Requirements Analysis for Smart Water Supply System

1. Introduction

A Smart Water Supply System is an IoT-based solution designed to monitor, manage, and optimize water distribution. The system ensures efficient water usage, prevents wastage, and detects leaks or anomalies in real time. This document outlines the requirements for building a functional Smart Water Supply System.

2. System Requirements

2.1 Functional Requirements

- **Water Flow Monitoring:** Measure and track water flow in pipelines.
- **Leak Detection:** Identify leaks and send alerts.
- **Water Level Monitoring:** Measure water levels in storage tanks.
- **Quality Monitoring:** Detect contaminants or changes in water quality.
- **Remote Control:** Enable remote shutoff or regulation of water flow.
- **Data Logging & Reporting:** Store water usage data for analytics.
- **Alerts & Notifications:** Notify users in case of abnormalities (low levels, leakage, or quality issues).
- **User Interface:** Provide a dashboard for monitoring and controlling the system.

2.2 Non-Functional Requirements

- **Scalability:** The system should support multiple locations and users.
- **Reliability:** Ensure continuous operation with minimal downtime.
- **Security:** Implement encryption and authentication for data transmission.
- **Low Power Consumption:** Devices should be energy-efficient for long-term operation.

3. IoT Devices Required

- **Water Flow Sensors** – Measure the flow rate in pipes.
- **Ultrasonic Sensors** – Monitor water levels in storage tanks.
- **pH & Turbidity Sensors** – Assess water quality.
- **Solenoid Valves** – Enable remote control of water supply.
- **Microcontrollers (e.g., Arduino, ESP8266, Raspberry Pi)** – Process sensor data and communicate with the cloud.
- **GSM/Wi-Fi Modules** – Facilitate network connectivity.

4. Software Requirements

- **Embedded Programming** – C/C++ for microcontroller-based sensor integration.
- **Backend Development** – Flask (Python) or Express (Node.js) for API services.

- **Database** – Firebase, MySQL, or PostgreSQL for storing water supply data.
- **Frontend** – Web or mobile dashboard using React, Angular, or Flutter.
- **MQTT Broker** – Mosquitto, HiveMQ, or EMQX for real-time messaging.
- **Cloud Services (Optional)** – AWS IoT, Google Cloud IoT, or ThingSpeak for data analytics.

5. Data Management

- **Water Usage Data:** Time-stamped records of consumption.
- **Sensor Readings:** Flow rate, pressure, water quality parameters.
- **Alerts & Notifications:** Logs of warnings, leak alerts, and actions taken.
- **User Profiles:** Information about registered users managing the system.

6. Real-Time Communication Requirements

- **MQTT Protocol:** For lightweight, real-time messaging between devices.
- **REST APIs:** For remote access, data retrieval, and control functionalities.
- **WebSocket Communication:** For real-time updates in web dashboards.
- **Push Notifications & SMS Alerts:** Immediate alerts to users upon issue detection.

7. Conclusion

Implementing a Smart Water Supply System requires a combination of hardware, software, and networking solutions to ensure efficient, real-time monitoring and management. This document provides the foundational requirements needed for designing an effective system.