**Smart Water Monitoring System**

**Author: Talha**

**Introduction**

**Purpose & Problem :**  
Water pipeline leaks result in significant resource loss—globally, utilities lose an estimated 15–30% of treated water before it reaches consumers. These leaks often go undetected until they cause costly damage and service disruption. Our project delivers a **real-time, geolocated leak detection system**, combining IoT sensing, machine learning, and GIS mapping to empower municipalities and facility managers with faster, data-driven response capabilities.

**Solution Overview :**

* **Sensors** measure key parameters (pressure, flow, temperature, vibration, motor RPM) and transmit data (via ESP8266 or similar) to a backend server.
* A **Flask API** ingests this sensor data in real-time and applies a trained **XGBoost ML model** to detect leaks.
* A **Streamlit dashboard** visualizes predictions and logs events on an interactive GIS map.

**Key Results**

* The XGBoost model shows strong performance (e.g., accuracy ≥  99 %, ROC‑AUC ≥  100%) on labeled test data.
* Rapid **location-based leak detection** potentially reduces non‑revenue water loss and speeds maintenance responses.
* All components are designed for **local operation**, with CSV-based logging—no cloud dependencies.

**Recommendations & Next Steps :**

* Integrate **SMS/email alerting** for immediate response notification.
* Connect real sensor hardware (ESP8266 + sensors), calibrate thresholds in-field.
* Upgrade storage to cloud platforms (e.g., Firebase, SQL) for broader scalability.
* Containerize components with Docker and implement CI/CD pipelines.
* Add model monitoring to detect drift and ensure long-term model robustness.

**Commercial & Social Impact :**

This system enables **early leak detection**, reducing water waste and enabling targeted repair interventions. Adopting this solution supports sustainable water management, operational efficiency, and improved infrastructure reliability—especially critical in aging urban water network.

# **2. Dataset & Preprocessing :**

- **Dataset :** `location\_aware\_gis\_leakage\_dataset.csv`

- **Features :**

- `pressure`

- `flow rate`

- `temperature`

- `vibration`

- `motor rpm`

- `operational hours`

- `latitude`, `longitude`

- `leakage detected` (target variable)

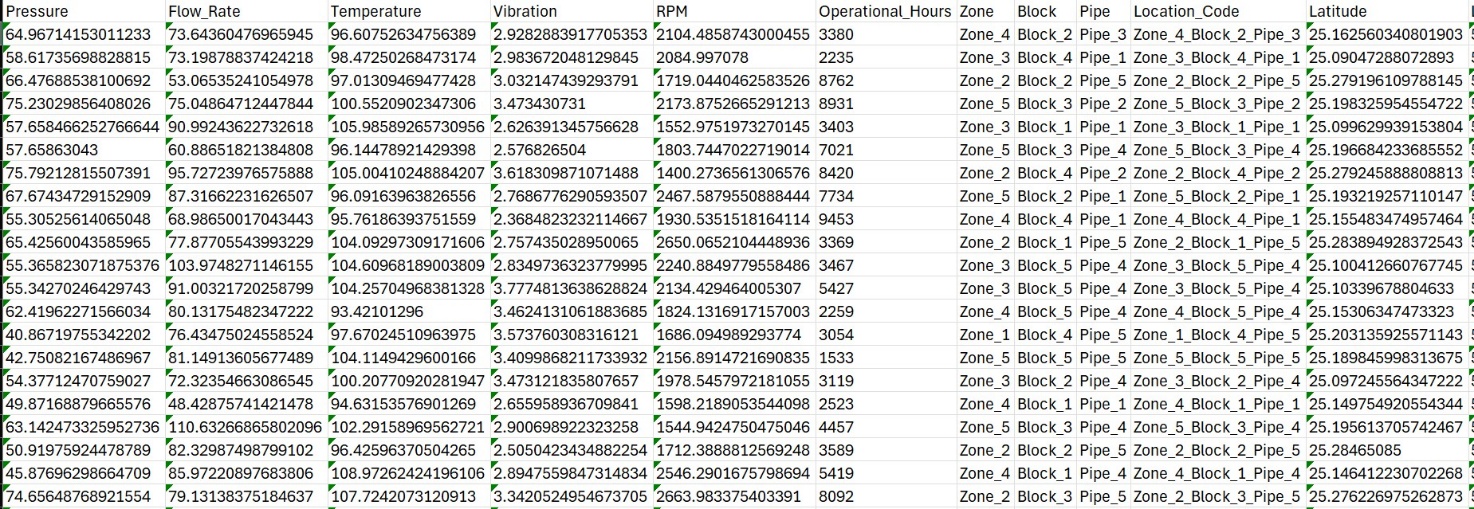
# **Preprocessing Steps :**

- Handled missing values

- Feature scaling using `scaler.joblib`

- Train–test split and label encoding

# **Sample Data Snapshot :**



# **3. Feature Engineering & Model Training**

**Input Features :**

Core sensor readings + coordinates

**Model Used :**

XGBoost classifier, trained via `model\_training.py`

**Artifacts Output :**

`xgboost\_model.joblib`

`scaler.joblib`

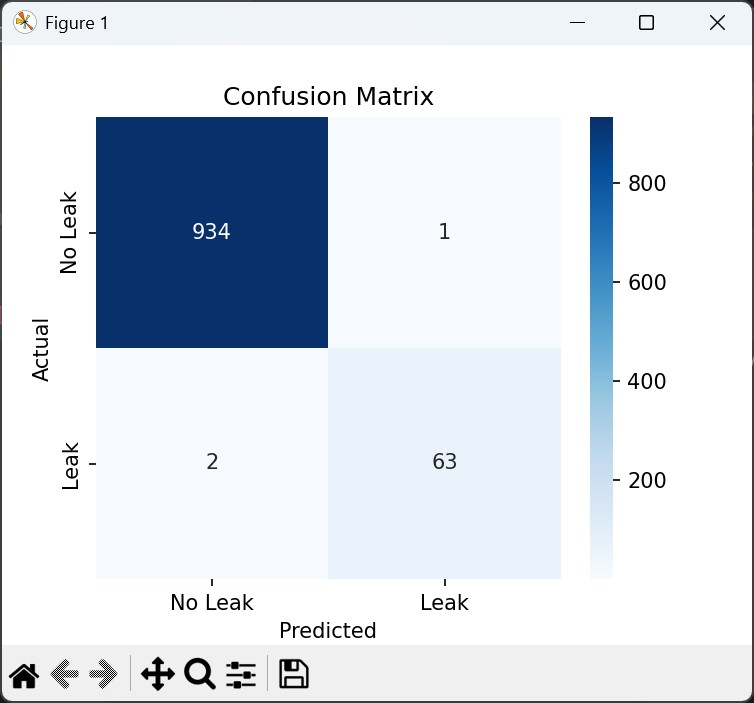
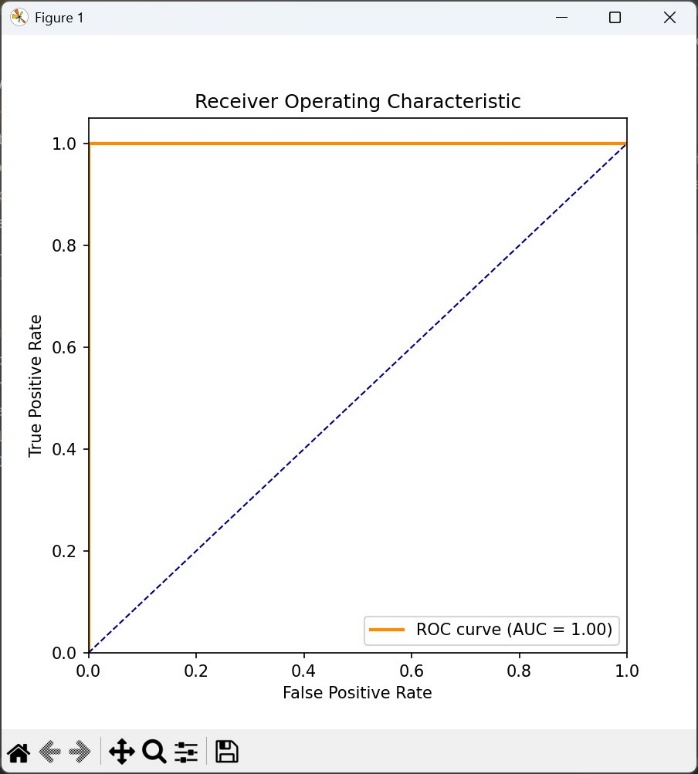
# **4. Model Evaluation :**

**Accuracy :** 99%

**Precision / Recall / F1‑Score :** 98%/96%/97%

**ROC‑AUC :** 100%

# **ROC curve and confusion matrix images:**



# **5. Deployment & API Design :**

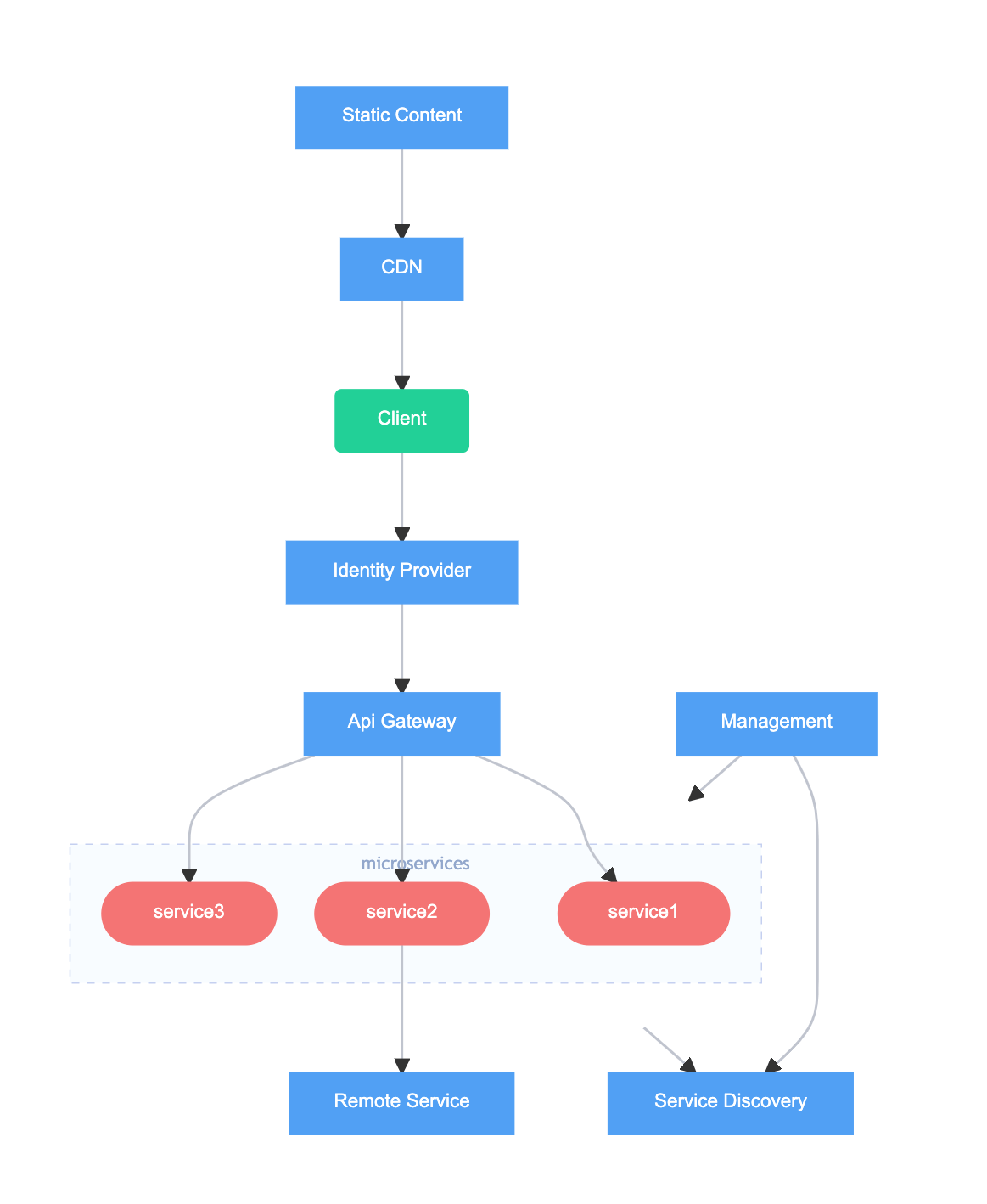
**Framework :** Flask (`app.py`)

**Endpoint :** `/predict`

**Input :** JSON with sensor data + latitude/longitude

**Output :** leak probability, leak label, timestamp

**Logging :** Results stored in `leakage\_log.csv`



# **6. Dashboard & Visualization**

**Front-end :** Streamlit (`dashboard.py`)

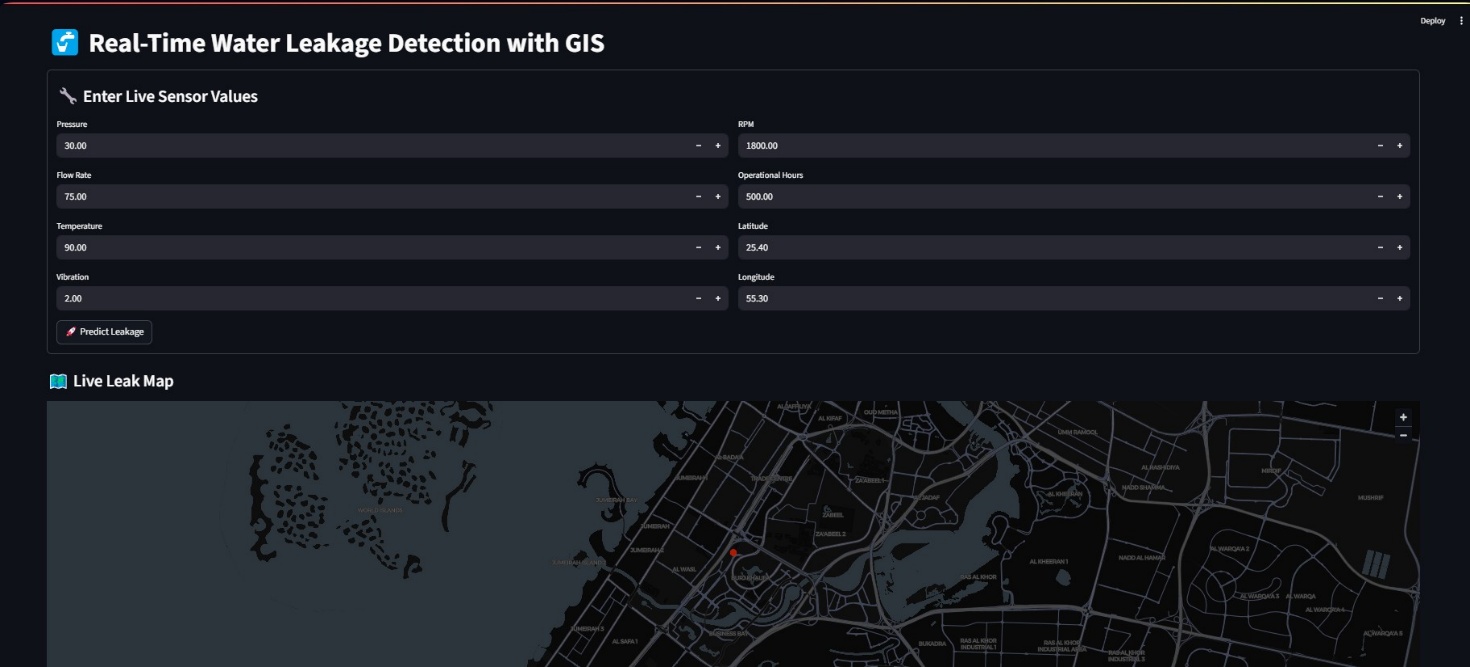
**Features :**

- Real-time sensor input

- Leak prediction output

- GIS map of leak event logs

- Data Source: Local CSV log file



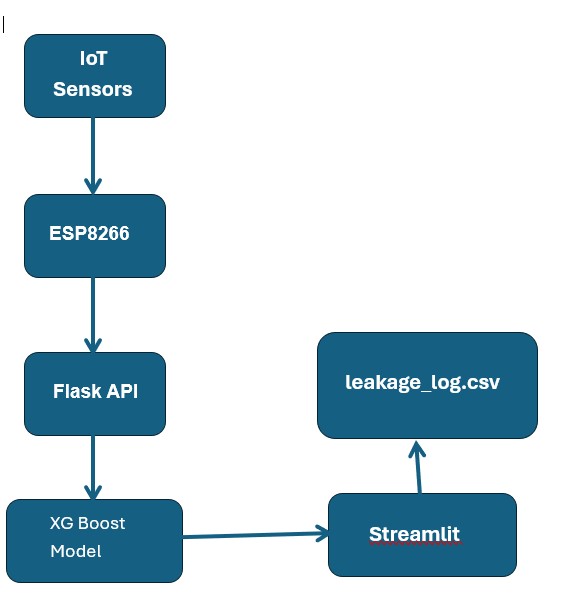
# **7. Hardware & System Architecture**

**Sensors Used :** Flow, pressure, vibration, motor RPM, temperature

**Controller :** ESP8266 (assumed) sending data to Flask API

**System Flow :**

[IoT Sensors] → [ESP8266] → [Flask API] → [XGBoost Model] → [Streamlit UI & CSV Logs]



---

# **8. Cloud Integration & Alerting**

**Current Storage :** Local CSV files

**Potential Enhancements :**

Add email/SMS alerting

Migrate to cloud storage solutions (e.g. Firebase, SQL database)

---

# **9. Deployment Guide**

**1. Install dependencies:**

pip install -r requirements.txt

**Train the model:**

python model\_training.py

**Launch API:**

python app.py

**Run the dashboard**:

streamlit run dashboard.py

**Intended Users**: IoT/data engineers, utility operators, facility managers

**Calibration Required:** Real-world sensor thresholds should be tuned based on environment

# **10. Business Value & Impact :**

1. Enables early leak detection, reducing water loss
2. Provides geospatial insights for targeted maintenance
3. Cuts operational costs and enhances sustainability
4. Supports proactive, data-driven utility management

# **11. Limitations & Future Work :**

1. No real hardware integration yet
2. Alerting system not yet implemented
3. Storage is local—no cloud scalability
4. Model monitoring and drift detection yet to be added
5. Recommend containerization (Docker) and CI/CD pipelines for production readiness

# **Appendix :**

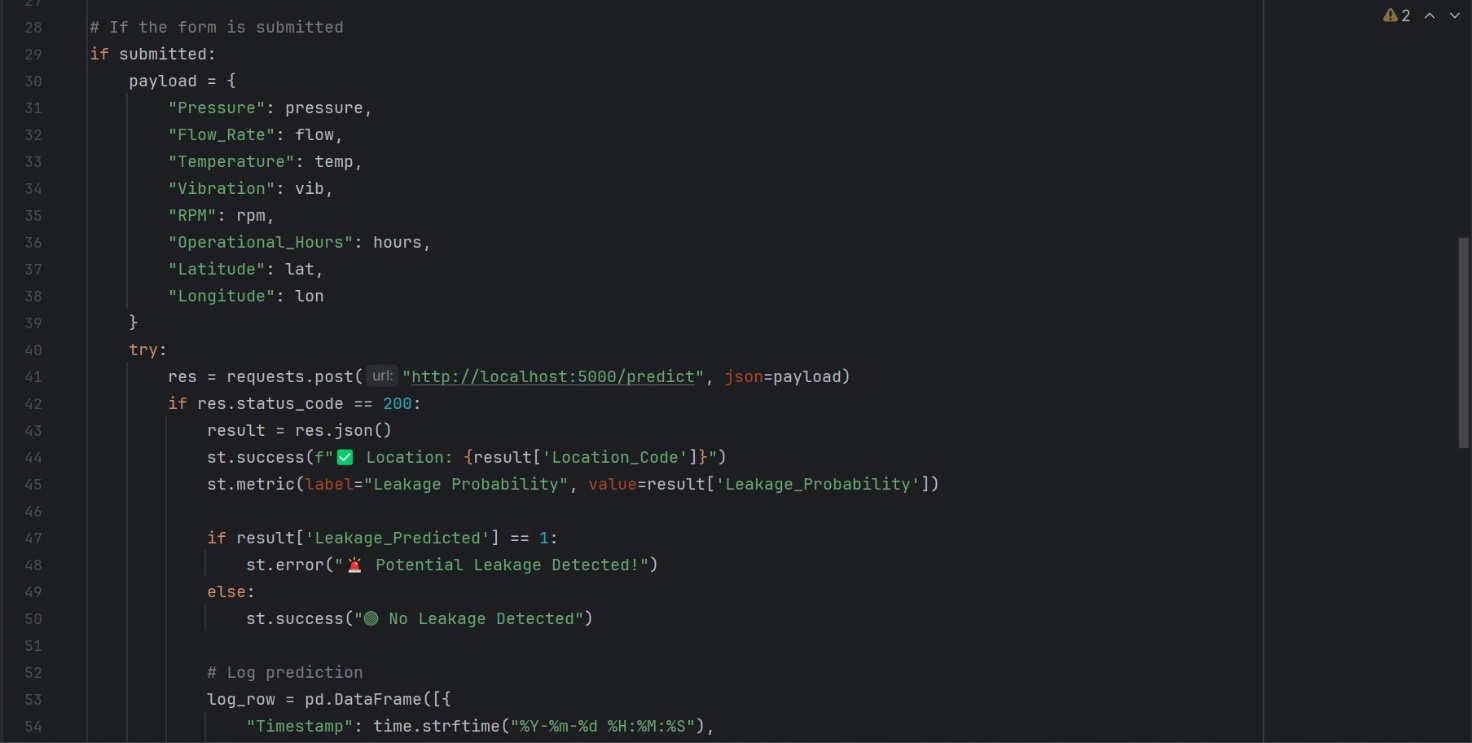
1. **Feature Dictionary :**

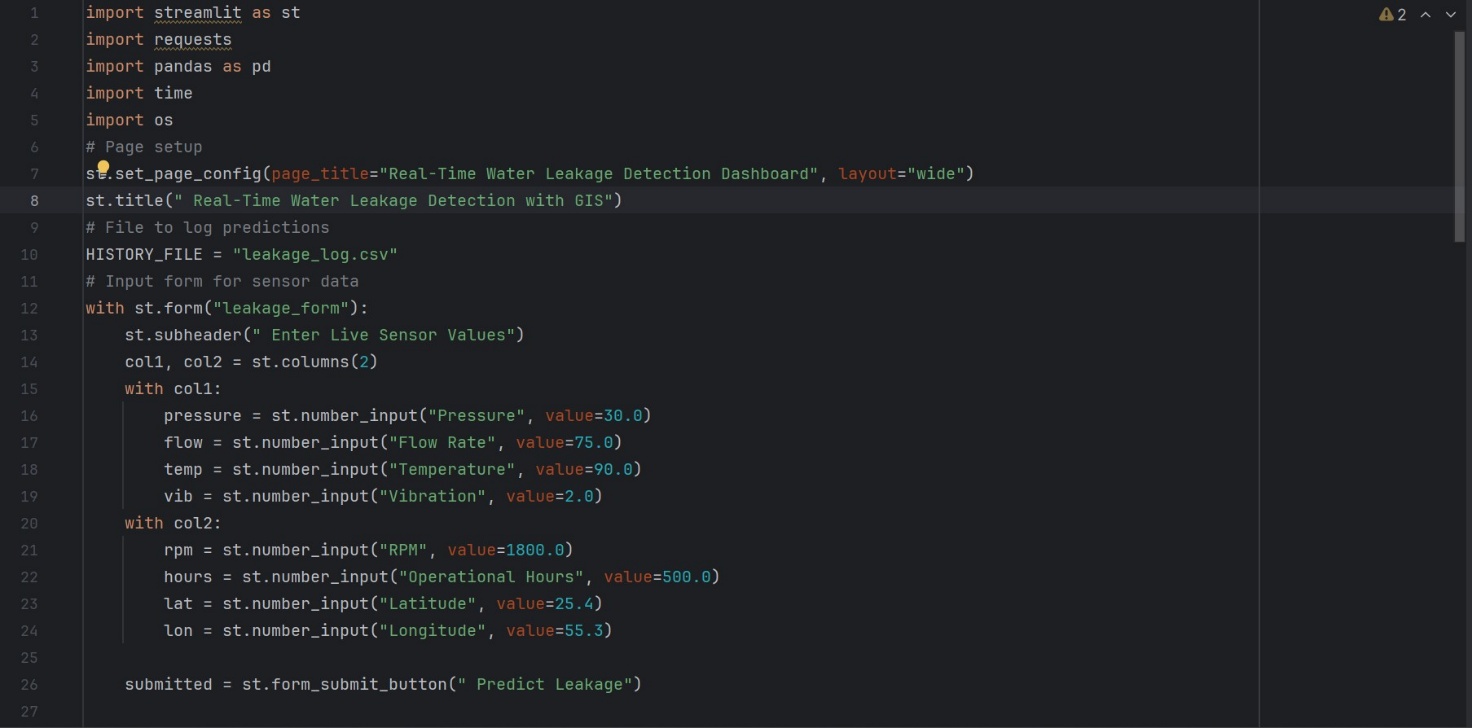
|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Unit** |
| pressure | Water pressure | psi |
| flow\_rate | Flow rate | m³/s |
| temperature | Sensor temperature | °C |
| vibration | Vibration reading | m/s² |
| motor\_rpm | Motor RPM | RPM |
| operational\_hours | Operational duration | Hours |
| latitude, longitude | Geo-coordinates of sensor | Decimal ° |
| leakage\_detected | Leak status label (0/1) | Binary |

1. **Key Code Snippets:**
2. **Model code Snippets:**

****

1. **App code Snippets:**
2. **Dashboard Code Snippets:**

****

****

****

1. **References**

Project best practices and documentation frameworks