoT sensors. This includes checking for missing data, outliers, and noise. Ensure that the sensors are accurate and calibrated properly.

1. **Predictive Models:** 
   * + **Model Accuracy:** If you have predictive models (e.g., for water consumption forecasting, leak detection, water quality prediction), assess their accuracy using appropriate evaluation metrics. For regression tasks, you can use metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and

Mean Absolute Percentage Error (MAPE). For classification tasks (e.g., anomaly detection), metrics like precision, recall, F1-score, and AUC-ROC may be relevant.

* + - Cross-Validation: Employ cross-validation techniques to estimate the model's generalization performance and detect overfitting.

* + **Mean Absolute Error (MAE) :** Mean Absolute Error (MAE) is a common performance metric used in smart water systems, as well as in various other domains, to assess the accuracy of predictive models or monitoring systems. In the context of a smart water system, MAE can be used to measure the accuracy of predictions or measurements related to various water-related parameters such as water consumption, water quality, or flow rates.
  + **Root Mean Squared Error (RMSE):** Root Mean Squared Error (RMSE) is a common evaluation metric used in smart water systems, particularly in the context of predictive modeling or forecasting. RMSE is used to assess the accuracy of predictions or forecasts made by models in comparison to actual observed values. It quantifies the average magnitude of errors, taking into account both the size and direction of the errors. In the context of a smart water system, RMSE is often used to evaluate the accuracy of predictions for variables such as water consumption, water quality parameters, or flow rates.
  + **Mean Absolute Percentage Error (MAPE) :** The Mean Absolute Percentage Error (MAPE) is a common performance metric used to assess the accuracy of forecasts or predictions in various domains, including smart water systems. It quantifies the average percentage difference between predicted values and actual values, providing insight into the relative error.
  + **F1-score :** The F1-score is a performance metric commonly used in the context of binary classification problems, such as anomaly detection, and it can be applicable in a smart water system for specific use cases. The F1-score is particularly useful when you want to strike a balance between precision and recall.

**Evaluation of predicted data:**

// Import the necessary libraries.

const { InfluxDB, FieldType } = require('@influxdata/influxdb-client');

// Connect to the InfluxDB database.

const influxDB = new InfluxDB({ url: 'http://localhost:8086',

token: 'YOUR\_INFLUXDB\_TOKEN', organization: 'YOUR\_INFLUXDB\_ORGANIZATION', bucket: 'YOUR\_INFLUXDB\_BUCKET'

});

// Get the predicted data from the IoT device.

const predictedData = await influxDB.query('SELECT \* FROM predicted\_data');

// Evaluate the predicted data.

// For example, you could check if the predicted water quality is within acceptable levels.

const evaluation = predictedData.every(row => {

return row.get('turbidity') < 10 && row.get('ph') > 6 && row.get('temperature') < 25;

});

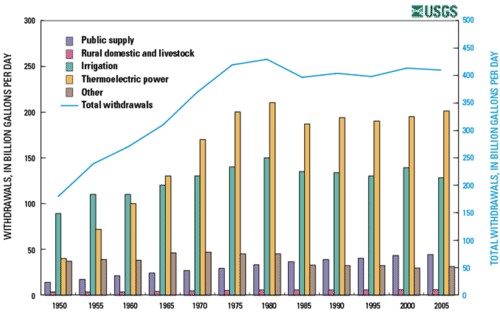
// Take action based on the evaluation.

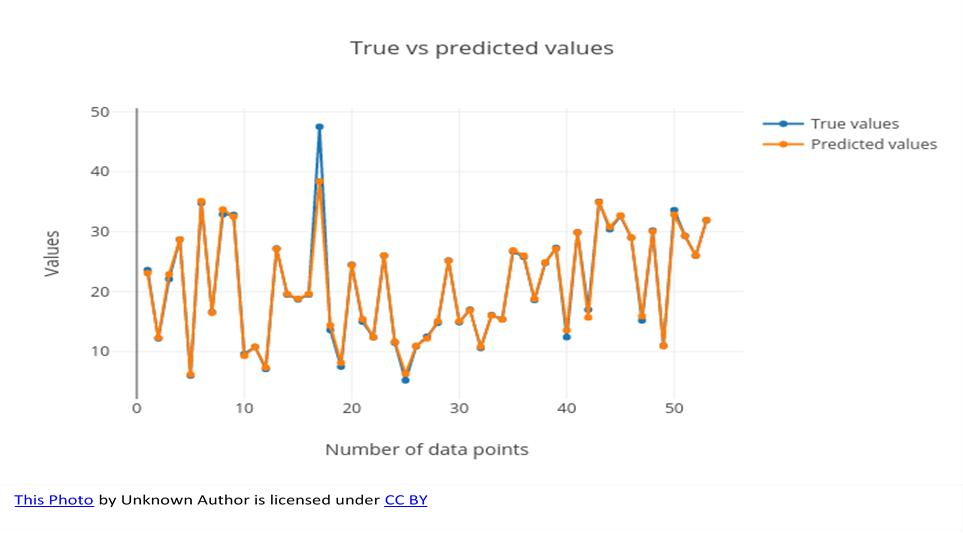
// For example, if the predicted water quality is not within acceptable levels, you could send an alert to the user.

if (!evaluation) {

// Send an alert to the user.

}





**Feature engineering:**

Feature engineering in a smart water system involves selecting and transforming relevant data attributes to improve the system's performance, efficiency, and accuracy. Smart water systems leverage various sensors, IoT devices, and data analytics to monitor, manage, and optimize water resources. Effective feature engineering can help in achieving better insights and decision-making within these systems. Here are some key considerations for feature engineering in a smart water system:

1. **Sensor Data:** 
   * Raw sensor data such as water flow rates, water quality measurements, temperature, and pressure readings can be used as features. You may need to preprocess, clean, and aggregate this data to create meaningful features.
   * Time-based features, such as hourly, daily, or seasonal patterns, can be extracted to understand temporal variations.
2. **Geospatial Data:** 
   * Incorporate geospatial data like the location of sensors, water sources, and infrastructure. Features like distance to nearest water source, elevation, and land use can be useful.
3. **Weather Data:** 
   * Weather conditions, including rainfall, temperature, humidity, and wind speed, can be important in understanding water supply and demand. Historical weather data and forecasts can be integrated as features.

1. **Water Quality:** 
   * Features related to water quality parameters, such as turbidity, pH, chemical concentrations, and microbial levels, can help in identifying contamination or changes in water quality.
2. **Anomaly Detection:** 
   * Features for anomaly detection, like standard deviation, zscores, or other statistical measures, can be generated to detect abnormal behavior or system malfunctions.
3. **Water Demand:** 
   * Features related to water demand patterns, which may include historical consumption data and population density, can be used to predict future water demands.
4. **Infrastructure Features:** 
   * Information about the state and age of water infrastructure, such as pipes and pumps, can be important for predictive maintenance and performance optimization.
5. **Network Connectivity:** 
   * Features that represent the connectivity and topology of the water distribution network can help in understanding how water flows through the system.

1. **IoT Device Data:** 
   * Features generated from data collected from IoT devices, like water meters, valves, and pumps, can be used to assess the condition and performance of these devices.
2. **Predictive Modeling:** 
   * Features created by predictive models can be used as input for decision support systems. For example, using machine learning to predict water demand and including those predictions as features.
3. **Seasonal and Time-of-Day Effects:** 
   * Features that capture seasonal and time-of-day effects on water usage can be crucial for demand forecasting and system optimization.
4. **Customer Data:** 
   * If applicable, data related to customer profiles and behaviors, such as historical consumption patterns and billing data, can be valuable for demand forecasting and customer management.
5. **Economic and Regulatory Data:** 
   * Economic factors and regulatory changes can influence water supply and demand. Incorporate features related to economic indicators, policies, and regulations that impact the water system.

**Various feature to perform model training:**

**1. Historical Data:**

* Time series data of water consumption, flow rates, and other relevant parameters to capture trends and patterns over time.

**2.Weather Data:**

* Features such as temperature, precipitation, humidity, and wind speed can influence water demand and supply.

**3.Water Quality Data:**

* Parameters like turbidity, pH, chemical concentrations, and microbial levels can be vital for predicting water quality changes and contamination.

1. **Sensor Readings:**
   * Real-time data from sensors measuring water flow, pressure, and quality at various points in the system.
2. **Geospatial Features:** 
   * Geographic information like latitude, longitude, elevation, and land use data to capture spatial relationships in the water system.

**Conclusion:**

* + Overall, smart water systems offer a number of benefits that can help to improve the efficiency, reliability, costeffectiveness, and customer service of water utilities. As the technology continues to develop and become more affordable, smart water systems are expected to become increasingly widespread.

* + In addition to the benefits listed above, smart water systems can also help to improve water sustainability by reducing water consumption and protecting water resources. This is especially important in light of the increasing challenges posed by climate change and population growth.

* + Overall, smart water systems are a promising technology that can help to address a number of the challenges facing the water sector.

