
Software Requirements Specification

for

AI for Sustainable Urban Drainage System for Effective Waterlogging Prediction and Management

Prepared by-

Chavhan

1.Tushar

2.Sakshi Ingale

3.Vrushali Dhage

4.Snehal Godage

Dr.D.Y Patil Institute Of Technology,Pimpri.

Date Created - 30/11/2025

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction.....	1
1.1 Purpose	1
1.2 Document Conventions.....	1
1.3 Intended Audience and Reading Suggestions.....	1
1.4 Product Scope	1
1.5 References.....	2
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	3
2.4 Operating Environment.....	3
2.5 Design and Implementation Constraints.....	3
2.6 User Documentation	4
2.7 Assumptions and Dependencies	4
3. External Interface Requirements	4
3.1 User Interfaces	4
3.2 Hardware Interfaces	4
3.3 Software Interfaces	5
3.4 Communications Interfaces	5
4. System Features.....	6
4.1 System Feature 1	6
4.2 System Feature 2 (and so on).....	7
5. Other Nonfunctional Requirements	8
5.1 Performance Requirements	8
5.2 Safety Requirements	8
5.3 Security Requirements	8
5.4 Software Quality Attributes.....	8
5.5 Business Rules	9
6. Other Requirements	9
Appendix A: Glossary.....	10
Appendix B: Analysis Models.....	10
Appendix C: To Be Determined List.....	10

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define all functional, non-functional, and system interface requirements for the project titled “AI for Sustainable Urban Drainage System for Effective Waterlogging Prediction and Management.”

This project uses Artificial Intelligence, IoT sensor networks, and Geographic Information System (GIS) technologies to predict, monitor, and manage urban flooding and waterlogging problems efficiently.

1.2 Document Conventions

This SRS follows the IEEE standard format.

- FR- denotes *Functional Requirements*.
- NFR- denotes *Non-Functional Requirements*.
- Priorities are classified as High, Medium, or Low.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Developers - to understand implementation requirements.
- Project Managers - to plan development and deployment.
- Researchers - for model training and validation details.
- End-users/City Authorities -for understanding system capabilities and limitations.

1.4 Product Scope

The proposed system focuses on the predictive and sustainable management of urban drainage networks using AI-based decision-making techniques. By integrating meteorological data, real-time IoT water-level sensor readings, and GIS-based drainage mapping, the system can effectively anticipate potential flooding events and optimize

water discharge operations. The primary objective is to predict waterlogging situations in real time, enabling authorities to make data-driven decisions for efficient drainage management. Additionally, the system aims to provide early warning alerts to both municipal operators and the public, helping to minimize infrastructure damage, reduce flood risks, and promote long-term sustainability in urban environments.

1.5 References

- “AI-Driven Sustainable Urban Drainage System for Effective Waterlogging Prediction and Management,” Review Paper, 2024.
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.
- Smart City Data Exchange Guidelines (Gov. of India).

2. Overall Description

2.1 Product Perspective

This project is a smart drainage management platform that integrates multiple data sources into a unified AI model. The system architecture consists of:

- Data Acquisition Layer: Collects weather, rainfall, and IoT sensor data.
- Processing Layer: Uses AI models such as LSTM, GRU, and BiTCN to forecast flooding.
- Control and Decision Layer: Generates alerts and assists in automated drainage control.
- Visualization Layer: Displays GIS-based interactive dashboards for stakeholders.

2.2 Product Functions

- Collect meteorological and hydrological data in real time.
- Predict flooding intensity and affected zones.
- Generate early warning alerts and risk levels.

- Support automatic pump control and valve actuation.
- Provide visual dashboards and historical analytics.

2.3 User Classes and Characteristics

User Class	Role	Technical Skill	Access Level
Municipal Operator	Monitors dashboard, manages alerts	Moderate	Full
Maintenance Staff	Receives pump control alerts	Basic	Partial
System Administrator	Configures sensors, models, and users	High	Full
Public User	Receives general alerts	Low	Limited

2.4 Operating Environment

- Hardware: IoT-based sensors, pumps, actuators, and edge gateways.
- Software: Python, TensorFlow, PostgreSQL, PostGIS, Web UI.
- Operating System: Linux/Ubuntu (Server), Android/iOS (App).
- Network: LoRaWAN, 4G/LTE, or Wi-Fi.

2.5 Design and Implementation Constraints

- Edge devices have limited processing power and memory, restricting complex AI computations locally.
- The system depends on third-party weather APIs, and any downtime or changes may affect data availability.
- Must comply with municipal and environmental data-sharing regulations to ensure data privacy and security.
- Network connectivity issues such as weak signals or latency can delay data transmission and alerts.
- Sensors must be installed in varied terrain and harsh weather conditions, requiring durable hardware.
- The system must integrate smoothly with existing SCADA and drainage infrastructure without disrupting operations.

- Budget and resource limitations may restrict large-scale deployment and frequent model retraining.

2.6 User Documentation

- Installation manual
- User and operator guide
- Maintenance manual
- API and data integration guide

2.7 Assumptions and Dependencies

- IoT sensors are correctly installed and calibrated.
- Weather APIs remain available and reliable.
- City GIS drainage data is up-to-date.

3. External Interface Requirements

3.1 User Interfaces

- Dashboard Interface: Interactive map with real-time sensor readings and alert levels.
- Mobile Interface: Push notifications for flood alerts.
- Admin Console: Configuration of thresholds, users, and model retraining.

3.2 Hardware Interfaces

- Water-level and rainfall sensors via LoRaWAN/NB-IoT.
- Actuators and pumps controlled via SCADA/PLC.

- Edge devices connected to cloud through MQTT protocol.

3.3 Software Interfaces

The system interacts with multiple software components and services to ensure seamless data flow, analysis, and visualization:

- APIs:
 - Weather Data API: The system integrates with external APIs such as OpenWeather or India Meteorological Department (IMD) to collect real-time and forecasted weather parameters including rainfall intensity, temperature, humidity, and wind speed.
 - GIS Data API: The system accesses city-level spatial servers to retrieve geospatial information like drainage network maps, catchment areas, and elevation data for accurate flood modeling.
 - RESTful APIs: Internal REST APIs are used for model inference, data exchange between the edge and cloud layers, and dashboard updates. These APIs enable communication between modules for prediction, alert generation, and visualization.

- Database:

The system uses PostgreSQL integrated with PostGIS for storing both non-spatial (sensor readings, user data, logs) and spatial data (drainage networks, waterlogging zones, and coordinates). This spatially enabled database supports efficient geospatial queries, mapping, and analysis required for AI-based flood prediction and decision-making.

3.4 Communications Interfaces

- The system uses HTTPS for secure web communication between the server, dashboard, and APIs.
- MQTT (Message Queuing Telemetry Transport) protocol enables lightweight, real-time data transfer between IoT sensors, edge gateways, and the cloud.

- LoRaWAN (Long Range Wide Area Network) is used for low-power, long-range communication between field sensors and base stations in urban areas.
- All communication channels are secured using TLS 1.2 or higher to ensure data confidentiality and integrity.
- The message transmission rate between sensors and the cloud is configurable, with a default frequency of every 5 minutes for regular updates.
- In case of network failure, data buffering and retransmission mechanisms are implemented to prevent data loss.
- Communication interfaces are designed to be scalable, supporting thousands of IoT devices across multiple drainage zones.

4. System Features

4.1 System Feature 1: Data Collection and Integration

4.1.1 Description and Priority

Collect meteorological, hydrological, and sensor data for analysis.

Priority: High

4.1.2 Stimulus/Response Sequences

- Sensors record rainfall and water levels.
- Edge gateway transmits to cloud.
- Data validated, cleaned, and stored.

4.1.3 Functional Requirements

- REQ-01: The system shall collect IoT and weather data in real time.
- REQ-02: The system shall perform data validation and noise filtering.
- REQ-03: The system shall integrate GIS-based drainage map data.

4.2 System Feature 2: Prediction and Analysis

4.2.1 Description and Priority

Perform AI-based prediction using temporal-spatial modeling.

Priority: High

4.2.2 Stimulus/Response Sequence

1. Data passed to AI models periodically.
2. Predictions generated for each node.
3. Confidence levels stored and visualized.

4.2.3 Functional Requirements

- REQ -04: System shall use LSTM/GRU/BiTCN models to predict flooding.
- REQ -05: Predictions shall include severity levels (low, medium, high).
- REQ -06: System shall calculate model accuracy metrics (R^2 , MAE).

4.3 System Feature 3: Alert and Decision Support

4.3.1 Description and Priority

Generate alerts, visual indicators, and decision support for operators.

Priority: High

4.3.2 Stimulus/Response Sequence

1. Prediction exceeds risk threshold.
2. Alert generated and sent to dashboard and mobile.
3. Optional actuation signal sent to pumps.

4.3.3 Functional Requirements

- REQ -07: System shall send automated notifications via app, SMS, and email.
- REQ -08: Alerts shall include timestamp, location, and severity.
- REQ -09: All alerts shall be logged for auditing.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Edge inference latency ≤ 300 ms.
- Prediction accuracy $\geq 93\%$ ($R^2 \geq 0.93$).
- Alert generation within 60 seconds of data input.
- Hit rate $\geq 90\%$.

5.2 Safety Requirements

- All automated actuation must require manual override.
- System must prevent simultaneous contradictory commands.
- Emergency stop function required for safety.

5.3 Security Requirements

- End-to-end encryption for data transfer.
- Role-based user authentication and access control.
- All system changes logged with timestamp.

5.4 Software Quality Attributes

- Availability: 99.5% uptime.
- Scalability: Supports up to 10,000 sensors.
- Maintainability: Modular design for easy updates.
- Usability: Intuitive map-based visualization.

- Reliability: Handles data gaps with fallback mechanisms.

5.5 Business Rules

- Only authorized personnel can trigger control actions.
- Alerts must follow municipal flood response protocol.
- Model retraining requires admin approval.

6. Other Requirements

- Compliance with municipal environmental and data regulations.
- Minimum data retention period: 3 years.
- Multi-language dashboard support (English, Hindi).

Appendix A: Glossary

Term	Description
AI	Artificial Intelligence
GIS	Geographic Information System
IoT	Internet of Things
LSTM / GRU / BiTCN	Deep learning algorithms for spatio-temporal prediction
SCADA	Supervisory Control and Data Acquisition
R ²	Coefficient of Determination (Model Accuracy)
Edge Computing	Local computation at IoT gateways to reduce latency

Appendix B: Analysis Models

- Data Flow: Sensors → Edge → Cloud → Model → Alert → User
- Use Case: Predict Flood → Generate Alert → Actuate Pump → Notify Public
- Entity Relationships: Sensor – Node – Event – Alert – Action

Appendix C: To Be Determined List

ID	Description
TBD-1	Final IoT hardware specifications
TBD-2	City GIS integration APIs
TBD-3	Exact alert thresholds for each rainfall category
TBD-4	Selection of AI model ensemble configuration