Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Proposal

on

"Kathmandu University Smart Meter Monitoring and Analysis System"

[Code No: COMP 303]

(For partial fulfillment of III Year/ I Semester in Computer Engineering)

Submitted by

Bijan Bakabal Thapa (Roll No. 7)
Manish Bhattarai (Roll No. 11)
Sadit Rasaili (Roll No. 40)
Sameep Kakshapati Shrestha (Roll No. 20)
Slok Pradhan (Roll No. 38)

Submitted to

Project Coordinator

Department of Computer Science and Engineering

Submission Date: 15/11/2025

Abstract

This project presents the design and development of a real-time Smart Meter Analytics System for Kathmandu University, aimed at improving energy management, operational reliability, and academic engagement. While traditional analog metering offers only cumulative consumption readings, modern smart meters generate high-resolution electrical data such as energy usage, voltage stability, outage occurrences, and load variability. To harness this potential, the proposed system will integrate directly with the NEA Smart Meter API, enabling continuous IoT-based data acquisition, automated refresh cycles, and secure data handling through OAuth authentication.

A fully interactive web-enabled dashboard, developed using React for the frontend and Django for the backend, will visualize real-time and historical data using intuitive graphs, trend analyses, and block-level comparisons. The system will further incorporate AI/ML forecasting models to provide predictive insights into electricity consumption, voltage trends, peak load periods, and potential outage risks—supporting evidence-based planning and early anomaly detection. A scalable MongoDB-backed data architecture, combined with containerized deployment using Docker, ensures efficiency, modularity, and the ability to integrate multiple smart meters across KU facilities.

Through comprehensive visual analytics, real-time monitoring, and forecasting intelligence, the Smart Meter Analytics System will enhance KU's energy transparency, operational resilience, and sustainability initiatives.

Keywords: Smart Meter, Real-Time Monitoring, Energy Analytics, IoT, Forecasting, Data Visualization, Smart Grid.

Table of Contents

List of Figures

Abbreviation

i. API: Application Programming Interface
ii. IoT: Internet of Things
iii. MQTT: Message Queuing Telemetry Transport
iv. NEA: Nepal Electricity Authority
v. UI: User Interface
vi. UAT: User Acceptance Testing
vii. AI: Artificial Intelligence
viii. ML: Machine Learning
ix. kWh: Kilowatt-hour
x. OAuth: Open Authorization (OAuth 2.0 security protocol)
xi. RAM: Random Access Memory
xii. SSD: Solid-State Drive
xiii. HTTPS: Hypertext Transfer Protocol Secure
xiv. CPU: Central Processing Unit (implied when mentioning processors)
xv. GB: Gigabyte
xvi. Mbps: Megabits per second
xvii. LTS: Long Term Support (Ubuntu version)
xviii. CSV: Comma-Separated Values
xix. PDF: Portable Document Format
xx. DB: Database

Chapter 1 Introduction

1.1 Background

Modern energy systems increasingly leverage real-time data to improve efficiency, reliability and transparency. Traditional analog metering technology is capable of measuring total consumption of energy but provides little visibility regarding energy usage, the stability of voltage, or the occurrence of outages. Conversely, the support for more energy consumption, coupled with adjacent electrical networks, leads to a greater need for smarter and data-driven monitoring systems.

Smart meters are advanced digital devices capable of recording numerous electrical data types including energy consumption, voltage, balance of power, duration of outages, and loads variability. These measurements are published through communication devices as streams of data interfaces or API's to cloud-based / remote data dashboards, or local monitoring systems. It is Kathmandu University's intention to leverage this data to develop a real-time dashboard in order to enhance a better energy management system with enhanced educational opportunities for research.

The project proposes to develop a fully interactive web-enabled dashboard that collects data from a smart meter in real-time using API calls and refreshes data visually. The dashboard could display current data points regarding energy consumption, a monthly trends graph, quick voltage stability checks, historical outages summary, and projected usage. The dashboard will enable better energy management for the university and related benefits for educational opportunities in smart-grid technologies.

1.2 Objectives

i. Develop a dynamic website for visualizing real-time smart meter data.
ii. Implement API integration for automatic data refresh.
iii. Display measurements including energy consumption, voltage range, outage events, and power balance / power ratio.
iv. Provide monthly analytics, summary reports, and graphical representations using cloud based storage.
v. Offer predictive insights through forecasting models.
vi. Build a scalable system that can be integrated with other multiple smart meters.

1.3 Significance and Motivation

Efficient energy management systems are becoming increasingly important to support modern infrastructure, especially among educational and research institutions. Kathmandu University operates a wide spectrum of laboratories, student hostels, offices, and facilities where the role of electrical energy consumption is vital to operations. When energy usage and conditions are not continuously witnessed, it can be difficult to detect issues that include but are not limited to line voltage drop, wasted energy, a power line draw attached to a 'ghost load,' or system component failure occurring from a local or distributed outage.

This project is motivated to address the feasibility requirement of a transparent, data-driven energy monitoring platform that is beneficial to administrators on university and students' learning opportunities. The monitoring system translates raw smart-meter data into unobtrusive visualizations for better improved decision making, reliability of power, and support for sustainability initiatives. Furthermore, this presents an academic asset where students can study real-world applications of an API, data visualization, smart technologies, and IoT-related monitoring based systems.

The capability of this system is not limited to Kathmandu University, and with future scaling considerations, it could lead to households, industries, or work related to smart-grid initiatives, making it a relevant, meaningful, and impactful technological contribution.

1.4 Expected Outcomes

The successful completion of the project will deliver following key outcomes:

i. **A Dynamic Web Dashboard**
   A web-based interface that is responsive and fully functional to deliver progressive smart meter readings for energy consumption, voltage, power balance, and outage information in real-time.

ii. **Automated Data Retrieval System**
   An API-based data collection system built to reliably and accurately collect smart meter measurements dynamically.

iii. **Comprehensive Visual Analytics**
   Different visual representations of data for clear insights and decision making.

iv. **Forecasting and Predictive Insights**
   Forecasting models in the system to predict energy consumption, voltage trends and risk of outage for planning and optimization.

v. **Scalable System Architecture**
   An extensible and modular system with the capability to add other smart meters and expand the system over the Kathmandu University.

Chapter 2 Related Works

In exploring development of smart meter data visualization and energy-monitoring platforms, reviewing existing systems is crucial to understanding the technological landscape and identifying the gaps our project aims to address. This chapter highlights several relevant platforms used for energy monitoring and smart metering, examining their strengths, limitations

2.1. ThingsBoard

ThingsBoard is a popular open-source IoT platform that supports real-time device monitoring, data ingestion, and customizable dashboards for energy consumption and voltage analytics. While it provides strong general-purpose IoT capabilities, including scalability and protocol support such as MQTT, it is not specifically designed for detailed smart meter analysis. Its dashboards require technical expertise and lack focused features like monthly outage visualization, power-balance tracking, and simplified user flows. In contrast, our project aims to deliver a more tailored, domain-specific solution optimized for smart meter data interpretation with intuitive, energy-centered analytics.

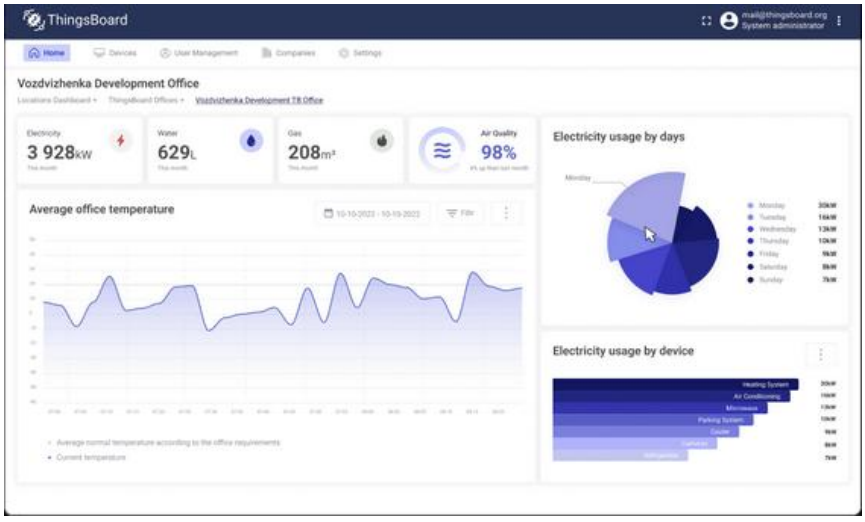

Figure 1 ThingsBoard UI

2.2. Kaa IoT

Kaa IoT offers a commercial smart metering solution with real-time consumption tracking, device management, and enterprise-level dashboards. Although powerful, the platform is proprietary and limits customization, making it less suitable for academic or flexible development needs. Its analytics are broad and enterprise-oriented rather than focused on detailed technical insights like localized outage patterns or custom power-balance computations. Our project seeks to address this gap by developing an open and customizable energy visualization platform that empowers users to perform fine-grained analysis and adapt the system to various deployment environments, from small installations to academic research.

Figure 2 Kaa IoT UI

2.3. OakMeter

OakMeter provides accessible interfaces for viewing electricity usage, peak load alerts, and basic consumption graphs. However, these applications offer limited analytical detail and typically do not include advanced features such as voltage trend analysis, monthly outage summaries, or technical power-balance metrics. They also function as closed ecosystems with little room for developer-level extension. Our project seeks to enhance this space by combining user-friendly visualization with richer analytics and smart meter–specific insights, offering a more flexible and technically robust alternative.

Figure 3 OakMeter UI

Chapter 3 Procedures and Methods

The development of the KU Smart Meter Analytics System will follow a structured and iterative approach to ensure the final product is functional, user-friendly, and reliable. The system will leverage NEA's smart meter API for real-time data acquisition and incorporate an AI model to forecast future electricity usage. An Agile development methodology will be adopted to enable flexibility, continuous improvements, and frequent validation throughout the project lifecycle.

3.1 Development Phases

i. Requirement Analysis

The initial phase will focus on identifying both functional and non-functional system requirements. This will involve conducting surveys and interviews with stakeholders to determine user priorities and expectations. The collected insights will guide decisions on the types of data to be visualized, refresh intervals, and prediction capabilities. Existing AI-based forecasting systems will also be reviewed to support the development of accurate energy consumption prediction models. The scope of the system will be clearly defined at this stage, including expected outcomes such as kWh forecasting and graphical interpretations of energy consumption across different KU blocks.

ii. System Design

During this phase, conceptual designs, wireframes, and UI prototypes will be created to illustrate user interactions and overall workflow. The primary goal will be to design a clean and intuitive dashboard that communicates complex smart meter data effectively. The backend architecture will be engineered to support real-time data synchronization with NEA's API and include secure authentication using OAuth.

Key design considerations include:

a. Seamless synchronization with all smart meters across KU.
b. Backend services capable of data retrieval, processing, and prediction.
c. OAuth-based secure authentication for authorized access.
d. A responsive, mobile-friendly interface that offers clarity without oversimplifying information.

iii. Implementation

This phase will involve translating the system design into a working product using the selected technology stack.

a. Frontend**:**

   The user interface will be developed using React to achieve a responsive, fast, and visually appealing dashboard compatible with multiple device types. The interface will focus on simplicity while presenting detailed insights retrieved from the meter data.

b. Backend**:**

   The backend will be developed using Django for REST API development and server-side operations. MongoDB will be used for efficient data storage and fast retrieval, supporting rapid forecasting computations. Docker will be utilized to containerize backend services, ensuring consistent deployment and scalability.

iv. Testing

A structured testing strategy will be adopted to validate the performance, accuracy, and security of the system.

Testing activities will include:

a. Unit Testing**:** Verifying individual components and functions.
b. Integration Testing**:** Ensuring smooth communication between frontend, backend, and database.
c. User Acceptance Testing (UAT)**:** Testing with real users to assess system usability, accuracy, and reliability.

Both automated and manual testing cycles will be used throughout development to detect and resolve issues early.

v. Documentation

Comprehensive documentation will be maintained throughout the development process. This will include system architecture diagrams, API documentation, deployment instructions, and user manuals. The final report will summarize project goals, methodology, challenges encountered, and outcomes, ensuring the system remains maintainable and extendable in the future.

Chapter 4 System Requirements Specifications

This section outlines the hardware, software, system, functional, and non-functional requirements necessary for the development and operation of the proposed KU Smart Meter Analytics System. These requirements ensure that the system is feasible, scalable, secure, and aligned with stakeholder needs.

4.1. Hardware Requirements

i. Server-Side Hardware (for hosting the system)

- Minimum 4-core processor (Intel Xeon / AMD equivalent)
- 8 - 16 GB RAM (recommended for real-time processing + AI workloads)
- 50 GB SSD storage (expandable as historical data grows)
- Stable network connectivity (100 Mbps recommended for API polling & user access)
- Linux-compatible machine (physical or cloud-based server)

ii. Client-Side Hardware (end-users)

- Any modern device:
  - Desktop / Laptop
  - Tablet
  - Smartphone
- Minimum 2 GB RAM
- Standard web browser support (Chrome, Firefox, Edge)

4.2. Software Requirements

i. Development Software

- Frontend Framework: React (TypeScript)
- Backend Framework: Django (Python)

- Database: MongoDB
- Containerization: Docker & Docker Compose
- Version Control: Git (GitHub)
- Design & Prototyping: Figma (for UI wireframes)

ii. Server-Side Software

- Operating System: Any Linux Distribution (Ubuntu 24.04.3 LTS preferred)
- Web Server: Nginx (reverse proxy + HTTPS)
- SSL Certificate: Let's Encrypt or KU-issued certificate
- Python Environment: Python 3.10+
- Node.js: For building and serving the frontend
- API Communication: HTTPS with OAuth support

iii. Client-Side Software

- Modern web browsers:
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge
- No installation needed (web-based interface)

4.3. System Requirements

i. System Architecture Requirements

- Must follow a client server architecture with separate frontend and backend layers.
- Backend must integrate with NEA Smart Meter API for real-time data acquisition.
- Data processing must support:
  - Validations

    - Cleaning
    - Aggregation
- AI forecasting engines must operate within the backend using trained models.
- Dashboard must provide real-time data visualization and forecasting insights.
- All backend services must be containerized using Docker for portability and easy deployment.

ii. Security Requirements

- OAuth 2.0 authentication for secure system access.
- HTTPS encryption for all transmitted data.
- Secure storage of API keys, tokens, and database credentials.
- Role-based access control for Admin and Standard users.

4.4. Functional Requirements

i. Data Acquisition

- The system shall fetch real-time energy consumption from NEA's Smart Meter API.
- The system shall automatically update data at configurable intervals.
- The system shall support multiple meters across all KU blocks.

ii. Data Storage and Processing

- The system shall store raw, historical, and processed data in MongoDB.
- The system shall clean and preprocess data before generating visualizations.
- The system shall provide aggregated data (hourly, daily, weekly).

iii. AI & Forecasting

- The system shall use an AI/ML model to predict electricity consumption.
- Forecast results must include future usage patterns and peak hours.
- The AI model shall be updatable with new data.

iv. Dashboard & Visualization

- The system shall provide a responsive web dashboard with:
  - Real-time usage
  - Historical trends
  - Forecasts
  - Block-level comparisons
- Users shall filter data by date, meter, and KU block.
- Users shall export data/graphs in CSV or PDF format.

v. Authentication & Access Control

- The system shall use OAuth for secure login.
- Admins shall manage system settings and AI model updates.
- Standard users shall only view consumption and forecasting data.

vi. System Interactions & APIs

- The backend shall expose REST APIs for frontend communication.
- The system shall log API failures, invalid data, and access violations.

vii. Testing & Quality Assurance

- The system shall undergo unit, integration, and UAT testing.
- Testing shall validate accuracy, usability, and forecast reliability.

viii. Documentation

- User manuals, architecture diagrams, and API documentation shall be provided.
- A deployment guide shall accompany the final system.

Chapter 5 Project Planning and Scheduling

5.1. Tasks

The Gantt chart provided in Figure 4 outlines the timeline for various tasks involved in project development, giving a clear picture of time allocation across different phases. This chart allows for structured planning by visually mapping out each task's start and end dates, contributing to an organized project schedule.

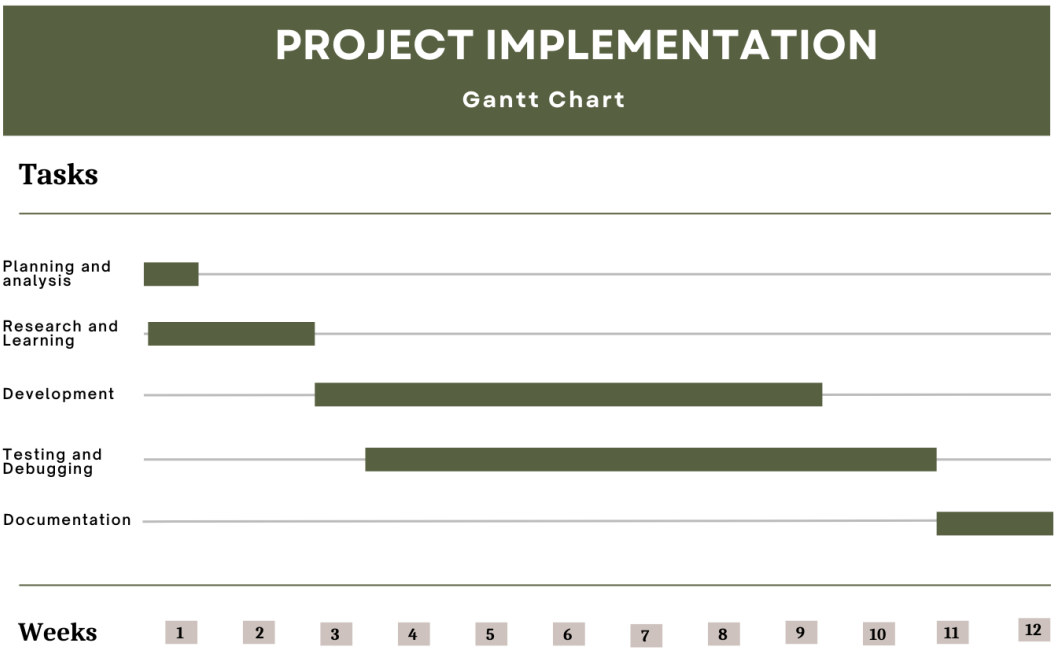

Figure 4 Gantt Chat

Tasks and Phases

i. **Planning and Analysis:** Initial stage focused on problem identification, requirement gathering, and setting project objectives. This phase is scheduled for the first week to ensure a clear project direction.

ii. **Research and Learning:** Occurs early in the project, spanning weeks 2 to 3, to gather necessary insights and knowledge related to the project scope.

iii. **Development:** The core phase of the project, stretching from week 4 to week 8, where the actual work on the project deliverables takes place.

iv. **Testing and Debugging:** Starts in week 5 and continue alongside development, extending to week 11 to identify and resolve any issues in the project.

v. **Documentation:** Final stage, occurring in week 12, to compile all findings, processes, and outcomes, ensuring a complete and accurate project record.

The Gantt chart structure provides a roadmap that helps monitor project progress and allocate resources efficiently. With a breakdown of terminal and summary elements, this visual tool aids in meeting project milestones and managing time effectively.

References

ThingsBoard. (n.d.). ThingsBoard: Open-source IoT platform. Retrieved from https://thingsboard.io/

KaaIoT Technologies. (n.d.). Kaa IoT platform documentation and smart metering solutions. Retrieved from https://www.kaaiot.com/

Oakter. (n.d.). OakMeter smart energy monitoring system. Retrieved from https://oakter.com/

NEA. (2023). Nepal Electricity Authority – Smart Metering Program Overview. Nepal Electricity Authority. https://www.nea.org.np/

MongoDB. (2023). MongoDB architecture and use cases. MongoDB Inc. https://www.mongodb.com/

Python Software Foundation. (2023). Python documentation. https://www.python.org/doc/

React. (2023). React documentation. Meta Platforms Inc. https://react.dev