ZLAN Switch System

Connecting Your Power Grid to the Internet

Table of Contents

1. Introduction
2. System Architecture
3. Frontend
4. Backend
5. Service Layer
6. Challenges & Solutions
7. Future Enhancements
8. Demo
9. Q&A

Overview

Project Overview:

Development of a comprehensive Meter Monitoring System for real-time electrical energy data visualization and analysis

Objectives:

- Collect accurate meter readings via Modbus TCP communication.
- Store data efficiently using FastAPI backend with SQLite database.
- Provide an interactive frontend dashboard with real-time updates and historical data filtering.
- Enable easy deployment and scalability across different devices.

Key Benefits:

- Improved energy monitoring and management.
- User-friendly interface for data insight.
- Reliable and extensible architecture.

System Architecture

Three Core Components:
Frontend: React-based dashboard for data visualization and user interaction.
Backend: FastAPI REST API with SQLite database for data storage and querying.
Service Layer: Python script communicating with energy meters via Modbus TCP, pushing data to backend.

Data Flow:
Service reads data from meters → sends readings to backend API → backend stores in database → frontend fetches and displays data in real-time.

Technologies Used:
React, shadcn/ui for frontend UI
FastAPI, SQLAlchemy, SQLite for backend
pymodbus for Modbus communication in service

Deployment:
Each component runs independently but communicates over HTTP (backend & frontend) and network (service & backend).

Frontend

Technology Stack:

- React with Next.js app router
- shadcn/ui components for polished UI
- Recharts for interactive data visualization

Key Features:

- Real-time meter readings with automatic refresh
- Date selector to filter data by specific day
- Connection status indicators and manual refresh button

User Experience:

- Clean, intuitive dashboard layout
- interactive line charts showing power, voltage, current trends
- Smooth transitions and tooltips for detailed data inspection

Backend

Framework & Database:

- FastAPI for high-performance REST API
- SQLite database for lightweight, file-based storage
- SQLAlchemy ORM for easy database interaction

Core Responsibilities:

- Receive and store meter readings via POST endpoints
- Serve filtered and paginated data via GET endpoints
- Handle date-based queries to support frontend filtering

Additional Features:

- CORS enabled for frontend-backend communication
- Environment variable configuration for flexibility
- Data persistence with efficient file/database management

Scalability:

- Modular design to allow future upgrades to more powerful databases if needed

Service Layer

Purpose:

- Continuously collect real-time data from electrical meters via Modbus TCP protocol.

Implementation:

- Python script using pymodbus library to communicate with meters over the network.
- Reads parameters like voltage, current, power, frequency, and energy at set intervals.

Data Handling:

- Posts collected readings to the backend API for storage
- Handles connection retries and error logging to ensure reliability.

Deployment:

- Runs independently, can be hosted on the same network as meters or backend.
- Easily configurable via environment variables (e.g., meter IP address).

Challenges

Handling Missing Data

- Null or incomplete meter readings addressed by interpolation and last-known-value fill strategies in the backend.

Real-time Updates & Connectivity:

- Ensured reliable data flow with periodic polling and connection status indicators on the frontend.

Date Filtering & User Interaction:

- Implemented flexible date selectors with proper formatting and backend support for efficient querying.

Scalability & Extensibility:

- Designed modular architecture allowing easy upgrade from SQLite to more robust databases as needed

Deployment Complexity:

- Simplified setup with environment variables and clear separation of concerns across components.

Future Enhancements

Advanced Data Analytics: Implement predictive analytics and anomaly detection for proactive energy management.

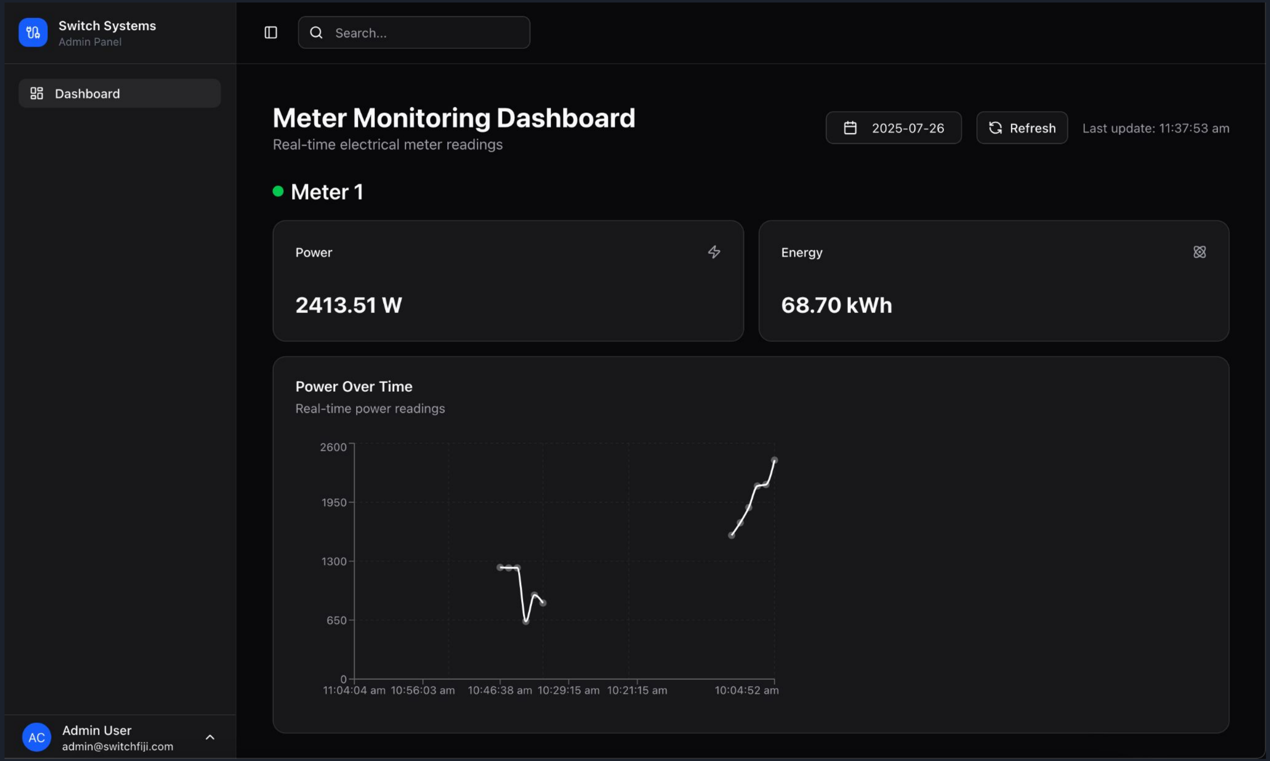
Mobile Application: Develop a mobile app to provide on-the-go access and notifications.

Extended Device Support:Add compatibility with additional meter models and communication protocols.

User Management & Security:Integrate authentication, role-based access, and encrypted data transmission.

Cloud Deployment & Scaling: Migrate backend to cloud infrastructure for higher availability and scalability.

Demo



Meter Monitoring Dashboard

Real-time electrical meter readings

Last update: 11:37:53 am

Meter 1

Power

2413.51 W

Energy

68.70 kWh

Power Over Time

Real-time power readings

Any Questions?