## The next-generation GIS-powered water control system for Munbon Irrigation Project combining real-time monitoring and advanced analytics

Thailand’s agricultural productivity and water security depend on modernizing irrigation infrastructure with intelligent digital systems. This comprehensive specification details the architecture, integration requirements, and implementation approach for the Automatic Water Control System for the Munbon Irrigation Project. The system will leverage GE iFix SCADA integration, real-time sensor networks, spatial data management, AI-powered optimization, and Thai-optimized user interfaces to create a modern web GIS application compliant with government standards.

## System Architecture

The Automatic Water Control System requires a multi-tiered architecture optimized for Thailand’s infrastructure conditions and regulatory environment while enabling sophisticated water management capabilities.

### Core architecture layers

The system architecture consists of three primary layers:

* 1. **Field layer**: Network of sensors and control devices deployed throughout the irrigation infrastructure, including AOS sensors, water level monitors, moisture content sensors connected via LoRA gateways, and controllable pumps/gates integrated with GE iFix SCADA
  2. **Edge layer**: Gateway devices that aggregate sensor data, perform initial validation, and enable local control even during connectivity disruptions. In rural areas, edge processing reduces bandwidth requirements while maintaining system functionality
  3. **Application layer**: Central servers hosting the web GIS application, databases, analytics modules, AI components, and user interfaces. This layer integrates with Thailand’s government systems and implements security requirements for critical infrastructure

![System Architecture Diagram]

### Data flow architecture

Data flows through the system in multiple pathways:

* **Sensor → Gateway → Server**: Environmental readings from field sensors pass through LoRA gateways to centralized servers
* **SCADA → Server**: GE iFix SCADA system provides pump status and control capabilities through integrated APIs
* **Server → UI**: Processed data is delivered to user interfaces via WebSockets for real-time updates
* **Analytics → Control**: Water demand calculations drive automated control recommendations through algorithmic processing
* **AI Pipeline**: Spatial-temporal data feeds AI models for predictive analytics and optimization

### Integration with existing systems

The system will integrate with several existing components:

* **GE iFix SCADA**: Integration using WebSpace with iFrame technology and OPC UA for data exchange
* **Royal Irrigation Department databases**: API integrations with existing water management systems
* **Thailand Meteorological Department**: Weather data feeds for contextual analysis
* **Government SSO**: Authentication through Thai government identity systems
* **AquaCrop Model**: Integration for crop water requirement predictions

# Detailed Model Architecture for AI-Based Water Distribution Control

### **Integration Flow**

[Spatial-Temporal Encoder] → [Graph Neural Network] → [Multi-Objective Optimizer] → [Real-time Controller]

↑ ↑ ↑ ↓

[Raw Data Sources] [Network State] [System Constraints] [Physical Actions]

↑ ↑ ↑ ↓

[Feedback Loop] ←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←← [SCADA/Sensors]

## 1. Spatial-Temporal Encoder

### **1.1. System Architecture**

* **Position**: Data preprocessing and feature extraction layer
* **Dependencies**: Raw data inputs (GIS, weather, crop models, historical data)
* **Outputs**: Enriched spatial-temporal features for downstream models
* **Integration**: Feeds processed features to GNN and provides demand forecasts to optimizer

### **1.2. Functional Requirements**

**1.2.1. Data Ingestion Requirements:**

* Handle multiple data formats: Shapefiles (.shp), raster (.tif), time series (.csv), weather APIs
* Process data at different spatial resolutions (1m to 10km) and temporal frequencies (1min to seasonal)
* Support real-time data streams with latency <5 minutes for weather updates
* Automated data quality checks and outlier detection

**1.2.2. Processing Requirements:**

* Scale to handle 10,000+ individual farm polygons simultaneously
* Process multi-band raster data (NDVI, soil moisture, DEM) efficiently
* Handle missing data through interpolation and imputation methods
* Support multiple coordinate systems and automatic reprojection

**1.2.3. Feature Engineering Requirements:**

* Generate spatial features: farm boundaries, crop type distributions, elevation gradients
* Extract temporal patterns: seasonal cycles, growth stages, weather trends
* Create derived features: crop water stress indices, effective rainfall, irrigation efficiency
* Maintain feature versioning for model reproducibility

**1.2.4. Output Specifications:**

* Standardized tensor format compatible with PyTorch/TensorFlow
* Spatial resolution: 100m x 100m grid cells minimum
* Temporal resolution: Hourly predictions with daily and seasonal context
* Uncertainty quantification for each spatial-temporal prediction

### **1.3. Spatial Processing Component**

* **Input Layer**: Processes GIS shapefiles converted to raster grids representing farm boundaries, crop types, soil properties, and elevation
* **Convolutional Layers**: Multi-scale feature extraction using different kernel sizes (3x3, 5x5, 7x7) to capture local patterns (individual farms) and regional patterns (irrigation districts)
* **Attention Mechanisms**: Self-attention to identify relationships between spatially separated but hydrologically connected areas
* **Hierarchy Processing**: Progressive downsampling to create multi-resolution representations from field-level (10m) to district-level (1km)

### **1.4. Temporal Processing Component**

* **Time Series Embedding**: Converts raw time series (weather, historical demand, crop phenology) into dense vector representations
* **Recurrent Layers**: LSTM/GRU networks to capture seasonal patterns, growth cycles, and long-term trends
* **Temporal Attention**: Focus on critical time periods (planting, flowering, harvest) that drive water demand
* **Multi-Scale Temporal**: Process data at different time resolutions (hourly weather, daily demand, seasonal cycles)

### **1.5. Fusion Mechanism**

* **Cross-Modal Attention**: Links spatial features with temporal patterns to understand how specific locations behave over time
* **Feature Pyramid Networks**: Combine multi-scale spatial and temporal features
* **Dynamic Weighting**: Learns importance of different data sources based on current conditions and uncertainty levels

### **1.6. Output Representation**

* High-dimensional embeddings that encode spatial-temporal demand patterns
* **Uncertainty** estimates for each spatial location and time step
* **Feature** maps highlighting critical demand hotspots and temporal transitions

## 2. Graph Neural Network

### **2.1. System Architecture**

* **Position**: Network topology processing and state representation
* **Dependencies**: Spatial-temporal features, real-time SCADA data, network topology
* **Outputs**: Node and edge embeddings representing hydraulic network state
* **Integration**: Uses encoder features, provides network state to optimizer, receives commands from controller

### **2.2. Functional Requirements**

**2.2.1. Graph Construction Requirements:**

* Dynamic graph updates as network topology changes (gates open/close)
* Support for 1,000+ nodes and 5,000+ edges minimum
* Real-time node attribute updates (<1 minute latency)
* Handling of heterogeneous node types (gates, pumps, tanks, demand points)

**2.2.2. Message Passing Requirements:**

* Bidirectional information flow (upstream supply signals, downstream demand signals)
* Support for different edge types (gravity flow, pumped flow, storage)
* Configurable message passing depth (3-5 layers typical)
* Attention mechanisms that respect hydraulic connectivity

**2.2.3. State Representation Requirements:**

* Node embeddings: 64-128 dimensional vectors per node
* Edge embeddings: Capture flow capacity, current utilization, hydraulic resistance
* Global network state: Overall system pressure, storage levels, operational mode
* Temporal consistency across multiple time steps

**2.2.4. Real-time Processing Requirements:**

* Update graph state within 30 seconds of receiving new sensor data
* Support incremental updates (only changed nodes/edges)
* Maintain sliding window of historical states (24-48 hours)
* Graceful handling of communication failures or sensor malfunctions

### **2.3. Graph Construction**

* **Nodes**: Represent physical infrastructure (gates, pumps, storage tanks, demand points)
* **Edges**: Capture hydraulic connections with attributes (capacity, distance, elevation difference, friction losses)
* **Dynamic Graph**: Structure can change based on gate positions and operational modes

### **2.4. Node Feature Engineering**

* **Static Features**: Physical properties (max capacity, elevation, type)
* **Dynamic Features**: Real-time measurements (flow rates, water levels, gate positions)
* **Contextual Features**: Downstream demand, upstream supply, hydraulic gradients
* **Historical Features**: Recent performance metrics, maintenance status, reliability indicators

### **2.5. Edge Feature Engineering**

* **Hydraulic Properties**: Pipe diameter, roughness, length, elevation profile
* **Flow Characteristics: Current flow rate, direction, velocity, pressure drop**
* **Operational Status**: Active/inactive, maintenance mode, flow restrictions
* **Capacity Utilization**: Current flow as percentage of maximum capacity

### **2.6. Message Passing Architecture**

* **Forward Propagation**: Information flows from source to demand points following hydraulic gradients
* **Backward Propagation**: Demand signals propagate upstream to influence supply decisions
* **Lateral Communication**: Adjacent nodes share information about local conditions and constraints
* **Multi-Layer Processing**: Each layer represents different time scales (immediate hydraulic response vs. longer-term planning)

### **2.7. Attention Mechanisms**

* **Flow-Aware Attention**: Weights based on hydraulic connectivity strength
* **Demand-Driven Attention**: Focus on paths leading to high-priority demand points
* **Constraint-Aware Attention: Emphasize bottlenecks and capacity limitations**
* **Temporal Attention**: Different attention patterns for different planning horizons

### **2.8. Output Generation**

* **Node Embeddings**: Rich representations capturing local hydraulic state and connectivity
* **Edge Embeddings**: Information about flow potential and constraints between nodes
* **Global Graph Representation**: Overall network state and health metrics

## 3. Multi-Objective Optimizer

### **3.1. System Architecture**

* **Position**: Central decision-making engine
* **Dependencies**: GNN state representations, demand forecasts, system constraints
* **Outputs**: Optimal control commands (gate positions, pump settings)
* **Integration**: Receives processed data from encoder and GNN, sends commands to controller

### **3.2. Functional Requirements**

**3.2.1. Optimization Problem Specification:**

* Handle 500+ decision variables simultaneously (gate positions, pump speeds)
* Support 5-10 competing objectives with user-defined weights
* Process 100+ operational constraints (flow limits, storage bounds, pressure requirements)
* Generate solutions within 5-15 minutes for tactical planning

**3.2.2. Solution Quality Requirements:**

* **Pareto-**optimal solutions for multi-objective scenarios
* Feasibility guaranteed under normal operating conditions
* Robustness to 10-20% uncertainty in demand forecasts
* Solution stability (minimal changes between optimization cycles)

**3.2.3. Scalability Requirements:**

* Linear scaling with network size up to 10,000 control points
* Parallel processing support for multiple scenarios
* Efficient warm-starting from previous solutions
* Memory usage <16GB for large networks

**3.2.4. Integration Requirements:**

* Direct interface with hydraulic simulation software (EPANET, InfoWorks)
* Real-time constraint updating based on current system state
* Explanation capability for operator understanding
* What-if analysis for operator decision support

### **3.3. Objective Function Design**

The optimizer balances multiple competing objectives through a weighted sum or Pareto optimization approach:

**3.3.1. Primary Objectives:**

* **Water Delivery Accuracy**: Minimize deviation between delivered and required water volumes
* **Energy Efficiency**: Reduce pumping costs by maximizing gravity flow utilization
* **Crop Yield Optimization**: Align water delivery timing with critical growth periods
* **System Resilience**: Maintain operational flexibility for unexpected events

**3.3.2. Secondary Objectives:**

* **Infrastructure Wear**: Minimize valve cycling and equipment stress
* **Water Quality**: Consider residence time and mixing effects
* **Equity**: Ensure fair distribution across different user groups
* **Environmental Impact**: Maintain minimum environmental flows

### **3.4. Constraint Handling**

* **Physical Constraints**: Flow capacity limits, storage bounds, hydraulic feasibility
* **Operational Constraints**: Minimum/maximum gate opening rates, pump operation limits
* **Temporal Constraints**: Delivery timing windows, peak demand restrictions
* **Regulatory Constraints**: Water rights, environmental flow requirements

### **3.5. Optimization Algorithm**

* **Hybrid Approach**: Combines gradient-based optimization with evolutionary algorithms
* **Hierarchical Decomposition**: Separates strategic (daily) and tactical (hourly) optimization
* **Constraint Preprocessing**: Uses physics-based models to eliminate infeasible solutions
* **Warm Starting**: Initializes optimization with solutions from previous time steps

### **3.6. Uncertainty Quantification**

* **Probabilistic Constraints**: Handle uncertain demand and supply through chance constraints
* **Robust Optimization**: Generate solutions that perform well under multiple scenarios
* **Risk Assessment**: Quantify probability of constraint violations and system failures
* **Adaptive Margins**: Adjust safety buffers based on uncertainty levels

### **3.7. Multi-Fidelity Optimization**

* **Surrogate Models**: Fast approximations of hydraulic behavior for initial optimization
* **High-Fidelity Validation**: Detailed hydraulic simulation for final solution verification
* **Active Learning**: Iteratively improve surrogate model accuracy in promising regions
* **Parallel Evaluation**: Simultaneous optimization across multiple scenarios

## 4. Real-time Controller

### **4.1 System Architecture**

* **Position**: Execution and monitoring layer
* **Dependencies**: Optimization commands, real-time feedback from field devices
* **Outputs**: Physical actuator commands, system status reports
* **Integration**: Executes optimizer commands, provides feedback to all upstream components

### **4.2 Functional Requirements**

**4.2.1. Command Execution Requirements:**

* Translate abstract optimization variables to specific device commands
* Support multiple communication protocols (Modbus, OPC-UA, DNP3)
* Command rate limiting to prevent equipment damage (max 10% change per minute)
* Coordinated sequencing of multiple devices to prevent hydraulic transients

**4.2.2. Real-time Monitoring Requirements:**

* Update cycle: 5-second intervals for critical parameters
* Support 10,000+ monitoring points simultaneously
* Automated alarm generation and escalation
* Historical data logging with 1-second resolution for 7 days

**4.2.3. Safety and Reliability Requirements:**

* Hardware interlocks that cannot be overridden by software
* Automatic fallback to manual control within 3 seconds of detected failure
* Redundant communication paths for critical control points
* Cybersecurity compliance (IEC 62443 standards)

**4.2.4. Performance Requirements:**

* Control loop latency <5 seconds from sensor reading to actuator command
* 99.9% uptime requirement with graceful degradation
* Support for planned maintenance without system shutdown
* Automatic self-diagnostics and health monitoring

## 5. Inter-Model Communication

### **5.1. Data Flow Specifications**

**5.1.1. Encoder → GNN:**

* Format: Structured tensors with spatial-temporal demand predictions
* Frequency: Every 15 minutes during operational hours
* Content: Farm-level water requirements, uncertainty bounds, spatial correlations

**5.1.2. GNN → Optimizer:**

* Format: Graph embeddings with node/edge attributes
* Frequency: Every optimization cycle (15-60 minutes)
* Content: Network hydraulic state, capacity utilization, bottleneck identification

**5.1.3. Optimizer → Controller:**

* Format: Command vectors with execution timestamps
* Frequency: Every 5-15 minutes
* Content: Target gate positions, pump speeds, priority levels, execution constraints

**5.1.4. Controller → All (Feedback):**

* Format: Real-time measurements and status updates
* Frequency: Continuous (5-second updates)
* Content: Actual device positions, flow measurements, alarm states, communication status

### **5.2. System Synchronization**

* **Master Clock**: All components synchronized to GPS time
* **Data Versioning**: Each component tracks data lineage and timestamps
* **Consistency Checks**: Automated validation of data flow between components
* **Error Propagation**: Failed components notify downstream systems immediately

### **5.3. Failure Modes and Recovery**

* **Component Isolation**: Each component can operate in degraded mode if others fail
* **State Persistence**: Critical system state saved to non-volatile storage
* **Automatic Recovery**: Components attempt restart with last known good state
* **Manual Override**: Operators can bypass any component and take direct control

This integrated architecture ensures that all four models work as a cohesive system while maintaining modularity for testing, maintenance, and upgrades. Each component has clearly defined interfaces and responsibilities, enabling reliable operation of the complete AI-driven water distribution control system.

# Model Predictive Control (MPC) with AI - Detailed Component Analysis

## Overview of MPC-AI Integration

The MPC with AI system combines traditional model predictive control algorithms with AI components to enhance prediction accuracy and system adaptability. Let me break down each component:

## 1.1. Neural Network Components (AI Models - Require Training)

### 1.1.1. LSTM for Demand Forecasting

**What it is:** A deep learning model that predicts water demand patterns using historical data and external factors.

#### **Architecture Details:**

* **Input Layer:** Multi-variate time series (past 7-30 days)
  + Historical water demand (hourly/daily)
  + Weather variables (temperature, humidity, solar radiation)
  + Crop growth stages (from phenology models)
  + Irrigation efficiency metrics
  + Day of week/month seasonality encoding
* **LSTM Layers:**
  + 2-3 stacked LSTM layers (128-256 hidden units each)
  + Dropout layers (0.2-0.3) between LSTM layers for regularization
  + Bidirectional LSTMs for capturing both past and future dependencies
* **Dense Layers:**
  + Fully connected layers with ReLU activation
  + Output layer with linear activation for regression
* **Output:** Water demand forecast for next 24-48 hours at hourly resolution

#### **Training Data Setup:**

* **Data Collection Sources:**
  + **SCADA Historical Logs:** 2-5 years of hourly flow measurements
  + **Weather Station Data:** Historical meteorological records
  + **Irrigation Records:** Farmer-reported irrigation events
  + **Crop Calendar Data:** Planting, growth stage, harvest dates
  + **Water Rights Database:** Allocated vs. actual usage
* **Data Preprocessing:**
  + **Time Series Normalization:** StandardScaler or MinMaxScaler
  + **Missing Value Handling:** Linear interpolation or forward-fill for short gaps
  + Feature Engineering:
  + Rolling averages (3-day, 7-day, 30-day)
  + Lag features (demand 1-7 days ago)
  + Seasonal decomposition components
  + Weather change indicators (temperature gradients, rainfall onset)

#### **Training Process:**

* **Dataset Split:** 70% train, 15% validation, 15% test
* **Sequence Length:** 168 timesteps (1 week) for input
* **Prediction Horizon:** 24-48 timesteps (1-2 days) for output
* **Loss Function:** Mean Squared Error with L2 regularization
* **Optimization:** Adam optimizer with learning rate scheduling
* **Training Duration:** 100-500 epochs with early stopping

### 1.1.2. CNN for Spatial Demand Distribution

**What it is:** A convolutional neural network that processes spatial data to predict water demand distribution across the irrigation area.

#### **Architecture Details:**

* **Input Channels:** Multi-band raster data
  + Land use classification maps
  + Soil property maps (texture, moisture retention)
  + Topographic data (elevation, slope, aspect)
  + Vegetation indices (NDVI, SAVI, LAI)
  + Irrigation infrastructure density
* **Convolutional Layers:**
  + Multiple conv blocks with increasing filters (32, 64, 128, 256)
  + Kernel sizes: 3x3, 5x5 for different spatial scales
  + Batch normalization and ReLU activation
  + Max pooling layers for dimensionality reduction
* **Attention Mechanisms:**
  + Spatial attention to focus on high-demand areas
  + Channel attention to weight different input features
* **Output Layers:**
  + Transpose convolution for upsampling
  + Final conv layer with sigmoid activation
  + Output: Spatial demand map (normalized 0-1)

#### **Training Data Setup:**

* **Data Collection Sources:**
  + **Satellite Imagery:** Landsat-8, Sentinel-2 for vegetation indices
  + **GIS Databases:** Land use, soil surveys, irrigation infrastructure
  + **Digital Elevation Models:** SRTM or LiDAR-derived elevation data
  + **Field Surveys:** Ground-truth demand measurements at specific locations
  + **Smart Meter Data:** Sub-district water usage measurements
* **Data Preprocessing:**
  + **Spatial Registration:** Align all rasters to common coordinate system
  + **Resolution Standardization:** Resample to consistent pixel size (30m-100m)
  + **Temporal Alignment:** Match satellite imagery dates with demand measurements
  + **Data Augmentation:** Rotation, flipping, noise addition for training
  + **Normalization:** Per-band z-score normalization

#### **Training Process:**

* **Patched-based Training:** Extract 256x256 pixel patches from large images
* **Ground Truth:** Interpolated demand surfaces from point measurements
* **Loss Function:** Combination of MSE and perceptual loss
* **Validation:** Spatial cross-validation (geographically separated test areas)
* **Training Duration:** 200-1000 epochs depending on dataset size

### 1.1.3. Physics-Informed Neural Networks (PINNs) for Hydraulic Modeling

**What it is:** Neural networks that incorporate physical laws (hydraulic equations) into the learning process to model water flow behavior.

#### **Architecture Details:**

* **Input Variables:**
  + Spatial coordinates (x, y, z)
  + **Time** coordinate (t)
  + Boundary conditions (inflow rates, gate positions)
  + Network topology parameters
* **Neural Network:**
  + **Deep** feedforward network (5-10 hidden layers)
  + **50-**200 neurons per layer
  + Hyperbolic tangent activation functions
  + Residual connections for deep networks
* **Physics Constraints:**
  + Continuity equation: ∇·v = 0
  + Momentum equation: ∂v/∂t + v·∇v = -∇p/ρ + ν∇²v
  + **Energy** equation for hydraulic head
  + Boundary conditions at gates, pumps, and outlets
* **Loss Function:**
  + Data loss: Comparison with observed measurements
  + Physics loss: Residual of differential equations
  + Boundary loss: Satisfaction of boundary conditions
  + Total loss: Weighted combination of all components

#### **Training Data Setup:**

* **Data Collection Sources:**
  + **CFD Simulations:** High-fidelity computational fluid dynamics results
  + **Laboratory Experiments:** Controlled hydraulic experiments
  + **Field Measurements:** Flow rates, pressure heads, velocities
  + **Historical SCADA Data:** Long-term operational records
  + **Hydraulic Model Results:** Calibrated EPANET/InfoWorks outputs
* **Data Preprocessing:**
  + **Dimensionless Variables:** Scale inputs to [-1, 1] for numerical stability
  + **Spatial Sampling:** Uniform and adaptive sampling of domain
  + **Temporal Grid:** Fine time discretization for transient phenomena
  + **Noise Addition:** Realistic noise levels to match sensor accuracy

#### **Training Process:**

* **Multi-objective Optimization:** Balance data fitting and physics compliance
* **Adaptive Weighting:** Dynamically adjust loss component weights
* **Transfer Learning:** Pre-train on simple geometries, fine-tune on complex networks
* **Validation:** Compare against independent experimental data

## 1.2. Non-AI Components (Data Sources - No Training Required)

### 1.2.1. AquaCrop Model Outputs

**What it is:** FAO's crop growth simulation model that calculates crop water requirements.

#### **Setup and Data Requirements:**

* **Software Installation:**
  + Download AquaCrop (standalone) or AquaData (plugin for GIS)
  + Install Python wrapper (aquacrop package) for automated execution
  + **Set** up batch processing capabilities for multiple fields
* **Input Data Collection:**
  + Climate **Data:**
    - * Daily temperature (min/max)
      * Precipitation
      * Reference evapotranspiration (ET₀)
      * Solar radiation
      * Wind speed and humidity
  + **Soil Data:** 
    - * Soil texture classification
      * Field capacity and wilting point
      * Saturated hydraulic conductivity
      * Soil depth and layering
  + **Crop Data:** 
    - * Crop type and variety
      * Planting dates
      * Growth cycle length
      * Crop coefficients (Kc curves)
      * Canopy cover development
  + **Management Data:** 
    - * Irrigation method and efficiency
      * Fertilizer application
      * Field capacity constraints

#### **Configuration Process:**

* **Spatial Discretization:** Divide irrigation area into homogeneous units
* **Crop Calendar Setup:** Define growing seasons and rotations
* **Irrigation Scenarios:** Configure different water application strategies
* **Model Calibration:** Use local yield and water use data for validation

#### **Output Data:**

* Daily crop miwater requirements (mm/day)
* Irrigation ting recommendations
* Yield predictions under water stress
* Water productivity indicators

### 1.2.2. Weather Forecasts

**What it is:** Meteorological predictions for temperature, rainfall, and evapotranspiration.

#### **Setup and Data Requirements:**

* **Data Source APIs:**
  + **Public Sources:**
    - * OpenWeatherMap API
      * NOAA/NWS API
      * European Centre for Medium-Range Weather Forecasts (ECMWF)
      * Local meteorological services
  + **Commercial Sources:**
    - * AccuWeather API
      * Weather Underground
      * IBM Weather API
      * Agricultural weather services
* **Data Collection Setup:**
  + **API Integration:**
    - * Register for API keys and rate limits
      * Set up automated data retrieval scripts
      * Implement error handling and retry logic
      * Configure data storage and archiving
  + **Real-time Processing:**
    - * Hourly data updates for short-term forecasts
      * Six-hourly updates for medium-range forecasts
      * Daily updates for seasonal outlooks

#### **Data Quality Control:**

* **Bias Correction:** Adjust forecasts based on local observations
* **Ensemble Processing:** Combine multiple forecast models
* **Uncertainty Quantification:** Confidence intervals for predictions
* **Validation:** Compare forecasts with actual weather

#### **Output Data:**

* **Temperature** forecasts (1-48 hours)
* **Precipitation** probability and intensity
* **Reference** evapotranspiration estimates
* Wind speed and solar radiation
* **Extreme** weather alerts

### 1.2.3. Historical Demand Patterns

**What it is:** Past water usage records used for pattern recognition and baseline establishment.

#### **Setup and Data Requirements:**

* **Data Sources:**
  + **SCADA Systems:**
    - * Flow meter readings
      * Gate position logs
      * Pump operation records
      * System pressure measurements
  + **Billing Records:**
    - * Monthly/quarterly water usage
      * Peak demand periods
      * Customer consumption patterns
  + **Agricultural Statistics:**
    - * Crop area and yield data
      * Irrigation practice surveys
      * Water rights allocation records
* **Data Management:**
  + **Database Setup:**
    - * Time-series database (InfluxDB, TimescaleDB)
      * Data cleaning and validation procedures
      * Automated data ingestion pipelines
  + **Quality Assurance:**
    - * Outlier detection and removal
      * Gap filling for missing data
      * Sensor calibration corrections
      * Cross-validation between multiple sources

#### **Pattern Analysis:**

* **Temporal Patterns:**
  + Daily usage cycles
  + Weekly patterns (weekend effects)
  + Seasonal variations
  + **Annual** trends
* **Spatial Patterns:**
  + Regional demand differences
  + Infrastructure-dependent patterns
  + Crop-type specific usage

#### **Output Data:**

* **Statistical baselines for demand**
* **Seasonal adjustment factors**
* **Peak demand predictions**
* **Usage efficiency metrics**

## MPC with AI Integration in Control Architecture

[Spatial-Temporal Encoder] → [Graph Neural Network] → [Multi-Objective Optimizer] → [Real-time Controller]

↑ ↑ ↑ ↓

[Raw Data Sources] [Network State] [System Constraints] [Physical Actions]

↑ ↑ ↑ ↓

[Feedback Loop] ←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←← [SCADA/Sensors]

| |

| [MPC with AI] |

↓ ↓

[Prediction Horizon] [Control Action Sequence]

## 2.1 Position in the Architecture

The MPC with AI system operates as an **integrated component** that:

### 2.1.1. Sits **between the Graph Neural Network and Multi-Objective Optimizer**

* + - * 1. **Takes** the network state representation from the GNN
        2. Works alongside the Multi-Objective Optimizer to determine optimal control strategies

### 2.1.2. Provides **a predictive bridge between current state and future actions**

* + - * 1. Simulates **system behavior over the prediction horizon (24-48 hours)**
        2. Evaluates **multiple possible control sequences**

### 2.1.3. Continuously **updates based on feedback loop**

* + - * 1. **Adjusts its internal models based on observed system responses**
        2. Refines **predictions based on actual outcomes**

## 2.2. Core Functions in System Flow

### 2.2.1. Input Processing

* + - * 1. **Receives spatial-temporal features from the encoder (weather patterns, crop water demands)**
        2. **Incorporates real-time network state from the GNN (current flow rates, gate positions)**
        3. **Processes system constraints (physical limits, operational rules, safety parameters)**

### 2.2.2. Prediction Generation

* + - * 1. **Creates a dynamic model of system behavior over the prediction horizon**
        2. **Simulates how different control sequences will affect the network state over time**
        3. **Evaluates the impact of uncertain factors (weather changes, demand variations)**

### 2.2.3. Control Sequence Optimization

* + - * 1. **Works alongside the Multi-Objective Optimizer to generate optimal control trajectories**
        2. **Balances multiple objectives (water delivery accuracy, energy efficiency, etc.)**
        3. **Produces a sequence of control actions over the prediction horizon**

### 2.2.4. Execution Monitoring

* + - * 1. **Compares actual system response with predicted behavior**
        2. **Updates internal models to improve future predictions**
        3. **Adapts to changing conditions through online learning**

## 2.3. Technical Implementation

The MPC with AI system is implemented through three key components:

### 2.3.1. Physics**-Informed Neural Networks (PINNs)**

* + - * 1. **Embed physical laws of hydraulics into neural network structure**
        2. **Combine data-driven learning with first-principles models**
        3. **Ensure predictions obey physical constraints like conservation of mass**

### **2.3.2. Receding Horizon Controller**

* + - * 1. **Continuously shifts the prediction horizon forward as time advances**
        2. **Executes only the first step of each optimized control sequence**
        3. **Re-optimizes control strategy at each time step**

### 2.3.3. State **Estimation and System Identification**

* + - * 1. **Uses Kalman filters or similar techniques to estimate unmeasured states**
        2. **Continuously updates system model parameters based on observed behavior**
        3. **Handles measurement noise and partially observable states**

This integration creates a cohesive system where the MPC with AI serves as the "brain" that translates the GNN's network state understanding and the Spatial-Temporal Encoder's predictive capabilities into optimal control actions, while continually learning and improving from the feedback loop.

# Limited Sensor Infrastructure Issue

Your constraints are common in irrigation systems, particularly in developing regions. Limited sensor coverage presents challenges but doesn't prevent implementing the proposed AI architecture. Here's how to adapt the system for your specific limitations:

## Working with Limited Mobile Field Sensors

## 1.1. Data Collection Strategy for Spatial-Temporal Encoder

### **1.1.1. Strategic Rotation of Mobile Sensors**

* + - * 1. **Sampling Plan**: Develop a statistical sampling plan that rotates mobile sensors across representative fields
        2. **Coverage Optimization**: Use Latin Hypercube or stratified sampling to ensure different soil types, elevations, and crop varieties are covered
        3. **Temporal Density**: Collect dense measurements at reference sites, sparse measurements at others

### **1.1.2. Proxy Measurements and Correlations**

* + - * 1. **Remote Sensing Integration**: Supplement ground sensors with satellite data (Sentinel-2, Landsat)
    - NDVI, NDWI, and land surface temperature as proxies for crop water stress
    - 10m resolution available every 5 days (Sentinel-2)
      * 1. **Environmental Correlations**: Develop transfer functions between weather station data and soil moisture
        2. **Crop Modeling**: Use AquaCrop to simulate moisture conditions between sensor visits

### **1.1.3. Data Augmentation and Interpolation**

* + - * 1. **Spatial Interpolation**: Use geostatistical methods (Kriging, IDW) to estimate values between sensor locations
        2. **Temporal Interpolation**: Apply Gaussian processes to reconstruct continuous time series from sparse measurements
        3. **Physics-Informed Gap Filling**: Use soil water balance equations to estimate moisture between measurements

## 1.2. Handling Limited Flow Measurement at Gates

### 1.2.1. Approaches for Graph Neural Network

* + - * 1. **Soft Sensing and Virtual Metering**
    - **Calibration Campaigns: Conduct periodic intensive measurement campaigns (3-4 gates at a time)**
    - **Empirical Models: Develop gate-specific rating curves relating opening percentage to flow**
    - **Virtual Flow Meters: Create regression models that estimate flow from gate position, upstream/downstream levels**
    - **Transfer Learning: Develop models at measured gates, transfer to similar unmeasured gates**
      * 1. **Flow Estimation from Water Balance**
    - **Mass Conservation: Apply continuity equations across the network**
    - **Volume Tracking: Measure water level changes in bounded canal segments to infer flow**
    - **Water Balance Models: Develop simplified hydraulic models using Manning's equation and gate characteristics**
    - **Uncertainty Quantification: Apply Bayesian methods to provide confidence intervals on flow estimates**
      * 1. **Visual Sensing Alternatives**
    - **Camera-Based Flow Estimation**: Install low-cost cameras to estimate flow from surface velocity patterns
    - **Structure-from-Motion**: Periodic drone surveys to establish channel geometry for flow calculation
    - **Staff Gauge Reading**: Establish simple staff gauges with QR reporting by field staff

## Adapted AI Model Implementation

## 2.1. Modified Spatial-Temporal Encoder

### **2.1.1. Technical Approach:**

* + - * 1. **Multi-resolution Architecture**
    - **Dense resolution at sensor locations, coarser elsewhere**
    - **Satellite data provides complete spatial coverage at lower temporal frequency**
    - **Confidence-weighted predictions based on data density**
      * 1. **Transfer Learning with Physical Constraints**
    - **Pre-train on regions with dense measurements**
    - **Fine-tune for areas with sparse measurements**
    - **Incorporate physical constraints (water balance, infiltration physics)**
      * 1. **Uncertainty-Aware Embeddings**
    - **Represent each spatial location with both prediction and uncertainty**
    - **Higher uncertainty weights for locations far from sensors**
    - **Propagate uncertainty through the system**

### 2.1.2 Data **Requirements:**

* + - * 1. **Minimum Measurement Density: At least 1 soil moisture measurement per 50-100 hectares of similar soil type**
        2. **Temporal Coverage: Rotating measurements ensuring each zone is measured every 1-2 weeks**
        3. **Reference Sites: 5-10 permanent reference sites with continuous monitoring**
        4. **Remote Sensing: Weekly satellite imagery acquisition and processing**
        5. **Weather Data: At least one weather station per irrigation district**

## 2.2. Simplified Graph Neural Network

### **2.2.1. Technical Approach:**

* + - * 1. **Topology-Focused Architecture**
    - **Emphasize known physical connectivity over precise flow values**
    - **Model water distribution as directed graph with capacity constraints**
    - **Learn flow patterns from limited observations and mass conservation**
      * 1. **Multi-scale Message Passing**
    - **Local updates based on nearest measured points**
    - **Global updates based on system-wide water balance**
    - **Time-delay effects modeled explicitly**
      * 1. **Hybrid Knowledge Integration**
    - **Incorporate gate rating curves from calibration campaigns**
    - **Integrate operator knowledge on typical flow patterns**
    - **Use physical hydraulic models to constrain** predictions

### **2.2.2. Data Requirements:**

* + - * 1. **Network Topology: Complete mapping of all canals and gates (already in Structure.xlsx)**
        2. **Gate Calibration: Rating curves for each gate type**
        3. **Periodic Measurements: Flow measurement campaigns (3-4 locations) rotating monthly**
        4. **Historical Operations: Records of gate positions and observed outcomes**
        5. **Canal Geometry: Cross-sectional surveys of main canals**

## 2.3. Robust Multi-Objective Optimizer

### **2.3.1. Technical Approach:**

* + - * 1. **Uncertainty-Aware Formulation**
    - **Explicit modeling of uncertainty in constraints and objectives**
    - **Robust optimization that accounts for estimation errors**
    - **Safety margins scaled with uncertainty level**
      * 1. **Hierarchical Decomposition**
    - **District-level water allocation with aggregate constraints**
    - **Local-level distribution with detailed constraints**
    - **Connected through consistency constraints**
      * 1. **Scenario-Based Optimization**
    - **Generate multiple feasible scenarios**
    - **Evaluate resilience across scenario ensemble**
    - **Select strategies robust to measurement uncertainties**

### **2.3.2. Data Requirements:**

* + - * 1. **Demand Requirements**: Daily water demand estimates by zone
        2. **Operational Constraints**: Gate operating rules, maximum rates of change
        3. **Prioritization Criteria**: Crop value, water rights, equity considerations
        4. **Gate Capacity**: Maximum flow capacity of each gate

## 2.4. Adapted MPC with AI

### **2.4.1. Technical Approach:**

* + - * 1. **Grey-Box Modeling**
    - **Combine first-principles hydraulic models with data-driven components**
    - **Physical equations for mass conservation and hydraulic behavior**
    - **AI models to estimate unmeasured parameters and correct for** model inaccuracies
      * 1. **Reduced-Order Modeling**
    - **Develop simplified hydraulic models that capture essential dynamics**
    - **Focus on dominant water pathways and critical control points**
    - **Use available measurements to calibrate simplified** models
      * 1. **Robust Control Design**
    - **Conservative control actions in areas with high uncertainty**
    - **More aggressive control where measurements are available**
    - **Explore-exploit balance to reduce uncertainty in critical areas**

### **2.4.2. Data Requirements:**

* + - * 1. **Gate Operations: Current positions and control capabilities**
        2. **Canal Levels: Even approximate staff gauge readings**
        3. **Reference Flow Measurements: Periodic flow measurements from campaigns**
        4. **Historical Patterns: Past relationships between inputs and observed outcomes**

## Practical Implementation Example

## 3.1. Water Distribution with Limited Flow Sensors

### **3.1.1. Morning Assessment**

* + - * 1. **Field staff report water levels at key points using mobile app**
        2. **Satellite data from yesterday processed for crop condition**
        3. **AI system integrates measurements with hydraulic model**

### **3.1.2. Demand Forecasting**

* + - * 1. **Spatial-Temporal Encoder predicts water needs by zone**
        2. **Uncertainty maps highlight areas needing verification**
        3. **Field staff directed to highest-uncertainty areas**

### **3.1.3. Distribution Planning**

* + - * 1. **GNN + Optimizer generates gate schedule considering uncertainties**
        2. **Higher margins in areas with limited measurements**
        3. **Conservative approach for gates without recent calibration**

### **3.1.4. Execution and Learning**

* + - * 1. **Operations implemented with close monitoring**
        2. **Staff report outcomes and any discrepancies**
        3. **System updates models based on observed responses**

By focusing on uncertainty quantification, strategic measurement campaigns, and hybrid modeling, you can successfully implement the proposed AI architecture despite sensor limitations. The system will continuously improve as it learns from each operation cycle and additional measurements.

## AI-Enhanced System Components

### AI Components Technology Stack

#### Spatial-Temporal Encoder

**Technology Stack:** - **Deep Learning Framework:** TensorFlow 2.x or PyTorch 1.x for neural network development - **Spatial Processing:** OpenCV, GDAL/OGR for geospatial data handling - **Temporal Processing:** Pandas for time series manipulation, NumPy for numerical operations - **Geospatial Libraries:** GeoPandas, Rasterio for satellite/GIS data processing - **Feature Engineering:** Scikit-learn for preprocessing pipelines - **Deployment:** TensorFlow Serving or TorchServe for model serving - **Containerization:** Docker with CUDA support for GPU acceleration

**Infrastructure Requirements:** - GPU Requirements: NVIDIA V100 or A100 for training, T4 for inference - Memory: 32-64GB RAM for large spatial datasets - Storage: High-speed SSD for raster data caching

#### Graph Neural Network

**Technology Stack:** - **GNN Framework:** PyTorch Geometric (PyG) or Deep Graph Library (DGL) - **Graph Processing:** NetworkX for graph manipulation and analysis - **Base Framework:** PyTorch for GNN implementations - **Optimization:** PyTorch Lightning for experiment management - **Graph Databases:** Neo4j for complex graph queries (optional) - **Message Passing:** Custom implementations using PyG’s MessagePassing class

**Infrastructure Requirements:** - GPU: Required for large graph processing - Memory: 16-32GB RAM depending on network size - Graph Storage: Redis for in-memory graph caching

#### Multi-Objective Optimizer

**Technology Stack:** - **Optimization Libraries:** - CVXPY for convex optimization problems - GEKKO for mixed-integer nonlinear programming - NSGA-II implementation (DEAP library) for multi-objective optimization - **Linear Programming:** Gurobi or CPLEX for commercial solutions, PuLP for open-source - **Constraint Handling:** OR-Tools by Google for complex constraint problems - **Parallel Processing:** Dask for distributed optimization - **Uncertainty Quantification:** SALib for sensitivity analysis

**Infrastructure Requirements:** - CPU: Multi-core processors (16+ cores) for parallel optimization - Memory: 32-64GB RAM for large optimization problems - Solver Licenses: Commercial solver licenses if using Gurobi/CPLEX

#### MPC AI Model

**Technology Stack:** - **MPC Framework:** - CasADi for optimal control and MPC implementation - GEKKO for dynamic optimization - Do-MPC for robust model predictive control - **Physics-Informed Neural Networks:** Custom PyTorch implementations - **Hydraulic Modeling:** Integration with EPANET-Python or WNTR - **Real-time Processing:** Apache Kafka for data streaming - **State Estimation:** Kalman filters using FilterPy

**Infrastructure Requirements:** - Real-time Requirements: Low-latency computing with <1 second response times - Edge Computing: NVIDIA Jetson for field deployment - High Availability: Kubernetes for container orchestration

## SCADA integration specifications

The GE iFix SCADA system forms the control backbone for water pumps and gates, requiring careful integration with the web GIS application.

### Enhanced Integration Options

#### Primary: Hybrid OPC UA + REST Gateway Approach

**Implementation Strategy:** - Develop a middleware REST gateway that communicates with iFix via OPC UA - Use OPC Router or similar tools to create REST endpoints that bridge to OPC UA servers - Web application communicates with this REST gateway for clean separation of concerns

**Advantages:** - Clean separation of concerns - Modern web-friendly REST interface - Can aggregate data from multiple sources - Better security isolation - Platform-independent communication with robust security

#### Secondary: Enhanced WebSpace Integration

**Alternative to iFrame:** - Deploy WebSpace as a standalone application - Use single sign-on (SSO) integration for seamless user experience - Link between applications through deep linking

**Advantages:** - WebSpace supports up to 70 simultaneous clients with full iFix functionality - Better performance than iFrame embedding - Maintains complete iFix feature set

### iFrame integration implementation

Integration with the GE iFix SCADA system will leverage WebSpace, GE’s HTML5 web client solution:

* **WebSpace configuration**: Dedicated WebSpace server (separate from Operations Hub and Configuration Hub)
* **iFrame embedding**: SCADA screens embedded within the main dashboard using HTML5 iFrames
* **JavaScript integration**: Customized JavaScript for loading and controlling SCADA applications
* **User authentication**: Credential passing between GIS application and WebSpace

### API integration methods

The system will use multiple API approaches to exchange data with the GE iFix SCADA system:

* **OPC UA protocol**: Primary communication method for SCADA integration, offering platform independence and robust security
* **REST API integration**: Secondary method using Historian REST APIs for historical data retrieval
* **MQTT integration**: For IoT sensor data integration, with the MQTT Client translating to OPC UA format

**Data exchange strategy:**

Field Device → SCADA System (OPC UA) → Backend Server → Web Dashboard (WebSockets)

### Real-time data access protocols

For efficient real-time data flow:

* **WebSockets**: Primary protocol for pushing real-time updates to client applications
* **OPC UA subscription model**: For efficient change-based data transmission from SCADA
* **MQTT for field devices**: For sensor data from remote locations with constrained bandwidth
* **Data throttling**: Configurable transmission rates based on data criticality (e.g., 5-second updates for critical pump status, 15-minute updates for soil moisture)

### Security implementation

Security measures for SCADA integration include:

* **Network segmentation**: SCADA networks isolated from public internet with properly configured firewalls
* **DMZ architecture**: Web servers in a DMZ between SCADA and users
* **TLS/SSL encryption**: For all web communications
* **Certificate management**: Proper handling of OPC UA security certificates
* **Audit logging**: Comprehensive logging of all SCADA access and control actions

## Sensor network specifications

The sensor network forms the foundation of the real-time monitoring capabilities, requiring careful specification for Thailand’s climate conditions.

### AOS sensor integration

AOS sensors will provide environmental monitoring with the following technical specifications:

* **Sensor specifications**: IP68-rated sensors with 3.3-5.5V DC operating voltage and -20°C to 70°C operating range
* **Transmission frequency**: Configurable from 5 minutes to 6 hours based on operational requirements
* **Data format**: JSON for API communication with timestamp, location, reading type, and value fields
* **SQL integration**: Time-series tables with partitioning by time periods and specialized indexes
* **Error handling**: Range checking, variance analysis, and cross-validation for data quality assurance

### Water level sensor implementation

Water level sensors will continuously monitor reservoir and canal levels:

* **Broadcast protocol**: Primary: LoRaWAN (868/915 MHz) with 10+ km range; Backup: NB-IoT for critical locations
* **Data frequency optimization**:
  + Adaptive sampling rates (1-6 hours during stable conditions, 5-30 minutes during critical periods)
  + Event-based transmission when measurements exceed preset thresholds
  + Contextual adaptation during rainfall or irrigation events
* **Integration methods**: Gateway-based collection with field aggregation to reduce bandwidth usage

### Moisture content sensors with LoRA gateway integration

Soil moisture monitoring will utilize:

* **Sensor types**: Capacitive sensors with ±3% accuracy for volumetric water content
* **LoRA specifications**: 923 MHz frequency band (Thailand allocation) with up to 10+ km range
* **Gateway specifications**: Solar-powered with battery backup, IP67-rated enclosures
* **Cellular network integration**:
  + LoRaWAN to cellular bridging through gateways
  + MQTT and HTTPS for reliable, secure data transmission
  + Store-and-forward capabilities for network outages

### Thailand-specific optimizations

Sensor deployments require specific considerations for Thailand’s environment:

* **Tropical climate adaptation**: IP68 rating for monsoon conditions; UV-resistant enclosures
* **Power management**: Solar with battery backup sized for cloudy monsoon periods
* **Regional adaptations**: Higher gateway density in mountainous northern areas; elevated placement in flood-prone central plains
* **Maintenance schedule**: Preventive maintenance before monsoon season (April)

## GIS architecture

The GIS component requires careful design to support water management workflows and performance in variable connectivity environments.

### Database system recommendation

**Recommendation: Hybrid approach with PostGIS and GeoPackage**

* **Primary database**: PostGIS for the central system, offering superior spatial functions, concurrency, and scalability
* **Field operations**: GeoPackage (Spatial SQLite) for field data collection and offline capabilities
* **Synchronization**: Bidirectional synchronization between field devices and central database

### Vector tiles implementation

Vector tiles will optimize map performance:

* **Tiling scheme**: Indexed tiling structure optimized for varying feature densities across Thailand’s landscape
* **Optimization techniques**:
  + Attribute filtering to reduce tile size
  + Geometric simplification with Douglas-Peucker algorithm
  + Feature culling at smaller scales
  + Coordinate precision reduction
* **Format and delivery**: MVT (Mapbox Vector Tile) format with hierarchical caching system

### Base layer selection

**Recommendation: Multi-tier base layer strategy**

1. **Primary base layer**: Mapbox for optimized performance in bandwidth-constrained environments
2. **Secondary option**: Google Hybrid for detailed satellite imagery during planning phases
3. **Specialized layers**: Thailand government WMTS services for administrative boundaries and official irrigation districts
4. **Offline support**: Pre-cached base map tiles for critical agricultural areas

### Data model for irrigation infrastructure

The GIS data model will include the following feature classes:

* **Water sources**: Reservoirs, rivers, canals, groundwater with capacity and current level attributes
* **Conveyance network**: Canal segments with capacity, flow direction, and maintenance attributes
* **Control structures**: Gates, weirs, pumps, valves with operational status and automation level
* **Irrigation zones**: Agricultural areas with crop type, irrigation method, and water allocation attributes
* **Monitoring points**: Sensor locations with type, calibration status, and transmission method
* **Service areas**: Administrative regions with population and predominant crop attributes

The model will include relationships defining network topology, hierarchical organization, and operational connections between components.

## Comprehensive Technical Architecture

### Frontend Architecture

**Core Framework:** - **Framework**: Next.js 13+ (React with App Router) for server-side rendering and optimal performance - **Language**: TypeScript for type safety and better development experience - **State Management**: Zustand or Redux Toolkit for predictable state management - **Real-time Communication**: Socket.IO client for real-time data updates - **UI Library**: Material-UI (MUI) or Tailwind CSS with Headless UI for consistent design

**GIS and Mapping:** - **Map Library**: Mapbox GL JS (primary) with Leaflet as fallback for broader browser support - **3D Visualization**: Three.js for 3D network visualization and advanced spatial analytics - **Chart Library**: Recharts or Chart.js for data visualization and analytics dashboards - **Spatial Analysis**: Turf.js for client-side geospatial operations and analysis

**Performance Optimization:** - **PWA**: Service Workers for offline functionality in rural areas - **Code Splitting**: Dynamic imports for route-based splitting and faster loading - **Image Optimization**: Next.js Image component with WebP support for bandwidth efficiency - **Caching**: React Query (TanStack Query) for intelligent server state management

**Deployment:** - **Platform**: Vercel or AWS Amplify for optimal Next.js deployment with global CDN - **CDN**: CloudFront for static asset delivery and improved performance - **Monitoring**: Sentry for error tracking, Vercel Analytics for performance monitoring

### Backend Architecture

**API Gateway and Microservices:** - **Framework**: Node.js with Express.js or Fastify for high-performance API development - **Language**: TypeScript for consistency with frontend and better maintainability - **API Documentation**: OpenAPI 3.0 with Swagger UI for comprehensive API documentation - **Authentication**: JWT with refresh tokens, OAuth 2.0 integration for Thai government SSO - **Rate Limiting**: Express-rate-limit or Redis-based solutions for API protection

**Microservices Architecture:**

├── API Gateway (Express.js) - Traffic routing and authentication  
├── Authentication Service (Node.js) - User management and SSO  
├── GIS Data Service (Node.js + PostGIS) - Spatial data operations  
├── SCADA Integration Service (Node.js + OPC UA) - Real-time SCADA communication  
├── AI Model Service (Python + FastAPI) - Machine learning model serving  
├── Notification Service (Node.js) - Alert and notification management  
├── File Processing Service (Python) - Data import/export operations  
└── Reporting Service (Node.js) - Report generation and analytics

**AI/ML Backend:** - **Framework**: FastAPI (Python) for high-performance AI model serving - **Model Serving**: TensorFlow Serving or MLflow Model Registry for production deployment - **Task Queue**: Celery with Redis for background processing and model training - **Model Training Pipeline**: Kubeflow or MLflow for complete ML lifecycle management

**Real-time Processing:** - **Message Broker**: Apache Kafka or Redis Streams for high-throughput event processing - **WebSocket Management**: Socket.IO with Redis adapter for clustering and scalability - **Event Processing**: Apache Kafka Streams or Node.js event handlers for real-time analytics

**Integration Layer:** - **SCADA Integration**: node-opcua for OPC UA communication with GE iFix - **External APIs**: Axios with retry and circuit breaker patterns for resilient integration - **File Processing**: Multer for uploads, Sharp for image processing and optimization

**Deployment:** - **Containerization**: Docker with multi-stage builds for optimized images - **Orchestration**: Kubernetes or Docker Swarm for container management - **Service Mesh**: Istio for advanced traffic management and security - **Monitoring**: Prometheus + Grafana for metrics, ELK stack for centralized logging

### Database Architecture

**Primary Databases:**

* 1. Spatial Database - PostgreSQL with PostGIS

water\_sources, canal\_network, control\_structures, irrigation\_zones, sensor\_locations

* 1. Time-Series Database - TimescaleDB (PostgreSQL Extension)

sensor\_readings, scada\_data, control\_commands, weather\_data, demand\_predictions, model\_performance

* 1. Document Database - MongoDB

// AI model configurations collection

// Optimization results collection

// System configuration collection

// Alert and notification templates

* 1. Cache and Session Store – Redis

# Real-time data cache

# User session management

# Real-time notifications

# AI model predictions cache

# System status cache

# Geospatial feature cache

1. Configuration and User Management – PostgreSQL

users, roles, system\_config, alarm\_config, audit\_log, user\_dashboards, maintenance\_schedule

with database schema and JSON details as described in Appendix A.

### Integration Architecture Specifications

**SCADA Integration (GE iFix) Enhanced:** - **Primary Method**: Hybrid OPC UA + REST Gateway for optimal performance and flexibility - **Secondary Method**: Enhanced WebSpace integration with SSO for full SCADA functionality - **Real-time Data Streaming**: WebSocket connections for sub-second data updates - **Failover Mechanisms**: Automatic fallback to backup communication channels - **Security**: End-to-end encryption with certificate-based authentication

**External System Integration:** - **Weather Services**: RESTful APIs with automatic retry and caching mechanisms - **Government Systems**: OAuth 2.0 integration with Thai Digital ID for seamless SSO - **AquaCrop Model**: Microservice architecture with dedicated Python service - **Emergency Systems**: Direct integration with national emergency response networks

**Security Implementation Enhanced:** - **Zero Trust Architecture**: Every component verified regardless of network location - **API Security**: JWT with short expiration, API rate limiting, request validation - **Database Security**: Row-level security, column encryption for sensitive data - **Network Security**: VPN access, firewall rules, intrusion detection - **Audit Compliance**: Complete audit trail for Thai government regulatory requirements

## Dashboard and user interface

The Thai-language dashboard interface will provide real-time visualization and control capabilities with enhanced AI-powered features.

### Real-time dashboard implementation

The dashboard will incorporate:

* **WebSocket implementation**: Primary communication channel for real-time updates with Socket.IO for compatibility and automatic fallbacks
* **AI-powered insights**: Integrated predictive analytics and optimization recommendations
* **Data optimizations**:
  + Data throttling based on value changes and network conditions
  + Client-side aggregation to reduce UI update frequency
  + Virtual scrolling and windowing for large datasets
  + Progressive loading with critical data prioritization
* **Chart visualizations**: Using Highcharts for water balance and efficiency analysis with templates for:
  + Water level monitoring with critical thresholds and AI predictions
  + Flow rate tracking with historical trends and forecasted patterns
  + Water quality multi-parameter displays with anomaly detection
  + Efficiency metrics with comparative analysis and optimization suggestions
  + AI model performance dashboards with real-time accuracy metrics

### Enhanced data export capabilities

Export functionality will include:

* **Format support**: CSV, XLSX, PDF, and GeoJSON exports for spatial data
* **Client-side implementation**: Direct browser downloads for smaller datasets
* **Server-side processing**: For large exports, scheduled reports, and complex analytics
* **Custom templates**: Pre-designed report templates with Thai government branding
* **AI-generated reports**: Automated insights and recommendations in exported documents
* **User experience**: Clear export options with date range selection, progress indicators, and email delivery

### Thai government UI requirements

The user interface will comply with Thai government standards:

* **Language support**: Primary Thai interface with English as secondary option, automatic language detection
* **Visual identity**: Clean, conservative design elements following DGA (Digital Government Agency) branding guidelines
* **Accessibility**: WCAG 2.1 AA compliance for accessibility standards
* **Mobile responsiveness**: Adaptive design optimized for field use on tablets and smartphones
* **Navigation patterns**: Standardized patterns consistent with gov.go.th design system
* **Performance**: Page load times under 3 seconds even on 3G connections

### Enhanced role-based access control

Access will be governed by granular role-based permissions:

* **Admin**: Full system configuration, user management, AI model deployment
* **AI Engineer**: Model training, deployment, performance monitoring
* **Analyst**: Data analysis, report generation, scenario modeling, AI insights access
* **Field Officer**: Monitoring, basic control functions, field data collection, mobile app access
* **Emergency Operator**: Override capabilities during critical situations
* **Viewer**: Read-only access to dashboards and reports
* **Guest**: Limited access for demonstrations and training

## Security and compliance

As a critical infrastructure system with AI components, enhanced security and compliance measures are essential.

### Enhanced Thai government security standards

The system will comply with:

* **Cybersecurity Act B.E. 2562 (2019)**: Implementation of required controls for Critical Information Infrastructure
* **NCSC Standards**: Compliance with NCSC 2023 guidelines for government systems
* **AI Governance Framework**: Adherence to Thailand’s AI Ethics Guidelines and NECTEC standards
* **Security Categorization**: Implementation of controls based on “high” impact classification
* **Regular Security Audits**: Quarterly penetration testing and annual comprehensive security assessments

### Enhanced data protection implementation

Data protection measures will include:

* **PDPA Compliance**: Full implementation of Personal Data Protection Act requirements
* **Data Localization**: Primary data center in Thailand, backup in Singapore (ASEAN compliance)
* **AI Data Governance**: Ethical use of data for AI training with consent management
* **Data Classification**: Multi-level classification with appropriate security controls
* **Retention Policies**: Automated data lifecycle management with secure deletion

### Advanced network security architecture

The network architecture will implement:

* **Zero Trust Network**: Every component authenticated and authorized
* **Micro-segmentation**: AI components isolated in secure network segments
* **Advanced Threat Detection**: ML-powered intrusion detection and prevention
* **Secure AI Pipeline**: Encrypted data flow throughout the AI/ML pipeline
* **Quantum-Safe Encryption**: Forward-looking encryption standards for long-term security

### Enhanced authentication system

Authentication will leverage:

* **Thai Digital ID Integration**: Seamless integration with national digital identity platform
* **Multi-Factor Authentication**: Mandatory for all administrative and AI management functions
* **Biometric Authentication**: For high-security areas and critical operations
* **Emergency Access Protocols**: Secure procedures for crisis situations
* **Session Management**: Advanced session monitoring with anomaly detection

## Database design patterns

The enhanced database architecture efficiently handles spatial, time-series, and AI/ML data.

### Advanced hybrid storage model

The system will use an optimized hybrid database approach:

* **Spatial data**: PostGIS with advanced spatial indexing and partitioning
* **Time-series data**: TimescaleDB with automatic compression and retention policies
* **AI/ML data**: Specialized storage for model artifacts, training data, and feature stores
* **Real-time streams**: Apache Kafka with custom serialization for high-throughput ingestion
* **Integration**: Advanced materialized views and streaming analytics for real-time insights

### Enhanced schema design patterns

The database schema includes optimized patterns for:

* **Temporal partitioning**: Automatic monthly partitioning for time-series data
* **Spatial clustering**: Geographic clustering for improved spatial query performance
* **AI feature stores**: Dedicated schemas for ML feature engineering and storage
* **Event sourcing**: Immutable event logs for complete system state reconstruction
* **CQRS implementation**: Separate read and write models for optimal performance

### Advanced query optimization patterns

Performance optimizations include:

* **Partial indexes**: Conditional indexes for frequently filtered data
* **Columnar storage**: Integration with Apache Parquet for analytics workloads
* **Query caching**: Multi-level caching strategy with Redis and application-level caching
* **Parallel processing**: Leveraging PostgreSQL’s parallel query capabilities
* **Real-time aggregations**: Continuous aggregates for instant analytics

### AI/ML data integration patterns

Data flows optimized for AI/ML workflows:

* **Feature engineering pipelines**: Automated feature extraction and transformation
* **Model versioning**: Complete lineage tracking for all AI model iterations
* **Real-time inference**: Low-latency data pipelines for real-time AI predictions
* **Training data management**: Efficient storage and retrieval of large training datasets

## Performance optimization

Enhanced performance optimization addresses the unique challenges of AI-powered systems in rural Thailand.

### Advanced client-side optimizations

Frontend performance enhanced through:

* **Progressive Web App (PWA)**: Sophisticated offline capabilities with local AI inference
* **Edge AI**: Client-side model inference for instant predictions without network calls
* **Adaptive streaming**: Dynamic quality adjustment based on network conditions
* **Predictive loading**: AI-powered prediction of user actions for proactive data loading
* **Efficient rendering**: Virtual DOM optimization and React Suspense for smooth UX

### Enhanced server-side optimizations

Backend performance improvements:

* **Edge computing**: Distributed AI inference nodes at provincial offices
* **Model optimization**: TensorFlow Lite and ONNX for efficient model serving
* **Caching strategies**: Multi-tier caching with Redis, CDN, and application caches
* **Load balancing**: Intelligent routing based on AI workload characteristics
* **Async processing**: Event-driven architecture for non-blocking operations

### AI-specific optimizations

Optimizations for AI components:

* **Model serving**: TensorFlow Serving with auto-scaling based on demand
* **Batch processing**: Efficient batch inference for non-real-time predictions
* **GPU utilization**: Dynamic GPU allocation for training and inference workloads
* **Model compression**: Quantization and pruning for faster inference
* **Pipeline parallelization**: Concurrent execution of AI pipeline stages

### Thailand-specific network optimizations

Enhanced optimizations for Thailand’s infrastructure:

* **Regional data centers**: Distributed deployment across Thailand’s regions
* **Satellite connectivity**: Backup connectivity for remote areas
* **Bandwidth optimization**: AI-powered traffic shaping and compression
* **Monsoon resilience**: Enhanced redundancy during extreme weather periods

## Implementation roadmap

The enhanced implementation roadmap incorporates AI development and deployment phases.

### Phase 1: Foundation with AI Infrastructure (0-6 months)

* Setup core infrastructure with AI/ML capabilities
* Implement enhanced SCADA integration with predictive analytics
* Deploy basic sensor network with edge AI processing
* Develop GIS database with spatial-temporal AI features
* Create intelligent dashboard with basic AI insights
* Establish AI/ML development and deployment pipelines

### Phase 2: AI Enhancement and Full Integration (6-12 months)

* Deploy spatial-temporal encoder for demand prediction
* Implement graph neural network for network optimization
* Develop multi-objective optimizer for automated control
* Deploy MPC AI model for real-time water management
* Enhance GIS functionality with AI-powered analytics
* Complete sensor network expansion with advanced analytics

### Phase 3: Advanced AI Optimization (12-18 months)

* Implement ensemble AI models for improved accuracy
* Deploy reinforcement learning for adaptive control
* Enhance automation with human-in-the-loop AI
* Develop advanced predictive maintenance systems
* Optimize performance based on AI usage patterns
* Complete integration with national smart agriculture initiatives

### Phase 4: Continuous Learning and Expansion (18+ months)

* Implement online learning for continuous model improvement
* Deploy federated learning across multiple irrigation projects
* Enhance AI explainability and trust mechanisms
* Develop advanced scenario planning and climate adaptation
* Expand to integrate with national water resource management

## Enhanced technology recommendations

Based on comprehensive research including AI/ML requirements:

| Component | Recommended Technology | Alternative |
| --- | --- | --- |
| SCADA Integration | Hybrid OPC UA + REST Gateway | Enhanced WebSpace with SSO |
| Database | PostgreSQL 15+ with PostGIS, TimescaleDB | MongoDB Atlas (document store) |
| AI/ML Platform | MLflow + Kubeflow + TensorFlow Serving | AWS SageMaker |
| GIS Server | GeoServer with vector tiles | MapServer with FastCGI |
| Base Map | Mapbox GL JS | OpenLayers with OSM |
| Frontend Framework | Next.js 13+ with TypeScript | Vue.js 3 with Nuxt |
| Real-time Communication | Socket.IO with Redis adapter | SignalR with Azure Service Bus |
| Visualization | Recharts + D3.js + Three.js | Chart.js + Highcharts |
| IoT Integration | Eclipse Mosquitto MQTT | AWS IoT Core |
| Authentication | OAuth 2.0 + Thai Digital ID | Auth0 with Thai SSO |
| Containerization | Kubernetes + Docker | Docker Swarm |
| Message Broker | Apache Kafka | RabbitMQ |
| Monitoring | Prometheus + Grafana + ELK | DataDog |
| AI Model Serving | TensorFlow Serving + NVIDIA Triton | Azure Machine Learning |

## Conclusion

The Enhanced Automatic Water Control System for Thailand’s Munbon Irrigation Project represents a paradigm shift in water management technology. By integrating advanced AI capabilities with traditional SCADA systems, real-time sensor networks, and comprehensive GIS functionality, the Royal Irrigation Department will achieve unprecedented precision and efficiency in water resource management.

The system’s AI-enhanced architecture provides:

* **Predictive capabilities** through spatial-temporal encoders and demand forecasting
* **Optimization intelligence** via graph neural networks and multi-objective optimization
* **Adaptive control** through model predictive control with physics-informed neural networks
* **Real-time insights** combining traditional monitoring with AI-powered analytics

The comprehensive technical architecture ensures scalability, security, and compliance with Thai government standards while providing the flexibility to evolve with advancing AI technologies. The phased implementation approach minimizes risk while delivering early value through foundational capabilities before advancing to sophisticated AI-driven automation.

Through careful integration of cutting-edge AI technologies with proven industrial control systems, this enhanced specification creates a robust foundation for Thailand’s digital transformation in agriculture and water management, supporting the nation’s sustainable development goals while ensuring food security and agricultural prosperity.

### Appendix – A

### Database Architecture

**Primary Databases:**

**1. Spatial Database - PostgreSQL with PostGIS**

-- Core spatial tables  
CREATE TABLE water\_sources (  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 type VARCHAR(50) CHECK (type IN ('reservoir', 'river', 'canal', 'groundwater')),  
 geometry GEOMETRY(POLYGON, 4326),  
 capacity\_m3 NUMERIC CHECK (capacity\_m3 > 0),  
 current\_level\_m NUMERIC,  
 max\_level\_m NUMERIC,  
 min\_operating\_level\_m NUMERIC,  
 created\_at TIMESTAMP DEFAULT NOW(),  
 updated\_at TIMESTAMP DEFAULT NOW()  
);  
  
CREATE TABLE canal\_network (  
 id SERIAL PRIMARY KEY,  
 segment\_name VARCHAR(255) NOT NULL,  
 geometry GEOMETRY(LINESTRING, 4326),  
 capacity\_m3\_s NUMERIC CHECK (capacity\_m3\_s > 0),  
 length\_m NUMERIC CHECK (length\_m > 0),  
 roughness\_coefficient NUMERIC DEFAULT 0.013,  
 upstream\_node\_id INTEGER,  
 downstream\_node\_id INTEGER,  
 operational\_status VARCHAR(20) DEFAULT 'active' CHECK (operational\_status IN ('active', 'maintenance', 'inactive')),  
 last\_maintenance DATE,  
 next\_maintenance DATE  
);  
  
CREATE TABLE control\_structures (  
 id SERIAL PRIMARY KEY,  
 gate\_id VARCHAR(50) UNIQUE NOT NULL,  
 name VARCHAR(255) NOT NULL,  
 type VARCHAR(50) CHECK (type IN ('gate', 'pump', 'valve', 'weir')),  
 geometry GEOMETRY(POINT, 4326),  
 max\_flow\_rate\_m3\_s NUMERIC CHECK (max\_flow\_rate\_m3\_s > 0),  
 automation\_level VARCHAR(20) DEFAULT 'manual' CHECK (automation\_level IN ('manual', 'semi-auto', 'automatic')),  
 scada\_tag VARCHAR(100),  
 installation\_date DATE,  
 maintenance\_schedule JSONB,  
 current\_position\_percent NUMERIC DEFAULT 0 CHECK (current\_position\_percent >= 0 AND current\_position\_percent <= 100),  
 operational\_status VARCHAR(20) DEFAULT 'operational'  
);  
  
CREATE TABLE irrigation\_zones (  
 id SERIAL PRIMARY KEY,  
 zone\_name VARCHAR(255) NOT NULL,  
 geometry GEOMETRY(POLYGON, 4326),  
 area\_hectares NUMERIC CHECK (area\_hectares > 0),  
 primary\_crop VARCHAR(100),  
 secondary\_crop VARCHAR(100),  
 irrigation\_method VARCHAR(50) CHECK (irrigation\_method IN ('flood', 'sprinkler', 'drip', 'furrow')),  
 water\_allocation\_m3 NUMERIC CHECK (water\_allocation\_m3 >= 0),  
 farmer\_count INTEGER DEFAULT 0,  
 district VARCHAR(100),  
 province VARCHAR(100),  
 planting\_season JSONB,  
 harvest\_season JSONB  
);  
  
CREATE TABLE sensor\_locations (  
 id SERIAL PRIMARY KEY,  
 sensor\_id VARCHAR(50) UNIQUE NOT NULL,  
 sensor\_type VARCHAR(50) CHECK (sensor\_type IN ('water\_level', 'flow', 'moisture', 'weather', 'pressure')),  
 geometry GEOMETRY(POINT, 4326),  
 installation\_date DATE,  
 calibration\_date DATE,  
 next\_calibration\_date DATE,  
 transmission\_method VARCHAR(20) CHECK (transmission\_method IN ('LoRa', 'cellular', 'wifi', 'satellite')),  
 gateway\_id VARCHAR(50),  
 maintenance\_status VARCHAR(20) DEFAULT 'good' CHECK (maintenance\_status IN ('good', 'warning', 'critical', 'offline')),  
 battery\_level\_percent NUMERIC CHECK (battery\_level\_percent >= 0 AND battery\_level\_percent <= 100),  
 signal\_strength\_dbm NUMERIC  
);  
  
-- Create spatial indexes  
CREATE INDEX idx\_water\_sources\_geom ON water\_sources USING GIST (geometry);  
CREATE INDEX idx\_canal\_network\_geom ON canal\_network USING GIST (geometry);  
CREATE INDEX idx\_control\_structures\_geom ON control\_structures USING GIST (geometry);  
CREATE INDEX idx\_irrigation\_zones\_geom ON irrigation\_zones USING GIST (geometry);  
CREATE INDEX idx\_sensor\_locations\_geom ON sensor\_locations USING GIST (geometry);  
  
-- Create composite indexes for common queries  
CREATE INDEX idx\_control\_structures\_type\_status ON control\_structures (type, operational\_status);  
CREATE INDEX idx\_sensor\_locations\_type\_status ON sensor\_locations (sensor\_type, maintenance\_status);

**2. Time-Series Database - TimescaleDB (PostgreSQL Extension)**

-- Real-time sensor readings  
CREATE TABLE sensor\_readings (  
 time TIMESTAMPTZ NOT NULL,  
 sensor\_id VARCHAR(50) NOT NULL,  
 reading\_type VARCHAR(50) NOT NULL,  
 value NUMERIC NOT NULL,  
 unit VARCHAR(20),  
 quality\_flag SMALLINT DEFAULT 0 CHECK (quality\_flag IN (0, 1, 2)), -- 0=good, 1=uncertain, 2=bad  
 processing\_status VARCHAR(20) DEFAULT 'raw' CHECK (processing\_status IN ('raw', 'validated', 'corrected')),  
 PRIMARY KEY (time, sensor\_id, reading\_type),  
 FOREIGN KEY (sensor\_id) REFERENCES sensor\_locations (sensor\_id)  
);  
  
SELECT create\_hypertable('sensor\_readings', 'time', chunk\_time\_interval => INTERVAL '1 day');  
  
-- SCADA data points  
CREATE TABLE scada\_data (  
 time TIMESTAMPTZ NOT NULL,  
 tag\_name VARCHAR(100) NOT NULL,  
 value NUMERIC,  
 string\_value TEXT,  
 quality VARCHAR(20) DEFAULT 'good',  
 alarm\_status BOOLEAN DEFAULT false,  
 operator\_id VARCHAR(50),  
 PRIMARY KEY (time, tag\_name)  
);  
  
SELECT create\_hypertable('scada\_data', 'time', chunk\_time\_interval => INTERVAL '1 day');  
  
-- Control commands log  
CREATE TABLE control\_commands (  
 time TIMESTAMPTZ NOT NULL,  
 device\_id VARCHAR(50) NOT NULL,  
 command\_type VARCHAR(50) NOT NULL,  
 command\_value NUMERIC,  
 command\_string VARCHAR(255),  
 operator\_id VARCHAR(50),  
 execution\_status VARCHAR(20) DEFAULT 'pending' CHECK (execution\_status IN ('pending', 'executing', 'completed', 'failed')),  
 error\_message TEXT,  
 acknowledgment\_time TIMESTAMPTZ,  
 PRIMARY KEY (time, device\_id, command\_type),  
 FOREIGN KEY (device\_id) REFERENCES control\_structures (gate\_id)  
);  
  
-- Weather data  
CREATE TABLE weather\_data (  
 time TIMESTAMPTZ NOT NULL,  
 station\_id VARCHAR(50) NOT NULL,  
 temperature\_c NUMERIC,  
 humidity\_percent NUMERIC CHECK (humidity\_percent >= 0 AND humidity\_percent <= 100),  
 rainfall\_mm NUMERIC CHECK (rainfall\_mm >= 0),  
 wind\_speed\_ms NUMERIC CHECK (wind\_speed\_ms >= 0),  
 wind\_direction\_degrees NUMERIC CHECK (wind\_direction\_degrees >= 0 AND wind\_direction\_degrees < 360),  
 evapotranspiration\_mm NUMERIC CHECK (evapotranspiration\_mm >= 0),  
 solar\_radiation\_wm2 NUMERIC CHECK (solar\_radiation\_wm2 >= 0),  
 barometric\_pressure\_hpa NUMERIC,  
 PRIMARY KEY (time, station\_id)  
);  
  
SELECT create\_hypertable('weather\_data', 'time', chunk\_time\_interval => INTERVAL '1 day');  
  
-- Water demand predictions  
CREATE TABLE demand\_predictions (  
 time TIMESTAMPTZ NOT NULL,  
 zone\_id INTEGER NOT NULL,  
 prediction\_horizon\_hours INTEGER NOT NULL,  
 predicted\_demand\_m3 NUMERIC CHECK (predicted\_demand\_m3 >= 0),  
 confidence\_interval\_lower NUMERIC,  
 confidence\_interval\_upper NUMERIC,  
 model\_version VARCHAR(50),  
 created\_at TIMESTAMPTZ DEFAULT NOW(),  
 PRIMARY KEY (time, zone\_id, prediction\_horizon\_hours),  
 FOREIGN KEY (zone\_id) REFERENCES irrigation\_zones (id)  
);  
  
-- AI model performance metrics  
CREATE TABLE model\_performance (  
 time TIMESTAMPTZ NOT NULL,  
 model\_name VARCHAR(100) NOT NULL,  
 model\_version VARCHAR(50) NOT NULL,  
 metric\_name VARCHAR(50) NOT NULL,  
 metric\_value NUMERIC NOT NULL,  
 validation\_dataset VARCHAR(100),  
 PRIMARY KEY (time, model\_name, model\_version, metric\_name)  
);  
  
-- Create indexes for efficient time-series queries  
CREATE INDEX idx\_sensor\_readings\_sensor\_time ON sensor\_readings (sensor\_id, time DESC);  
CREATE INDEX idx\_scada\_data\_tag\_time ON scada\_data (tag\_name, time DESC);  
CREATE INDEX idx\_control\_commands\_device\_time ON control\_commands (device\_id, time DESC);  
CREATE INDEX idx\_weather\_data\_station\_time ON weather\_data (station\_id, time DESC);  
  
-- Create continuous aggregates for common analytics  
CREATE MATERIALIZED VIEW hourly\_sensor\_avg  
WITH (timescaledb.continuous) AS  
SELECT time\_bucket('1 hour', time) AS hour,  
 sensor\_id,  
 reading\_type,  
 AVG(value) AS avg\_value,  
 MIN(value) AS min\_value,  
 MAX(value) AS max\_value,  
 COUNT(\*) AS reading\_count  
FROM sensor\_readings  
WHERE quality\_flag = 0  
GROUP BY hour, sensor\_id, reading\_type;  
  
CREATE MATERIALIZED VIEW daily\_water\_balance  
WITH (timescaledb.continuous) AS  
SELECT time\_bucket('1 day', time) AS day,  
 SUM(CASE WHEN command\_type = 'flow\_in' THEN command\_value ELSE 0 END) AS total\_inflow\_m3,  
 SUM(CASE WHEN command\_type = 'flow\_out' THEN command\_value ELSE 0 END) AS total\_outflow\_m3,  
 COUNT(DISTINCT device\_id) AS active\_devices  
FROM control\_commands  
WHERE execution\_status = 'completed'  
GROUP BY day;

**3. Document Database - MongoDB**

// AI model configurations collection  
{  
 \_id: ObjectId,  
 model\_name: "spatial\_temporal\_encoder\_v1.2",  
 model\_type: "tensorflow",  
 version: "1.2.0",  
 status: "active", // active, deprecated, training  
 config: {  
 input\_shape: [168, 50],  
 hidden\_layers: [128, 64, 32],  
 activation: "relu",  
 optimizer: "adam",  
 learning\_rate: 0.001  
 },  
 training\_data: {  
 start\_date: "2023-01-01",  
 end\_date: "2024-01-01",  
 features: ["temperature", "rainfall", "soil\_moisture", "crop\_stage"],  
 data\_sources: ["sensor\_readings", "weather\_data", "crop\_calendar"]  
 },  
 performance\_metrics: {  
 mse: 0.0234,  
 mae: 0.156,  
 r2\_score: 0.94,  
 validation\_accuracy: 0.89,  
 prediction\_horizon\_hours: 48  
 },  
 deployment\_config: {  
 endpoint: "/api/v1/predict/demand",  
 scaling: "auto",  
 max\_instances: 10,  
 resource\_requirements: {  
 cpu: "2 cores",  
 memory: "8GB",  
 gpu: "optional"  
 }  
 },  
 feature\_importance: {  
 "temperature": 0.35,  
 "rainfall": 0.28,  
 "soil\_moisture": 0.22,  
 "crop\_stage": 0.15  
 },  
 created\_at: ISODate(),  
 last\_updated: ISODate(),  
 created\_by: "system\_admin",  
 model\_file\_path: "/models/spatial\_temporal/v1.2/model.h5"  
}  
  
// Optimization results collection  
{  
 \_id: ObjectId,  
 optimization\_id: "opt\_20241201\_001",  
 timestamp: ISODate(),  
 optimization\_type: "multi\_objective", // single\_objective, multi\_objective, pareto  
 status: "completed", // pending, running, completed, failed  
 input\_parameters: {  
 time\_horizon\_hours: 48,  
 demand\_forecast: [  
 {zone\_id: 1, hourly\_demand: [...]},  
 {zone\_id: 2, hourly\_demand: [...]}  
 ],  
 network\_constraints: {  
 max\_flow\_rates: {...},  
 storage\_limits: {...},  
 operational\_constraints: {...}  
 },  
 objective\_weights: {  
 water\_delivery\_accuracy: 0.4,  
 energy\_efficiency: 0.3,  
 equipment\_wear\_minimization: 0.2,  
 water\_conservation: 0.1  
 },  
 uncertainty\_parameters: {  
 demand\_uncertainty: 0.15,  
 weather\_uncertainty: 0.10  
 }  
 },  
 solution: {  
 gate\_schedules: [  
 {  
 gate\_id: "G001",  
 schedule: [  
 {time: "2024-12-01T06:00:00Z", position\_percent: 75, flow\_rate\_m3s: 12.5},  
 {time: "2024-12-01T07:00:00Z", position\_percent: 80, flow\_rate\_m3s: 13.2}  
 ]  
 }  
 ],  
 pump\_schedules: [  
 {  
 pump\_id: "P001",  
 schedule: [  
 {time: "2024-12-01T06:00:00Z", status: "on", flow\_rate\_m3s: 8.5, power\_kw: 45.2}  
 ]  
 }  
 ],  
 expected\_outcomes: {  
 total\_energy\_consumption\_kwh: 2450,  
 water\_delivery\_efficiency\_percent: 94.5,  
 predicted\_crop\_yield\_improvement\_percent: 8.2  
 }  
 },  
 performance\_metrics: {  
 objective\_value: 0.87,  
 pareto\_efficiency: true,  
 constraint\_violations: 0,  
 computation\_time\_s: 45.7,  
 convergence\_iterations: 125  
 },  
 sensitivity\_analysis: {  
 demand\_sensitivity: 0.12,  
 weather\_sensitivity: 0.08,  
 equipment\_sensitivity: 0.05  
 },  
 validation\_results: {  
 back\_testing\_accuracy: 0.91,  
 real\_world\_validation: {  
 implemented: true,  
 actual\_vs\_predicted\_deviation: 0.07  
 }  
 },  
 created\_by: "auto\_scheduler",  
 approved\_by: "operator\_001",  
 implementation\_status: "deployed" // pending, approved, deployed, archived  
}  
  
// System configuration collection  
{  
 \_id: ObjectId,  
 config\_type: "ai\_models",  
 config\_name: "model\_ensemble\_weights",  
 config\_value: {  
 spatial\_temporal\_encoder: 0.4,  
 graph\_neural\_network: 0.35,  
 physics\_informed\_nn: 0.25  
 },  
 description: "Weights for ensemble model predictions",  
 valid\_from: ISODate(),  
 valid\_until: ISODate(),  
 created\_by: "ai\_engineer",  
 approval\_status: "approved"  
}  
  
// Alert and notification templates  
{  
 \_id: ObjectId,  
 template\_name: "water\_level\_critical",  
 alert\_type: "critical",  
 condition: {  
 sensor\_type: "water\_level",  
 operator: "less\_than",  
 threshold\_value: 2.0,  
 threshold\_unit: "meters",  
 duration\_minutes: 15  
 },  
 notification\_config: {  
 channels: ["email", "sms", "dashboard"],  
 recipients: ["duty\_officer", "irrigation\_manager"],  
 escalation\_levels: [  
 {level: 1, delay\_minutes: 5, recipients: ["duty\_officer"]},  
 {level: 2, delay\_minutes: 15, recipients: ["irrigation\_manager"]},  
 {level: 3, delay\_minutes: 30, recipients: ["district\_director"]}  
 ]  
 },  
 message\_templates: {  
 thai: "ระดับน้ำในอ่างเก็บน้ำ {reservoir\_name} ต่ำกว่าระดับวิกฤต {current\_level}m",  
 english: "Water level in {reservoir\_name} is critically low at {current\_level}m"  
 },  
 active: true,  
 created\_at: ISODate()  
}

**4. Cache and Session Store - Redis**

# Real-time data cache  
munbon:realtime:sensor:{sensor\_id}:latest -> {  
 "value": 25.6,  
 "timestamp": "2024-12-01T10:30:00Z",  
 "quality": "good",  
 "unit": "m3/s"  
}  
  
munbon:realtime:scada:{tag\_name}:latest -> {  
 "value": 85.5,  
 "timestamp": "2024-12-01T10:30:00Z",  
 "alarm": false,  
 "quality": "good"  
}  
  
# User session management  
munbon:session:{session\_id} -> {  
 "user\_id": "user123",  
 "role": "field\_officer",  
 "permissions": ["read\_sensors", "basic\_control"],  
 "last\_activity": "2024-12-01T10:30:00Z",  
 "expires\_at": "2024-12-01T18:30:00Z"  
}  
  
# Real-time notifications  
munbon:notifications:{user\_id} -> [  
 {  
 "id": "notif\_001",  
 "type": "alert",  
 "message": "Gate G001 malfunction detected",  
 "timestamp": "2024-12-01T10:25:00Z",  
 "read": false,  
 "priority": "high"  
 }  
]  
  
# AI model predictions cache  
munbon:predictions:demand:{zone\_id}:{date} -> {  
 "predictions": [12.5, 13.2, 14.8, ...],  
 "confidence\_intervals": [[11.2, 13.8], [12.1, 14.3], ...],  
 "model\_version": "v1.2",  
 "generated\_at": "2024-12-01T06:00:00Z"  
}  
  
# System status cache  
munbon:system:health -> {  
 "database\_status": "healthy",  
 "ai\_models\_status": "healthy",  
 "scada\_connection": "connected",  
 "sensor\_network": "98% operational",  
 "last\_check": "2024-12-01T10:30:00Z"  
}  
  
# Geospatial feature cache  
munbon:gis:features:{layer\_name}:{bbox} -> {  
 "features": [...],  
 "cached\_at": "2024-12-01T10:00:00Z",  
 "expires\_at": "2024-12-01T11:00:00Z"  
}

**5. Configuration and User Management - PostgreSQL**

-- User management with Thai government integration  
CREATE TABLE users (  
 id SERIAL PRIMARY KEY,  
 username VARCHAR(100) UNIQUE NOT NULL,  
 email VARCHAR(255) UNIQUE,  
 full\_name\_thai VARCHAR(255),  
 full\_name\_english VARCHAR(255),  
 role VARCHAR(50) NOT NULL CHECK (role IN ('admin', 'analyst', 'field\_officer', 'viewer', 'emergency\_operator')),  
 permissions JSONB DEFAULT '[]',  
 thai\_citizen\_id VARCHAR(13) UNIQUE, -- For Thai government integration  
 department VARCHAR(100),  
 position VARCHAR(100),  
 phone\_number VARCHAR(20),  
 emergency\_contact VARCHAR(20),  
 last\_login TIMESTAMPTZ,  
 account\_status VARCHAR(20) DEFAULT 'active' CHECK (account\_status IN ('active', 'inactive', 'suspended', 'locked')),  
 password\_last\_changed TIMESTAMPTZ DEFAULT NOW(),  
 failed\_login\_attempts INTEGER DEFAULT 0,  
 account\_locked\_until TIMESTAMPTZ,  
 created\_at TIMESTAMP DEFAULT NOW(),  
 created\_by VARCHAR(100),  
 last\_modified TIMESTAMPTZ DEFAULT NOW(),  
 two\_factor\_enabled BOOLEAN DEFAULT false,  
 preferred\_language VARCHAR(5) DEFAULT 'th' CHECK (preferred\_language IN ('th', 'en'))  
);  
  
-- Role-based permissions  
CREATE TABLE roles (  
 id SERIAL PRIMARY KEY,  
 role\_name VARCHAR(50) UNIQUE NOT NULL,  
 description\_thai TEXT,  
 description\_english TEXT,  
 permissions JSONB NOT NULL DEFAULT '[]',  
 created\_at TIMESTAMP DEFAULT NOW(),  
 is\_system\_role BOOLEAN DEFAULT false  
);  
  
-- System configuration  
CREATE TABLE system\_config (  
 key VARCHAR(100) PRIMARY KEY,  
 value JSONB NOT NULL,  
 description\_thai TEXT,  
 description\_english TEXT,  
 category VARCHAR(50),  
 data\_type VARCHAR(20) CHECK (data\_type IN ('string', 'number', 'boolean', 'json', 'array')),  
 validation\_rules JSONB,  
 is\_sensitive BOOLEAN DEFAULT false,  
 requires\_restart BOOLEAN DEFAULT false,  
 last\_modified TIMESTAMPTZ DEFAULT NOW(),  
 modified\_by VARCHAR(100),  
 version INTEGER DEFAULT 1  
);  
  
-- Alarm and alert configurations  
CREATE TABLE alarm\_config (  
 id SERIAL PRIMARY KEY,  
 alarm\_name VARCHAR(255) NOT NULL,  
 alarm\_code VARCHAR(50) UNIQUE,  
 description\_thai TEXT,  
 description\_english TEXT,  
 condition\_expression TEXT NOT NULL,  
 severity VARCHAR(20) DEFAULT 'medium' CHECK (severity IN ('low', 'medium', 'high', 'critical')),  
 notification\_methods JSONB DEFAULT '["dashboard"]',  
 auto\_acknowledge BOOLEAN DEFAULT false,  
 escalation\_rules JSONB,  
 enabled BOOLEAN DEFAULT true,  
 category VARCHAR(50),  
 related\_systems JSONB DEFAULT '[]',  
 created\_at TIMESTAMP DEFAULT NOW(),  
 created\_by VARCHAR(100),  
 last\_tested TIMESTAMPTZ,  
 test\_status VARCHAR(20)  
);  
  
-- Audit trail for all system changes  
CREATE TABLE audit\_log (  
 id BIGSERIAL PRIMARY KEY,  
 timestamp TIMESTAMPTZ DEFAULT NOW(),  
 user\_id VARCHAR(100),  
 action VARCHAR(100) NOT NULL,  
 resource\_type VARCHAR(50),  
 resource\_id VARCHAR(100),  
 old\_values JSONB,  
 new\_values JSONB,  
 ip\_address INET,  
 user\_agent TEXT,  
 session\_id VARCHAR(255),  
 success BOOLEAN NOT NULL,  
 error\_message TEXT,  
 additional\_context JSONB  
);  
  
-- Create hypertable for audit log (time-series optimization)  
SELECT create\_hypertable('audit\_log', 'timestamp', chunk\_time\_interval => INTERVAL '1 week');  
  
-- Dashboard layouts and user preferences  
CREATE TABLE user\_dashboards (  
 id SERIAL PRIMARY KEY,  
 user\_id INTEGER NOT NULL REFERENCES users(id),  
 dashboard\_name VARCHAR(255) NOT NULL,  
 layout\_config JSONB NOT NULL,  
 is\_default BOOLEAN DEFAULT false,  
 is\_public BOOLEAN DEFAULT false,  
 created\_at TIMESTAMP DEFAULT NOW(),  
 last\_modified TIMESTAMPTZ DEFAULT NOW(),  
 UNIQUE(user\_id, dashboard\_name)  
);  
  
-- System maintenance schedules  
CREATE TABLE maintenance\_schedules (  
 id SERIAL PRIMARY KEY,  
 equipment\_id VARCHAR(100) NOT NULL,  
 equipment\_type VARCHAR(50),  
 maintenance\_type VARCHAR(50) CHECK (maintenance\_type IN ('preventive', 'corrective', 'emergency', 'calibration')),  
 scheduled\_date DATE NOT NULL,  
 estimated\_duration\_hours NUMERIC,  
 assigned\_technician VARCHAR(100),  
 status VARCHAR(20) DEFAULT 'scheduled' CHECK (status IN ('scheduled', 'in\_progress', 'completed', 'postponed', 'cancelled')),  
 completion\_date TIMESTAMPTZ,  
 notes\_thai TEXT,  
 notes\_english TEXT,  
 cost\_estimate NUMERIC,  
 actual\_cost NUMERIC,  
 created\_at TIMESTAMP DEFAULT NOW(),  
 created\_by VARCHAR(100)  
);  
  
-- Create indexes for efficient queries  
CREATE INDEX idx\_users\_role ON users (role);  
CREATE INDEX idx\_users\_status ON users (account\_status);  
CREATE INDEX idx\_users\_thai\_id ON users (thai\_citizen\_id);  
CREATE INDEX idx\_audit\_log\_user\_action ON audit\_log (user\_id, action);  
CREATE INDEX idx\_audit\_log\_timestamp ON audit\_log (timestamp DESC);  
CREATE INDEX idx\_alarm\_config\_severity ON alarm\_config (severity, enabled);  
CREATE INDEX idx\_maintenance\_schedules\_date ON maintenance\_schedules (scheduled\_date);