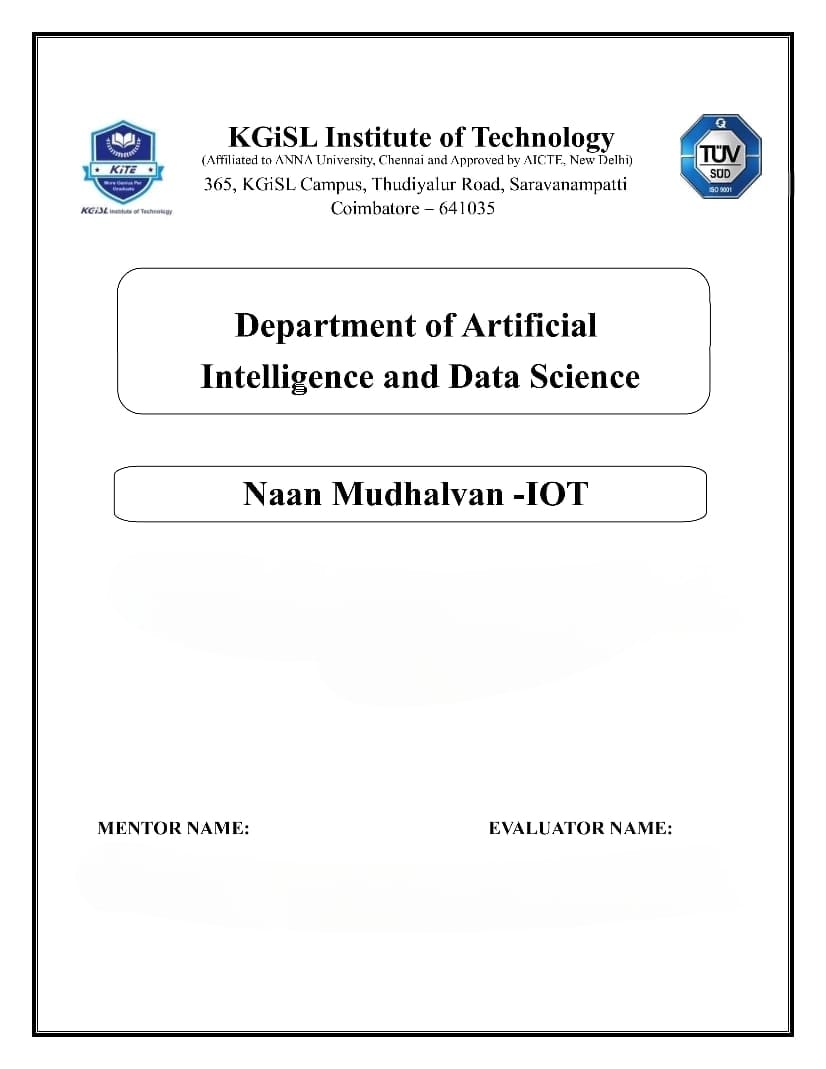
****

**PROBLEM STATEMENT : SMART WATER MANAGEMENT**

Mr.MOHANKUMAR.M

Ms.AKILANDESWARI.M

**SMART WATER MANAGEMENT**

**Frontend Development:**

User Interface (UI):

The UI is designed with a responsive layout, ensuring compatibility across various devices such as desktops, tablets, and mobile phones. The navigation is intuitively structured for seamless user interaction, featuring user dashboards that prominently display real-time air quality metrics.

Data Visualization:

To enhance user understanding, the platform employs diverse data visualization techniques. It includes charts, graphs, and numerical representations for both real-time and historical data. Additionally, interactive maps are incorporated to visually represent geographical variations in air quality.

User Interaction:

Interactive forms enable users to set preferences and thresholds. User-triggered actions, such as data downloads, are facilitated through buttons or toggles. Notifications are prominently displayed to alert users about relevant updates or alerts. Accessibility considerations adhere to standards, ensuring inclusivity for users with disabilities and those with diverse needs.

**Backend Development:**

Server-Side Logic:

The backend efficiently handles HTTP requests and responses, incorporating robust business logic for the processing and analysis of air quality data.

Database Integration:

Efficient storage of real-time and historical data is achieved through seamless integration with a suitable Database Management System (DBMS). Query optimization strategies are implemented to enhance data retrieval performance.

API Development:

RESTful API endpoints are established to facilitate communication with the frontend. Authentication and authorization mechanisms ensure secure data interaction.

**Web Development Technologies:**

Frontend Framework:

Dynamic UIs are constructed using frontend frameworks like React.js, Vue.js, or Angular.

Backend Framework:

Node.js, in conjunction with Express, Django, Flask, or a suitable alternative, serves as the backend framework.

Database:

Storage is facilitated by MongoDB, PostgreSQL, MySQL, or another database system based on project requirements.

Data Encryption:

Both data transmission and storage benefit from encryption, complemented by the implementation of secure authentication practices.

****

**FULL PROCEDURE:**

**Smart water management in IoT (Internet of Things) involves using sensors, data analytics, and automation to monitor and control water resources efficiently. This can be applied to various use cases such as agriculture, urban water supply, industrial processes, and more. Here's a procedure for implementing smart water management in IoT:**

**Define Objectives and Use Cases: Clearly define the objectives of your smart water management system. Determine the specific use cases you want to address, whether it's optimizing irrigation, reducing water wastage in a city, or monitoring water quality in industrial processes.**

**Sensor Deployment: Deploy a network of sensors to collect data about water-related parameters. These sensors can include flow meters, water level sensors, water quality sensors, and weather sensors. Ensure that the sensors are compatible with IoT protocols and can transmit data to a central server or cloud platform.**

**Data Collection and Transmission: Collect data from the sensors and transmit it to a central data storage or cloud platform. IoT communication protocols like MQTT or HTTP can be used to ensure efficient data transfer.**

**Data Storage and Management: Store the collected data in a centralized database or cloud storage. Ensure data security and redundancy to prevent data loss.**

**Data Analysis and Visualization: Implement data analytics algorithms to process and analyze the collected data. Use tools like machine learning and artificial intelligence to gain insights from the data. Create dashboards and visualization tools to present the data in a user-friendly format for stakeholders.**

**Real-time Monitoring: Set up a real-time monitoring system that can alert you to critical events or anomalies in the water system. This can include setting up thresholds for specific parameters and sending notifications when they are exceeded.**

**Control and Automation: Implement control mechanisms to automate actions based on the data and analysis. For example, you can use IoT devices to remotely control irrigation systems, valves, or pumps to optimize water usage.**

**Remote Access and Mobile Apps: Develop mobile apps or web interfaces that allow authorized personnel to access and control the system remotely. This enables quick responses to changing conditions.**

**Data Security: Ensure robust security measures to protect the data and the system from cyber threats. Use encryption, authentication, and access controls to safeguard the IoT infrastructure.**

**Scalability: Design the system with scalability in mind, so it can accommodate additional sensors and devices as needed to expand the scope of your water management.**

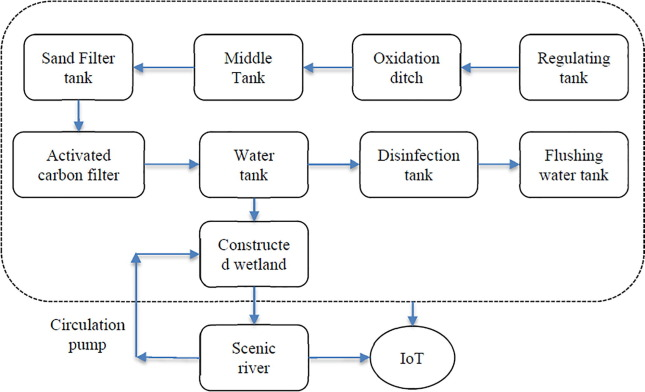
**Testing and Validation: Conduct thorough testing to ensure the system functions as expected. This includes testing sensors, data transmission, analytics, and control mechanisms.**

**Regulatory Compliance: Ensure that your smart water management system complies with local and national regulations related to water usage, data privacy, and environmental standards.**

**Maintenance and Support: Establish a maintenance plan to ensure that sensors and equipment are functioning correctly. Provide ongoing support to address any technical issues or updates.**

**Data Optimization: Continuously optimize your data collection and analytics to improve efficiency and resource utilization. This can involve refining algorithms, upgrading sensors, or adjusting control parameters.**

**Data Feedback Loop: Use the insights gained from data analysis to make informed decisions and improve water management strategies. Regularly review the system's performance and adjust it as necessary.**

****

**SOURCE CODE:**

#define \_DISABLE\_TLS\_

#include &lt;ThingerESP8266.h&gt;

#include &lt;ESP8266WiFi.h&gt;

#include &lt;SPI.h&gt;

#include &lt;Wire.h&gt;

#include &lt;Adafruit\_GFX.h&gt;

#include &lt;Adafruit\_SSD1306.h&gt;

#define OLED\_RESET LED\_BUILTIN

#define USERNAME &quot;test123&quot;

#define DEVICE\_ID &quot;SWM&quot;

#define DEVICE\_CREDENTIAL &quot;ABCDEFGHIJ&quot;

#define SSID &quot;test123&quot;

#define SSID\_PASSWORD &quot;test123&quot;

Adafruit\_SSD1306 display(OLED\_RESET);

byte indikator = 13;

byte sensorInt = 0;

byte flowsensor = D3;

float konstanta = 4.5; //konstanta flow meter

volatile byte pulseCount;

float debit;

float harga;

unsigned int flowmlt;

Unsigned long totalmlt;

unsigned long old-time;

ThingerESP8266 thing(USERNAME, DEVICE\_ID, DEVICE\_CREDENTIAL);

void setup()

{

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);

display.clearDisplay();

display.display();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 0);

// Inisialisasi port serial

Serial.begin(9600);

pinMode(flowsensor, INPUT);

pulseCount = 0;

debit = 0.0;

flowmlt = 0;

totalmlt = 0;

old-time = 0;

harga = 0.0;

// digital pin control example (i.e. turning on/off a light, a relay, configuring

aparameter, etc)

thing[&quot;sensor&quot;] &gt;&gt; [](pson&amp; out){

digitalWrite(flowsensor, HIGH);

attachInterrupt(digitalPinToInterrupt(D3), pulseCounter, FALLING);

out[&quot;debit&quot;] = debit;

out[&quot;volume&quot;] = totalmlt;

out[&quot;harga&quot;] = harga;

};

}

void loop()

{

thing.handle();

display.clearDisplay();

if((millis() - old-time) &gt; 1000)

{

detachInterrupt(sensorInt);

debit = ((1000.0 / (millis() - old-time)) \* pulseCount) / konstanta;

old-time = millis();

flowmlt = (debit / 60) \* 1000;

totalmlt += flowmlt;

harga = totalmlt\*0.002;

unsigned int frac;

Serial.print(&quot;Debit air: &quot;);

Serial.print(int(debit));

Serial.print(&quot;L/min&quot;);

Serial.print(&quot;\t&quot;);

display.setCursor(0, 0);

display.print(&quot;Debit air: &quot;);

display.setCursor(60, 0);

display.print(int(debit));

display.setCursor(85, 0);

display.print(&quot;L/min&quot;);

Serial.print(&quot;Volume: &quot;);

Serial.print(totalmlt);

Serial.print(&quot;mL&quot;);

Serial.print (&quot;\t&quot;);

display.setCursor(0, 12);

display.print(&quot;Volume: &quot;);

display.setCursor(50, 12);

display.print (totalmlt);

display.setCursor(100, 12);

display.print(&quot;mL&quot;);

display.print(&quot;\t&quot;);

Serial.print(&quot;Harga: &quot;);

Serial.print(&quot;Rp &quot;);

Serial.println(harga);

display.setCursor(0, 24);

display.print(&quot;Harga: &quot;);

display.setCursor(45, 24);

display.print(&quot;Rp &quot;);

display.setCursor(70, 24);

display.println(harga);

display.display();

pulseCount = 0;

attachInterrupt(digitalPinToInterrupt(D3), pulseCounter, FALLING);

}

}

void pulseCounter()

{

// Increment the pulse counter

pulseCount++;

THANK YOU