Software Design Document (SDD)

1. System Overview

The system follows a three-tier architecture:

1. Frontend: HTML,CSS,Javascript (UI & user interaction).
2. Backend: Node.js, Python, SQL(API handling).
3. Database: MySQL, PHP, XAMPP (data storage & retrieval).

2. Architectural Design

Components:

* API Layer – Handles requests between frontend & database.
* Database Layer – Stores sensor readings & user data.
* Machine Learning Module – Detects anomalies in sensor readings.

3. API Design

| Endpoint | Method | Description |
| --- | --- | --- |
| /api/login | POST | User authentication |
| /api/register | POST | Create new user |
| /api/historical-data | GET | Fetch past sensor readings |
| /api/real-time-data | GET | Retrieve live sensor readings |
| /api/anomaly-detection | POST | ML-based anomaly detection |

4. Database Schema

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

role VARCHAR(50) CHECK (role IN ('admin', 'operator', 'user'))

);

CREATE TABLE sensor\_data (

id SERIAL PRIMARY KEY,

sensor\_id VARCHAR(50),

line INTEGER,

temperature FLOAT,

timestamp TIMESTAMP

);

5. Use Case Diagram

[To be drawn]

6. Testing Plan

| Test Case | Expected Outcome | Pass/Fail |
| --- | --- | --- |
| User login with incorrect credentials | Displays error message | [To be filled] |
| Fetching real-time sensor data | Returns live sensor readings | [To be filled] |
| Anomaly detection API | Flags out-of-bound values | [To be filled] |

7. Deployment Strategy

* Development: Local PostgreSQL & Node.js environment.

8. Next Steps

* Confirm API framework with client.
* Develop frontend wireframes.
* Set up GitHub repository for code management.