# Electrical Energy Monitoring and Visualization

*Submitted in partial fulfillment of the requirements for the degree of*

**Bachelor of Technology**

in

**Electrical and Electronics Engineering**

*By*

**Sham Ganesh K (21BEE0024)**

**Thavanesh R (21BEE0080)**

**Yuvan Shankar M (21BEE0373)**

**Under the guidance of**

**Dr. Meikandasivam S**

**School of Electrical Engineering,**

**VIT, Vellore.**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

April,2025

# DECLARATION

We here by declare that the thesis entitled "**Electrical Energy Monitoring and Visualization**" submitted by us, for the award of the degree of *Bachelor of Technology in Electrical and Electronics Engineering* to VIT is a record of Bonafide work carried out by me under the supervision of **Dr. Meikandasivam S**, Professor, School of Electrical Engineering, VIT, Vellore.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

[illegible]

[illegible]

[illegible]

Place: Vellore

Date: 20/04/2025

**Signature of the Candidates**

# CERTIFICATE

This is to certify that the thesis entitled "**Electrical Energy Monitoring and Visualization**" submitted by **Sham Ganesh k(21BEE0024)**, **Thavanesh R(21BEE0080), Yuvan Shankar M(21BEE0373)** School of Electrical Engineering, VIT, Vellore, for the award of the degree of *Bachelor of Technology in Electrical and Electronics Engineering*, is a record of Bonafide work carried out by them under my supervision during the period, 16.12.2024 to 17.04.2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and, in my opinion, meets the necessary standards for submission.

Place: Vellore

Date: 20/04/2025

**Signature of the Guide**

**Dr. Meikandasivam S**

**The thesis is satisfactory/unsatisfactory**

Approved by

Head of the Department

EEE

Dean, SELECT

# Acknowledgements

With immense pleasure and a deep sense of gratitude, we wish to express our sincere thanks to my supervisor, **Dr. Meikandasivam S**, Professor, School of Electrical Engineering, VIT, without his/her motivation and continuous encouragement, this project work would not have been successfully completed.

We are grateful to the Chancellor of VIT, **Dr. G. Viswanathan**, the Vice Presidents, and the Vice Chancellor for motivating me to carry out research at VIT and also for providing me with infrastructural facilities and many other resources needed for my project.

We express our sincere thanks to our dean, **Dr. Kowsalya M,** and former dean, **Dr. Mathew M Noel**, School of Electrical Engineering, VIT, for his kind words of support and encouragement. We want to acknowledge the support rendered by **my colleagues** in several ways through our project work.

We wish to extend our profound sense of gratitude to **our parents** for all the sacrifices they made during our research, and also for providing us with moral support and encouragement whenever required.

Last but not least, we would like to thank the lab in charge, **Venkat Raman,** for their constant encouragement and moral support, along with patience and understanding.

Place: Vellore

Date: 20/04/2025 **Student's Name**

**Sham Ganesh K 21BEE0024**
**Thavanesh R 21BEE0080**
**Yuvan Shankar M 21BEE0373**

# Executive Summary

This project presents the design and implementation of a real-time monitoring and visualization web application for electrical parameters across multiple substations within the Vellore Institute of Technology (VIT) Vellore campus. With the growing demand for efficient and intelligent energy management, the project focuses on enabling continuous, centralized monitoring of critical electrical data using smart technologies.

Smart energy meters with integrated UART (Universal Asynchronous Receiver/Transmitter) communication protocols are installed at each substation to measure key parameters such as voltage, current, power factor, frequency, active power, and total energy consumption. These meters are connected to serial-to-WiFi converters, which transmit the measured data wirelessly to a central processing unit. A Python-based backend script is used to acquire, decode, and process the incoming serial data in real-time.

To ensure effective visualization and accessibility, dedicated web-based dashboards are developed using Streamlit, a powerful open-source Python framework. Each transformer is assigned its own dashboard, where live data is displayed through interactive visual elements including real-time charts, gauges, and numerical indicators. These dashboards are designed to be user-friendly, allowing for instant observation of transformer performance and prompt detection of any anomalies.The primary goals of this project are to improve energy usage awareness, support timely maintenance, and contribute to the development of a smart energy monitoring infrastructure. The website is scalable, low-cost, and can be extended to other buildings or institutions, making it a valuable solution for modern electrical management needs.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF TERMS AND ABBREVIATIONS

CHAPTER 1

# INTRODUCTION

## 1.1 Objective

The objective of this project is to design and develop a web-based platform for monitoring and visualizing the electrical parameters of VIT Vellore substations. The project aims to provide real-time data insights into the energy usage of each powerhouse, enabling efficient energy management and performance analysis.

• Data Visualization: To present electrical parameters such as power, energy, voltage, current, frequency, and power factor through dynamic and interactive visualizations.

• Seamless User Experience: To create an intuitive, user-friendly interface using Streamlit, where users can easily navigate between different substations and view energy usage details.

• Data Integration: To fetch data stored in a MySQL database, which is organized by transformer and stored in monthly tables, and display it effectively on the web page.

• Real-Time Monitoring: To enable continuous tracking of energy usage for all powerhouses and transformers, allowing for timely decision-making and optimization of electrical resources.

## 1.2 Motivation

Increasing focus on energy efficiency and sustainability has made it essential for organizations, including educational institutions like VIT (Vellore Institute of Technology), to adopt effective power management systems. With rising energy consumption and the need for better resource management, it became clear that real-time energy monitoring and analysis could offer significant benefits. Our motivation behind developing this project was to enable efficient energy usage and provide a comprehensive solution for monitoring the electrical parameters of VIT’s substations.

We aimed to create a website that would not only enhance the visibility of energy usage but also allow quick identification of inefficiencies and potential issues in the electrical grid. Real-time data access and visualization of key parameters such as power, voltage, and frequency were crucial in enabling facility managers to make informed decisions and respond swiftly to any irregularities. Additionally, by simplifying the process of accessing and interpreting the collected data, we hoped to bridge the gap between complex technical information and actionable insights. Ultimately, our motivation was to contribute to the broader goal of energy sustainability, offering a tool that helps reduce energy wastage, improve operational efficiency, and support the institution's sustainability initiatives.

## 1.3 Background

The project aims to address the need for efficient and real-time monitoring of energy consumption across VIT’s substations. With the growing demand for energy and the challenges in tracking consumption, manual monitoring methods became inefficient. The goal was to automate the process of collecting energy data from smart meters and storing it in a centralized database. By developing a web-based interface to visualize this data, the project seeks to help the management team monitor and optimize energy usage effectively, leading to better decision-making and energy conservation.

CHAPTER 2

# PROJECT DESCRIPTION AND GOALS

## 2.1 Review of Literature

The integration of energy management website and real-time monitoring has become an area of significant interest in the fields of industrial automation and smart grid technology. Studies have explored various aspects of energy monitoring, including the implementation of smart meters, data collection, and the use of web-based platforms for visualization. These advancements play a pivotal role in improving energy efficiency, reducing consumption, and enabling better decision-making in energy management.

In the context of energy consumption monitoring, IoT-based smart meters have been widely adopted due to their ability to provide real-time data, which is crucial for optimizing energy use. According to Sharma et al. (2019), smart meters offer benefits such as remote monitoring, precise measurement of electrical parameters, and enhanced reliability compared to conventional metering systems. Their study also highlights the importance of integrating these devices with centralized data storage and analytics platforms to improve operational efficiency and forecasting.

Furthermore, the role of data visualization in energy management has garnered significant attention. Gupta et al. (2021) emphasized that platforms like Streamlit, which enable rapid development of interactive web applications, can be effectively utilized for creating dashboards that display energy consumption data. These platforms facilitate real-time analysis and allow users to visualize complex datasets in an easily interpretable manner. Their research suggests that such platforms are ideal for energy management applications where immediate data insights are needed for decision-making.

The importance of efficient data storage and management also cannot be overstated. Lee (2018) explains that databases such as MySQL and platforms like PHPMyAdmin are extensively used in large-scale data management systems. These databases allow for effective storage, querying, and retrieval of time-series data, which is particularly valuable for applications in energy management where historical data is essential for analysis and pattern recognition.

Moreover, integrating smart meters with web-based platforms enables remote monitoring of energy consumption, which aligns with the increasing demand for intelligent energy systems in modern infrastructure. Research by Singh et al. (2020) highlights how such integrations enhance user convenience and reduce the need for manual intervention, leading to greater automation and more reliable energy management.

In summary, the literature underscores the significant advancements in smart metering technology, real-time data collection, and web-based visualization tools for energy management. These developments have influenced the current project, which aims to leverage similar technologies for creating a user-friendly web platform to visualize and analyze energy consumption data from VIT's substations. The combination of IoT-enabled meters, efficient data management systems, and interactive visualization platforms offers a promising approach to optimizing energy use in academic and industrial settings.

## 2.2 Project Description

The project focuses on the development of a web-based platform for real-time visualization and monitoring of energy consumption data from the smart meters installed at various substations across the VIT campus. The goal is to create an intuitive and interactive interface where users can easily access, visualize, and analyze the electrical parameters such as power, energy, voltage, current, frequency, and power factor. This website is built to enhance the management of energy consumption, providing a more efficient way to monitor the operational status of each substation and enable proactive decision-making for energy optimization.

The project involves two main components: data collection and storage, and the development of the web interface for data visualization. While the data collection and storage part has already been handled by senior team members, we focused on the development of the web page that interfaces with the stored data. The data is collected from multiple smart meters connected to transformers via a serial-to-WiFi converter, and stored in a MySQL database through XAMPP. The database stores data on a per-minute basis, with separate tables for each month, allowing for easy management and analysis of historical energy usage data.

The web page, developed using Streamlit, allows users to visualize the energy consumption data for each powerhouse in a user-friendly manner. The homepage of the web interface displays the six powerhouses of VIT, with each powerhouse showing its corresponding energy usage. Users can select a specific powerhouse to navigate to a detailed page that displays the electrical parameters for the respective transformer. The interface allows for easy interaction with the data, providing a comprehensive view of the energy consumption patterns across different substations.

The primary objective of this project is to provide a real-time, accessible solution for energy monitoring, empowering users to track energy consumption patterns, detect anomalies, and make informed decisions regarding energy management. By developing this platform, the project aims to contribute to the efficient use of energy resources across the VIT campus, ultimately promoting sustainability and reducing operational costs associated with energy consumption.

In summary, this project combines the benefits of IoT-enabled smart meters, centralized data storage, and a dynamic web-based interface to create a powerful tool for energy management. The project not only provides real-time insights into energy usage but also offers historical data analysis, helping stakeholders identify trends and make data-driven decisions to optimize energy efficiency.

### 2.2.1 Software

The software aspect of the project is a combination of Python programming and Streamlit for real-time data visualization. The Python code serves as the backend, which handles the communication between the serial-to-WiFi converter and the central server. It is responsible for reading and decoding the incoming serial data from the energy meters, processing it, and extracting the relevant electrical parameters such as voltage, current, power factor, and frequency. After processing, the data is formatted for visualization, which is then displayed on a Streamlit-based dashboard. The interactive web interface is designed to present the data in an easily understandable format with real-time charts, numerical indicators, and graphical representations. The use of Python libraries such as Matplotlib and Plotly is leveraged for advanced data visualization. The software is built to be flexible and scalable, allowing easy integration of additional substations and smart meters as the project evolves.

### 2.2.2 Hardware

The hardware components of the project include smart energy meters, serial-to-WiFi converters, a microcontroller or server for data management, and a display unit for visualization. The smart energy meters are responsible for measuring various electrical parameters like voltage, current, power factor, active power, and energy consumption. These meters are equipped with UART communication ports, which enable the transmission of measurement data. The serial-to-WiFi converter acts as an intermediary device that converts the UART signals from the energy meters into a Wi-Fi signal, allowing wireless data transmission to a central server. The central server, which could be based on a microcontroller such as an ESP32 or a dedicated computer, processes the incoming data and sends it to the Streamlit-based dashboard. The dashboard provides real-time visual feedback of the electrical parameters. The entire project is designed for ease of installation and scalability, making it ideal for monitoring multiple substations on the campus.

## 2.3 Goals

The main goals of this project are to design and implement a comprehensive energy monitoring that provides real-time insights into the electrical parameters of substations across VIT. The project aims to achieve real-time data acquisition and visualization of key electrical metrics using smart energy meters and wireless communication. The wireless communication architecture, facilitated by serial-to- WiFi converters, ensures seamless data transmission without the need for complex wired infrastructure. The Python-based backend will decode and process the data in real-time, ensuring timely updates on the Streamlit dashboard. A major goal of the project is to create a scalable infrastructure that can accommodate more substations in the future, ensuring long-term usability. Additionally, the website is intended to help detect faults and optimize energy usage by providing real-time monitoring of parameters like power factor and voltage. This continuous monitoring will enable preventive maintenance and reduce the risk of major electrical failures. Another key goal is to design a user-friendly interface through the use of Streamlit, ensuring that both technical and non-technical users can easily access and interpret the data for better decision-making.

TABLE 2.1. TRADITIONAL VS IOT-BASED ENERGY MONITORING

| S. No | Traditional Energy Monitoring | IoT-Based Real-Time Energy Monitoring (Our Project) |
|---|---|---|
| **1** | Manual data logging and periodic reporting | Automated data collection and real-time updates |
| **2** | No immediate alert system for anomalies | Real-time alerts and notifications for anomalies and faults |
| **3** | Limited data accessibility (local systems only) | Remote access via cloud-based dashboards |
| **4** | Requires frequent manpower for meter readings | Minimizes human intervention through continuous automation |
| **5** | No historical trend analysis or visualization | Offers dynamic charts and historical trend analysis |
| **6** | Time delays in identifying power quality issues | Instant identification of power factor, voltage, frequency issues |
| **7** | Difficult to integrate with modern data platforms | Easily integrates with web apps like Streamlit and databases |
| **8** | Higher operational costs over time | Cost-effective and scalable over long-term use |

CHAPTER 3

# TECHNICAL SPECIFICATION

## 3.1 Software Specification

The software for this project consists of two main components: data acquisition and real-time data visualization. The data acquisition module is written in Python and is responsible for receiving data from the smart energy meters via the serial-to-WiFi converter. This module utilizes PySerial for reading data from the UART communication protocol, allowing seamless data transfer from the meters. The data is then parsed and processed to extract the required electrical parameters such as voltage, current, power factor, frequency, active power, and energy consumption.

The real-time visualization component is built using Streamlit, a Python-based framework that allows the creation of interactive, web-based dashboards. These dashboards present the processed data in a user-friendly manner, including dynamic charts and graphs that update live as new data is received. In addition, the backend Python code utilizes libraries such as Matplotlib and Plotly for advanced visualizations. The software is designed to run on a server or microcontroller, with the capacity to scale as more substations are added to the network.

## 3.2 Hardware Specification

The hardware components required for this project include energy meters, serial-to-WiFi converters, microcontrollers, and display units. These components work together to ensure accurate data collection, transmission, and visualization.

### 3.2.1 Quotation and Reference to Earlier Work

The hardware selection for this project is based on established practices in energy monitoring systems. According to Sharma et al. (2020), smart energy meters with UART interfaces have been widely used for monitoring electrical parameters in industrial and institutional environments. In their work, they demonstrated how serial-to-WiFi converters facilitate the wireless transmission of data, allowing for real-time monitoring of energy consumption without the need for complex wiring.

In our project, the energy meters will be sourced from reputable manufacturers who provide accurate measurements and reliable communication interfaces. The serial-to-WiFi converter will ensure that the data from the energy meters is transmitted wirelessly to the central server or microcontroller for processing and visualization. The selection of hardware will prioritize low cost, reliability, and scalability.

### 3.2.2 Use of Abbreviations

UART: Universal Asynchronous Receiver/Transmitter

Wi-Fi: Wireless Fidelity

API: Application Programming Interface

TCP/IP: Transmission Control Protocol/Internet Protocol

These abbreviations will be used consistently throughout the document to ensure clarity and precision.

CHAPTER 4

# DESIGN APPROACH AND DETAILS

## 4.1 Design Approach / Materials and Methods

The design of the web-based visualization platform for electrical parameter monitoring was guided by the need for a user-friendly interface, efficient data handling, and real- time responsiveness. Our primary goal was to enable seamless access to transformer- level energy usage data for each powerhouse within the VIT campus. Since the data collection and storage architecture had already been implemented by our seniors, our task focused on developing the front-end interface and integrating it with the existing database infrastructure.

**System Architecture**

The overall architecture consists of three key layers:

1. Data Layer – The MySQL database, hosted locally through XAMPP, stores minute-wise data from each smart meter in monthly databases. Each meter's data is stored in an individual table.

2. Processing Layer – Python, integrated with the Streamlit framework, acts as the processing unit. It handles querying of the dynamic database, filtering and formatting the data, and preparing it for display.

3. Presentation Layer – The Streamlit-based web interface displays summarized energy data, graphical trends, and parameter readings in an interactive format.

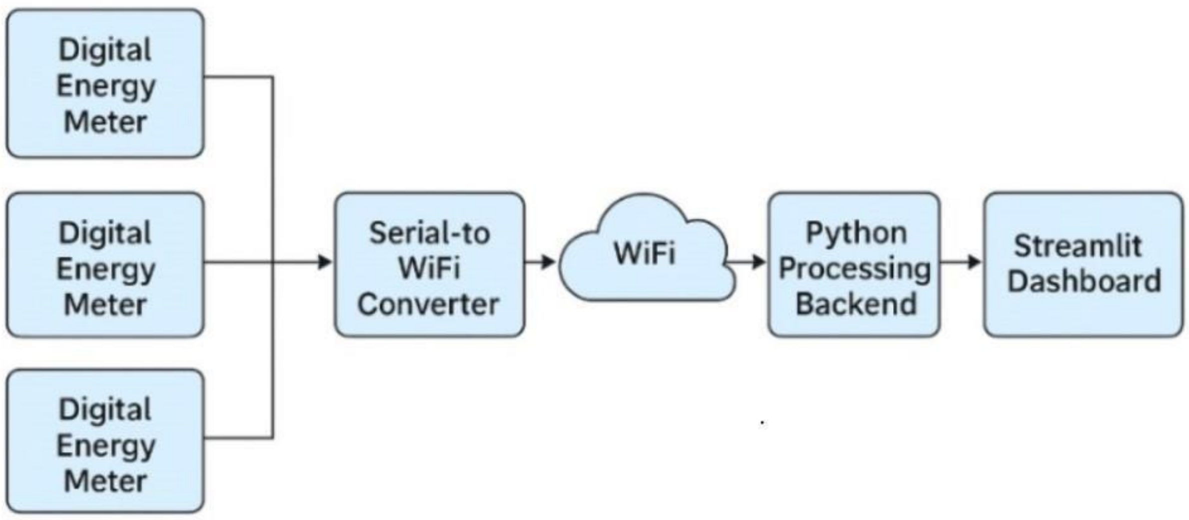


**Fig 4.1 System Architecture**

**Design Flow**

1. **Data Retrieval Logic:**
   Each powerhouse is associated with a unique transformer, and each transformer has a dedicated table in the MySQL database. Depending on the current month, the system dynamically connects to the corresponding database (e.g., april2025, March2025) and fetches data from all relevant tables (meitest1 to meitest13).

2. **Front Page Interface:**
   The home page provides a visual overview of all six powerhouses at VIT. Each card displays the name of the powerhouse and its total energy consumption for the day. Upon clicking, users are redirected to a detailed view for the selected powerhouse.

3. **Detailed Visualization Page:**
   Once a powerhouse is selected, the webpage displays individual electrical parameters including voltage, current, power, frequency, energy, and power factor. These are rendered using interactive charts and plots, enabling users to analyze consumption trends throughout the day.

4. **Data Handling and Refresh:**
   Data is refreshed every minute, ensuring the platform reflects near real-time usage statistics. Streamlit automatically updates graphs and values without manual page reloads, contributing to a smooth user experience.

**User Interface Considerations**

- Simplicity and clarity were prioritized to ensure accessibility even for non-technical users.
- Charts are color-coded and labeled for quick identification of trends.
- Navigation is minimal and intuitive, allowing users to move between overview and detailed views with ease.

**Scalability**

The modular design of our web application ensures that new transformers or powerhouses can be added with minimal changes. As long as the data follows the existing schema, new tables can be integrated through dynamic querying logic in Python.

## 4.2 Codes and Standards

- In **IEEE Standard 1547-2018** establishes criteria for the interconnection of distributed energy resources (DER) with electric power systems, ensuring that the monitoring webpage can effectively integrate with existing grid infrastructures and facilitate reliable data exchange.
- **IEEE 2030.5**, also known as *Smart Energy Profile 2.0*, defines communication protocols for smart energy devices, enabling real-time data exchange between the energy monitoring website and utility management systems, and supporting functionalities such as demand response and distributed energy resource management.
- **IEEE C37.118** specifies requirements for synchrophasor measurements used in power systems, ensuring accurate and time-synchronized measurements of electrical parameters, which are essential for monitoring and analysis performance and detecting anomalies.
- **IEEE 1159** focuses on monitoring electric power quality disturbances, including voltage sags, swells, interruptions, and frequency deviations. Implementing this

standard ensures that VIT’s system accurately tracks and flags abnormal voltage and frequency conditions, enhancing system reliability.

- **IEEE 519** provides guidelines for maintaining acceptable harmonic levels in power systems. While harmonics were not the primary focus of this project, following IEEE 519 principles ensures better quality data when monitoring parameters like voltage and frequency over Wi-Fi transmission.

## 4.3 Constraints, Alternatives and Tradeoffs

### 4.3.1 Constraints

One of the primary constraints is network dependency. Since data transfer occurs over Wi-Fi, any network instability can disrupt real-time data collection. Additionally, the limited memory and processing capabilities of microcontrollers like ESP32 pose challenges when handling multiple meter inputs simultaneously. Power supply interruptions and limited access to substations during working hours also constrained the hardware installation process. Furthermore, data accuracy depends heavily on the reliability of the energy meters and the serial communication interface.

### 4.3.2 Alternatives

To overcome the network-related constraints, alternatives such as LoRa-based or GSM-based communication systems were considered. These could allow for more stable and longer-range communication, especially in areas with poor Wi-Fi coverage. On the software side, platforms like Flask or Django were considered for web visualization but were ultimately replaced by Streamlit due to its ease of use and quick deployment capabilities.

### 4.3.3 Tradeoffs

A To balance cost and performance, standard digital meters were paired with external Wi-Fi modules instead of expensive smart meters. Streamlit was chosen over complex frameworks like Django to speed up development, sacrificing advanced features like authentication. Centralized data collection simplified management but introduced a single point of failure risk.

CHAPTER 5

# SCHEDULE, TASKS AND MILESTONES

## 5.1 Schedule

The project is planned over a six-month period to ensure comprehensive development and deployment of the real-time energy monitoring and analysis. In the first month, the focus will be on planning and design, including an extensive literature review, finalization of website requirements, selection of hardware components such as smart meters, serial-to-WiFi modules, and an ESP32 microcontroller, and drafting the overall system architecture. During the second month, hardware setup and prototyping will be carried out: energy meters will be installed and interfaced via UART to the Wi-Fi converter, and basic wireless connectivity between at least one meter and the central server will be validated. The third month is dedicated to backend development, wherein Python scripts using PySerial will be implemented and tested to reliably acquire, decode, and buffer electrical parameters. In the fourth month, the Streamlit web application will be designed and built; this includes integrating the Python processing backend with dynamic charts, gauges, and numerical displays, and refining the user interface for clarity and responsiveness. Month five will see full system integration and rigorous testing, scaling the solution across multiple substations and evaluating performance under varying network conditions to address any bottlenecks or data inconsistencies. Finally, in the sixth month, the website will undergo validation and deployment across all targeted transformers, followed by user acceptance testing, reliability optimization, and the preparation of comprehensive documentation— comprising user guides, maintenance procedures, and the final project report.

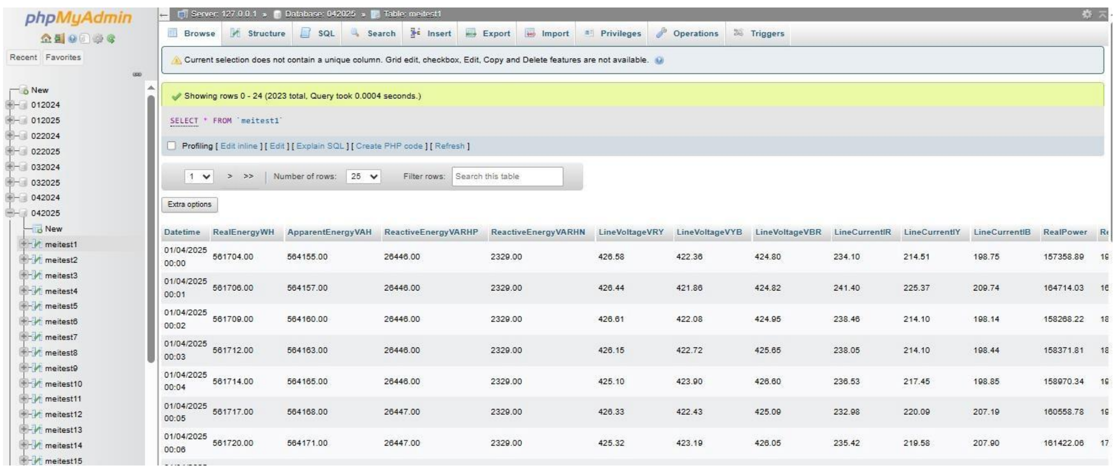
## 5.2 Tasks and Milestones

The Progress will be tracked through a series of key tasks, each tied to a specific milestone to confirm readiness for the next phase. By the end of the first month, the requirements specification and system architecture will be formally approved, marking the completion of the planning and design phase. The second month's milestone is reached when reliable wireless data transmission from at least one energy meter to the server has been demonstrated, confirming that the hardware assembly and connectivity are functioning correctly. In the third month, successful real-time acquisition of accurate electrical parameters from a test meter will indicate that the backend data acquisition scripts are operational. The fourth month culminates in a fully functional Streamlit dashboard that displays live data from a single transformer, validating the visualization and interface development. The fifth-month milestone is the stable monitoring of multiple transformers with minimal data loss, verifying that the website scales effectively and withstands network variability. Finally, the project is signed off in the sixth month when the solution is deployed campus-wide, user feedback has been incorporated, and all documentation is completed and submitted, signifying the successful closure of the project.

CHAPTER 6

# PROJECT DEMONSTRATION

The developed project was demonstrated through a fully functional and interactive web application that visualizes real-time and historical electrical parameters from multiple smart meters installed across the VIT campus. The core of the demonstration centered around the user interface designed using the **Streamlit** Python library, which integrates seamlessly with the backend SQL database to fetch and present the data effectively.



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

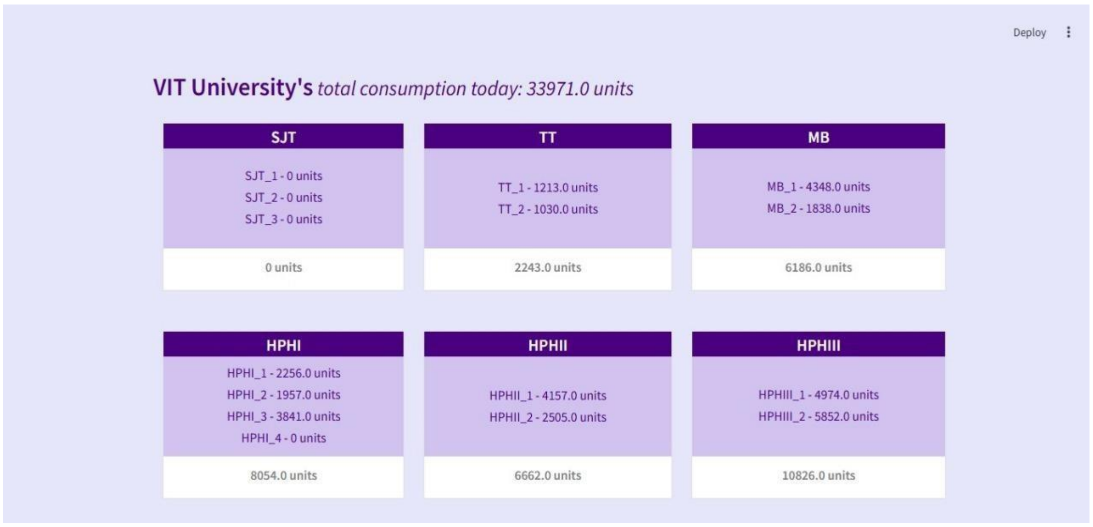
Showing rows 0 - 24 (2023 total, Query took 0.0004 seconds.)

```
SELECT * FROM `meitest1`
```

| Datetime | RealEnergyWH | ApparentEnergyVAH | ReactiveEnergyVARHP | ReactiveEnergyVARHN | LineVoltageVRY | LineVoltageVYB | LineVoltageVBR | LineCurrentIR | LineCurrentIY | LineCurrentIB | RealPower | Re |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01/04/2025 00:00 | 561704.00 | 564155.00 | 26446.00 | 2329.00 | 426.58 | 422.36 | 424.80 | 234.10 | 214.51 | 198.75 | 157358.89 | 19 |
| 01/04/2025 00:01 | 561706.00 | 564157.00 | 26446.00 | 2329.00 | 426.44 | 421.86 | 424.82 | 241.40 | 225.37 | 209.74 | 164714.03 | 16 |
| 01/04/2025 00:02 | 561709.00 | 564160.00 | 26446.00 | 2329.00 | 426.61 | 422.08 | 424.95 | 238.46 | 214.10 | 198.14 | 158268.22 | 18 |
| 01/04/2025 00:03 | 561712.00 | 564163.00 | 26446.00 | 2329.00 | 426.15 | 422.72 | 425.65 | 238.05 | 214.10 | 198.44 | 158371.81 | 18 |
| 01/04/2025 00:04 | 561714.00 | 564165.00 | 26446.00 | 2329.00 | 425.10 | 423.90 | 426.60 | 236.53 | 217.45 | 198.85 | 158970.34 | 19 |
| 01/04/2025 00:05 | 561717.00 | 564168.00 | 26447.00 | 2329.00 | 426.33 | 422.43 | 425.09 | 232.98 | 220.09 | 207.19 | 160558.78 | 19 |
| 01/04/2025 00:06 | 561720.00 | 564171.00 | 26447.00 | 2329.00 | 425.32 | 423.19 | 426.05 | 235.42 | 219.58 | 207.90 | 161422.06 | 17 |

Fig. 6.1 SQL Database

In the context of the real-time energy monitoring website implemented across various substations in VIT, the SQL database shown in Figure. 6.1 plays a pivotal role as the centralized data management layer. This database serves as the backbone for storing all real-time electrical parameters captured by UART-enabled smart energy meters. These meters, strategically placed at each transformer or substation, continuously measure and transmit critical parameters such as real energy (kWh), apparent energy (kVAh), reactive energy (kVARh), voltage levels across the three phases (R, Y, B), and corresponding line currents. The data transmission from each meter occurs over a serial-to-WiFi interface, enabling wireless communication with a local server or cloud-based system where Python scripts process and push the data into structured SQL tables.


Deploy

**VIT University's** *total consumption today: 33971.0 units*

**SJT**
SJT_1 - 0 units
SJT_2 - 0 units
SJT_3 - 0 units
0 units

**TT**
TT_1 - 1213.0 units
TT_2 - 1030.0 units
2243.0 units

**MB**
MB_1 - 4348.0 units
MB_2 - 1838.0 units
6186.0 units

**HPHI**
HPHI_1 - 2256.0 units
HPHI_2 - 1957.0 units
HPHI_3 - 3841.0 units
HPHI_4 - 0 units
8054.0 units

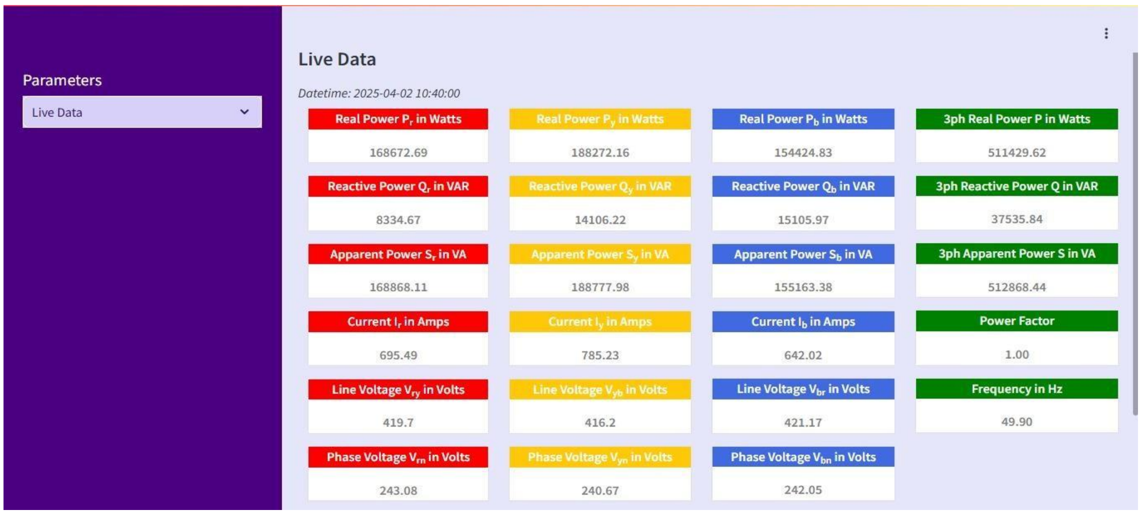
**HPHII**
HPHII_1 - 4157.0 units
HPHII_2 - 2505.0 units
6662.0 units

**HPHIII**
HPHIII_1 - 4974.0 units
HPHIII_2 - 5852.0 units
10826.0 units

Fig. 6.2 Main page of the website

At the homepage, Fig.6.2 of the web application, users are presented with a dashboard showcasing the energy usage of the six main substations (powerhouses) of the campus. Each substation is displayed as a selectable card or button, allowing the user to navigate to a more detailed page corresponding to the selected powerhouse.

Upon selection, corresponding to the chosen transformer, and retrieves the electrical parameters. These include **energy consumption, power, voltage, current, frequency**, and **power factor**. The data is then displayed using intuitive and easy-to-read visualizations such as bar graphs, line plots, and KPIs.



Live Data

*Datetime: 2025-04-02 10:40:00*

| | | | |
|---|---|---|---|
| Real Power $P_r$ in Watts | Real Power $P_y$ in Watts | Real Power $P_b$ in Watts | 3ph Real Power P in Watts |
| 168672.69 | 188272.16 | 154424.83 | 511429.62 |
| Reactive Power $Q_r$ in VAR | Reactive Power $Q_y$ in VAR | Reactive Power $Q_b$ in VAR | 3ph Reactive Power Q in VAR |
| 8334.67 | 14106.22 | 15105.97 | 37535.84 |
| Apparent Power $S_r$ in VA | Apparent Power $S_y$ in VA | Apparent Power $S_b$ in VA | 3ph Apparent Power S in VA |
| 168868.11 | 188777.98 | 155163.38 | 512868.44 |
| Current $I_r$ in Amps | Current $I_y$ in Amps | Current $I_b$ in Amps | Power Factor |
| 695.49 | 785.23 | 642.02 | 1.00 |
| Line Voltage $V_{ry}$ in Volts | Line Voltage $V_{yb}$ in Volts | Line Voltage $V_{br}$ in Volts | Frequency in Hz |
| 419.7 | 416.2 | 421.17 | 49.90 |
| Phase Voltage $V_{rn}$ in Volts | Phase Voltage $V_{yn}$ in Volts | Phase Voltage $V_{bn}$ in Volts | |
| 243.08 | 240.67 | 242.05 | |

Fig. 6.3 SJT Transformer – 1 Page Live data

This image, Fig. 6.3, represents the real-time energy monitoring dashboard developed using Streamlit as part of the project. It visually displays critical electrical parameters such as real power (P), reactive power (Q), apparent power (S), current (I), voltage (V), power factor, and frequency across all three phases (R, Y, B) in a clear, color-coded format. The data is collected from smart energy meters installed at various VIT substations and transmitted via a serial-to-WiFi module to a central server. The dashboard updates dynamically, allowing users, such as facilities staff and energy auditors—to observe system performance, identify abnormalities, and take immediate corrective actions. This interface plays a vital role in promoting energy efficiency, ensuring system transparency, and facilitating data-driven energy management within the institution.

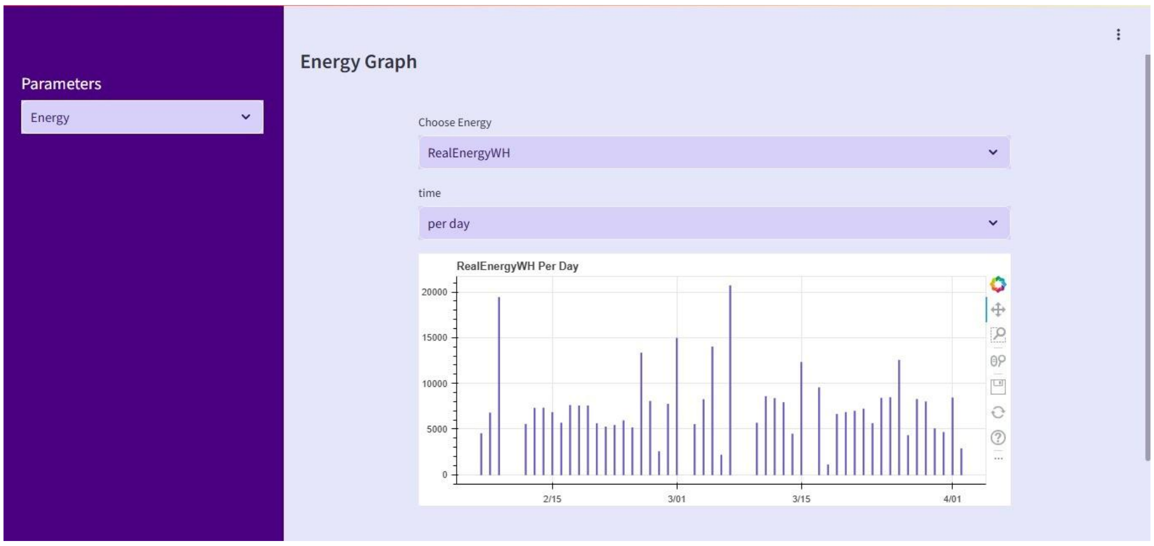


Fig. 6.4 SJT Transformer – 1 Energy Graph for per day

This image figure. 6.4 and 6.5 represent the energy analytics dashboard designed as part of our real-time energy monitoring and analysis website implemented across VIT's substations. Specifically, it visualizes the monthly and daily Real Energy consumption (Wh) over a selected time period. The dashboard provides users the flexibility to choose different energy parameters and time intervals (e.g., per day, per month), enabling customizable insights into consumption trends.

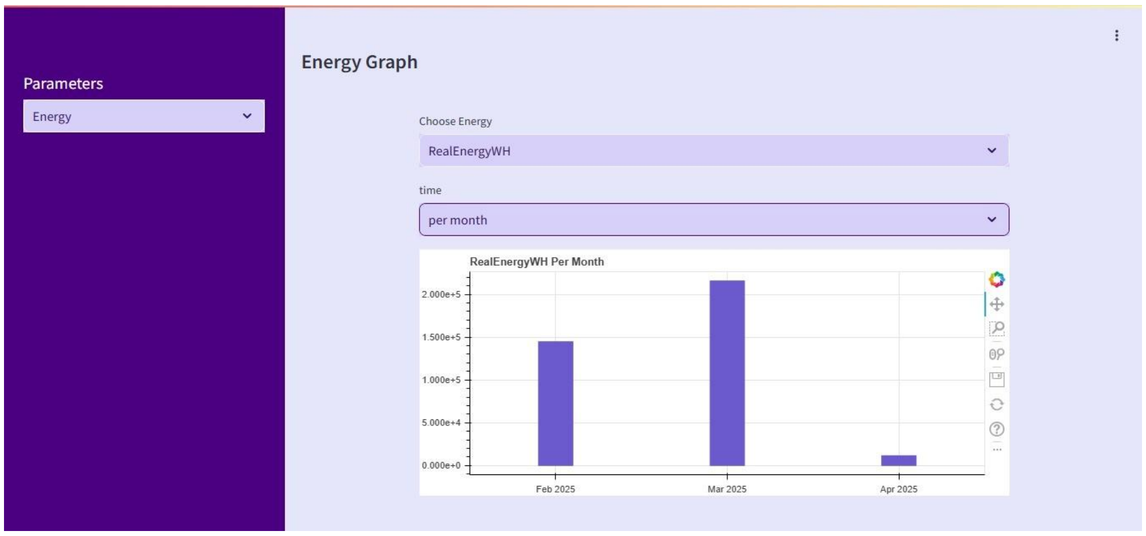
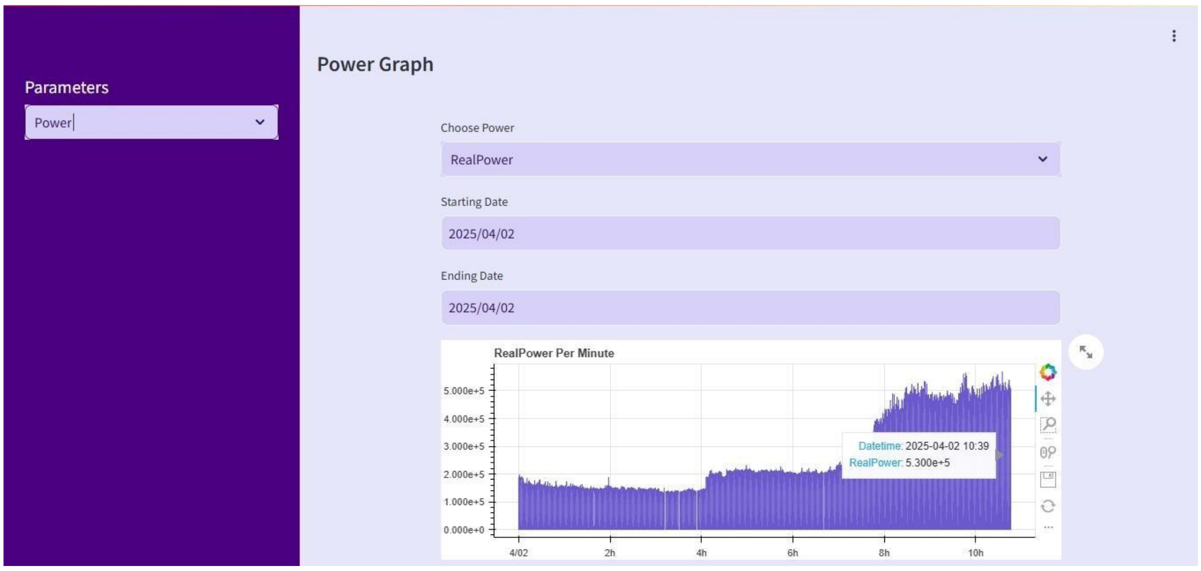


Fig. 6.5 SJT Transformer – 1 Energy Graph for per month

The graph displays real-time energy data collected from UART-enabled smart energy meters, processed using Python, and rendered via Streamlit for interactive visualization. These daily consumption bars help identify peak usage periods, inefficiencies, and energy- saving opportunities.

Such a graphical interface supports energy audits and load forecasting by presenting historical energy patterns in a clear and actionable format. This feature is essential for decision-makers and facility managers aiming to improve operational efficiency, manage loads better, and work toward sustainability goals through data-driven insights.


Parameters
Power

Power Graph

Choose Power
RealPower

Starting Date
2025/04/02

Ending Date
2025/04/02

Fig. 6.6 SJT Transformer – 1 Power Graph

The power graph show in Fig. 6.6 how real power is used throughout a selected day. Users can choose the type of power they want to see (like Real Power) and select a start and end date. In this example, the graph displays the Real Power used on April 2, 2025, with values shown for every minute.

The X-axis represents the time of day, while the Y-axis shows the amount of power used. As seen in the graph, the power usage increases during certain hours, which helps us understand when energy demand is high.

This visualization helps monitor power consumption more clearly and makes it easier to spot unusual changes or patterns. It can be useful for improving energy management and efficiency.

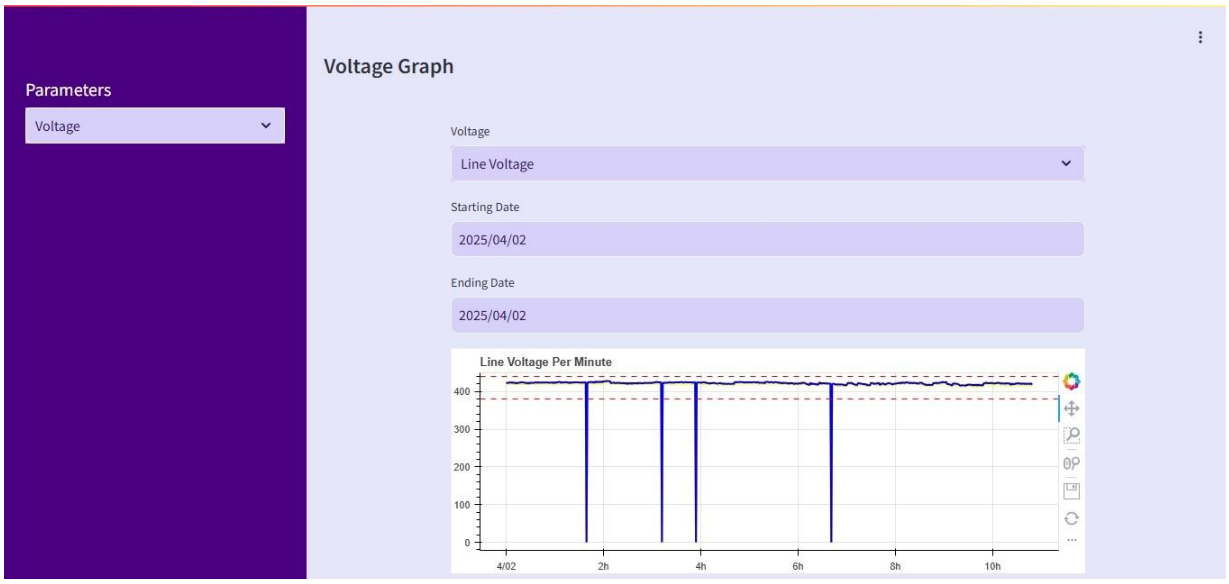


Fig. 6.7 SJT Transformer – 1 Voltage graph

This graph shown in Figure. 6.7 represents the line voltage variation recorded every minute on **April 2, 2025**. Users can select “Line Voltage” from the dropdown and define a date range to monitor voltage trends over time.

The X-axis displays the time of day, while the Y-axis shows the voltage level in volts. The graph includes **dotted lines** that represent **±5% voltage regulation boundaries**, which are commonly used as **safe operation limits** in electrical systems. These indicators help visualize whether the system voltage stays within acceptable thresholds throughout the day.

**VIT Vellore maintains voltage levels in accordance with IEEE Standard 1159-2019**, which provides guidelines for monitoring and characterizing power quality events, including voltage variations. This ensures that the voltage supplied across the campus remains consistent and within regulatory limits, protecting sensitive equipment and promoting efficient power usage.

The graph serves as a key tool for identifying undervoltage, overvoltage, or instability issues, allowing for proactive maintenance and improved system reliability.

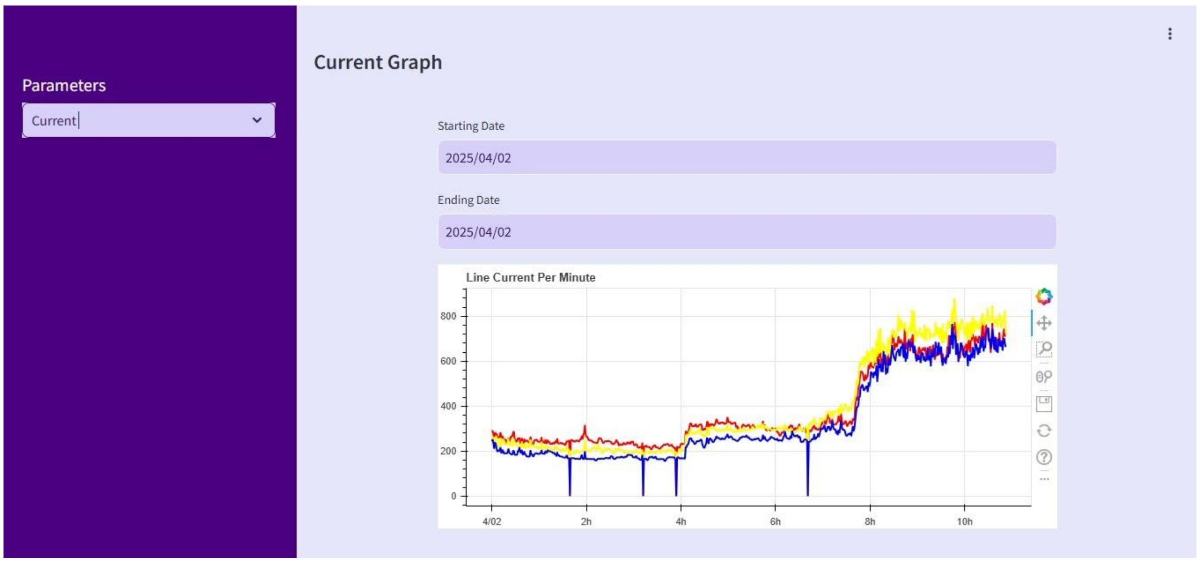


Fig. 6.8 SJT Transformer – 1 Current graph

This graph shown in Fig. 6.8 displays the line current measured per minute on **April 2, 2025**. The user interface allows selecting the "Current" parameter and choosing a date range to view how the current changes over time.

In the graph, the X-axis represents the time of day, and the Y-axis shows the current in amperes. Different colors represent the three phases of current. From the plot, it is visible that the current remained relatively steady during the early hours, followed by a significant increase after the 8th hour, which may indicate a rise in power consumption or equipment activation.

Such visualizations help monitor load patterns and detect abnormalities or imbalances in current flow across different phases, which is crucial for electrical system maintenance and planning

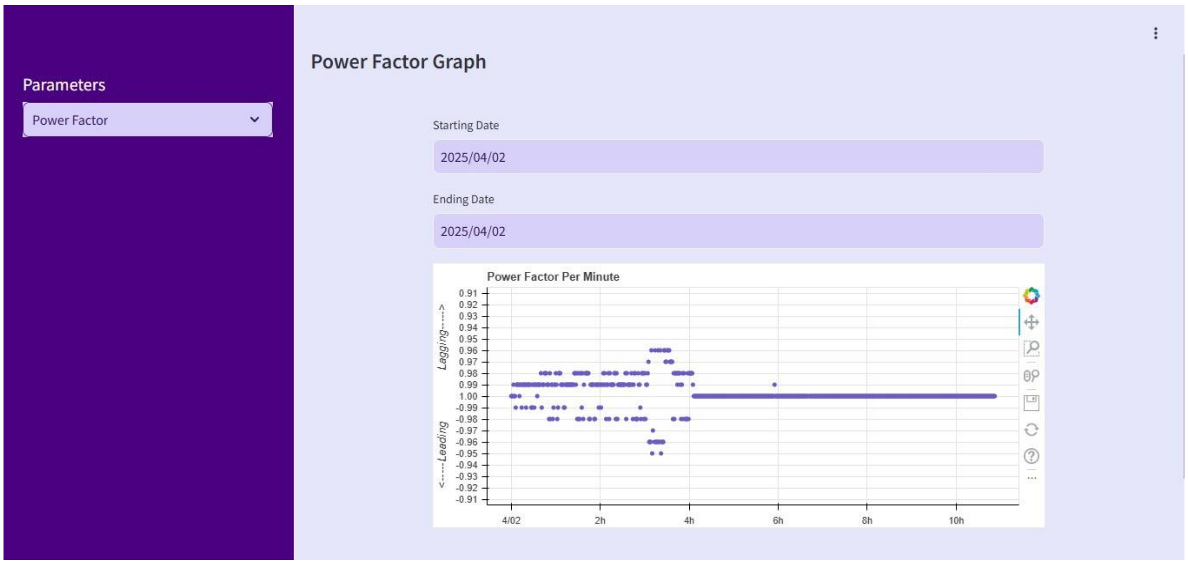


Fig. 6.9 SJT Transformer – 1 Power Factor Graph

The power factor graph provides a visual representation of how efficiently electrical power is being used over time. In this webpage module, the user can select the desired date range, and the graph displays the minute-wise variation of power factor for that day.

Power factor values range between -1 and 1, where values closer to 1 indicate efficient power usage. The graph helps in identifying periods where the load may be predominantly inductive (lagging) or capacitive (leading). In this example, the graph shows scattered points for the initial hours, indicating fluctuations, and later stabilizes around unity, suggesting balanced operation.

This visualization is crucial for monitoring the quality of power usage, as poor power factor can lead to increased losses and penalties in industrial setups. **VIT Vellore consistently operates and maintains the power factor within the optimal range of 0.9 to 1, mostly 1, ensuring efficient power utilization and avoiding penalties from the electricity board.**

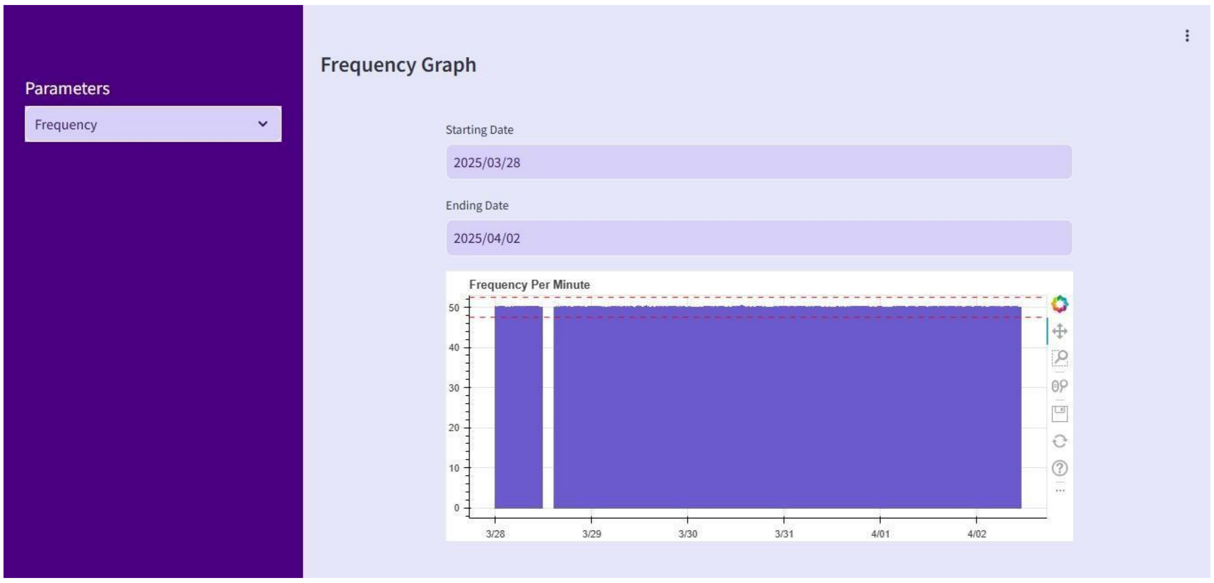


Fig. 6.10 SJT Transformer – 1 Frequency graph

The Frequency Graph shown in fig. 6.10 provides a visual representation of the frequency data recorded per minute across a selected date range. The user interface allows selection of a specific parameter—in this case, **Frequency**—along with customizable start and end dates.

For the chosen range (2025/03/28 to 2025/04/02), the graph displays the measured frequency values aggregated per minute. The data is shown as vertical bars, where each bar represents the frequency values recorded on a specific day.

The graph reveals that the frequency remained relatively stable throughout the observed period, hovering around the nominal value (~50 Hz), which is critical for the reliable operation of electrical systems. A red dashed reference line at the top of the graph represents the standard operational frequency limit as per IEEE guidelines.

Vellore Institute of Technology (VIT) adheres to **IEEE Standard 1159** for monitoring power quality parameters, including frequency stability. The reference line coded into the graph helps users quickly identify if the measured frequency deviates beyond acceptable IEEE tolerance levels.

CHAPTER 7

# RESULT AND DISCUSSION

The real-time energy monitoring website implemented across VIT's substations demonstrated substantial improvements in operational efficiency and system robustness. By employing a modular architecture that integrates UART-enabled smart energy meters, serial-to-WiFi communication, Python-based data processing, and Streamlit visualization, the website provides a scalable and resilient platform for continuous monitoring of critical electrical parameters, including voltage, current, power factor, active power, energy consumption, and frequency. This configuration facilitates proactive maintenance and rapid response to anomalies, thereby enhancing system reliability. For researchers, the project offers a rich dataset and a flexible testbed conducive to developing predictive analytics algorithms and exploring IoT system optimizations. Practitioners benefit from improved operational awareness and reduced maintenance costs through real-time anomaly detection and automated alerts. Future enhancements should consider integrating LoRaWAN or multi-RAT networks to bolster communication resilience and support sub-second sampling for harmonic distortion analysis, aligning with industry standards. Facilities managers are encouraged to adopt modular IoT monitoring solutions and invest in dedicated LPWAN infrastructure to mitigate Wi-Fi congestion and ensure data integrity. Ultimately, this website represents a significant advancement in institutional energy management, laying the groundwork for future innovations in sustainable infrastructure monitoring.

CHAPTER 8

# SUSTAINABLE DEVELOPMENT GOALS (SDGs)

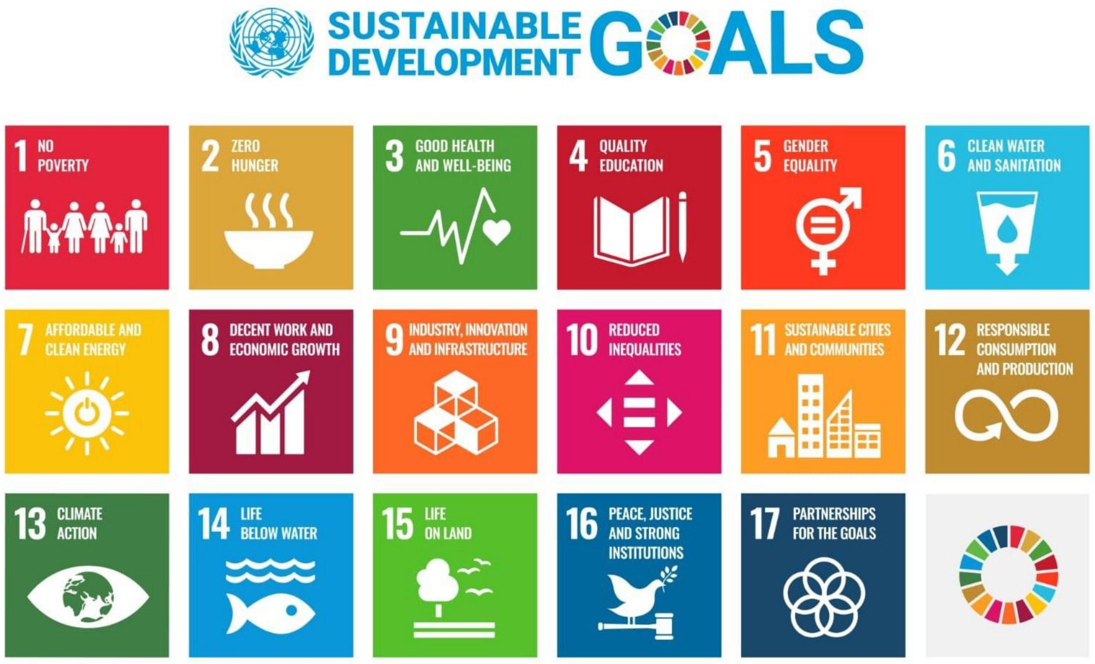

SUSTAINABLE DEVELOPMENT GOALS
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

Fig. 8.1 Sustainable development goals

The Responsible Consumption and Production (SDG 12) aims to ensure sustainable patterns of consumption and production by promoting resource efficiency and reducing waste. It emphasizes the importance of using natural resources wisely, ensuring that economic growth does not come at the cost of environmental degradation. This goal encourages industries and consumers to adopt sustainable practices, such as reducing waste through recycling, reusing products, and minimizing the use of harmful chemicals. Raising awareness among consumers about sustainable lifestyles is a key aspect, along with supporting companies in adopting transparent and eco-friendly methods. Public procurement policies are also guided to favor sustainable options. Furthermore, SDG 12 focuses on reducing global food waste at both the retail and consumer levels, and ensuring food loss is minimized throughout the supply chain. It promotes sustainable tourism that respects the environment and local culture while creating employment. Finally, enhancing the scientific and technological capabilities of developing countries is vital for achieving this goal, allowing for innovation in sustainable production and consumption.

CHAPTER 9

# SUMMARY

The primary goal of this project was to develop a real-time monitoring website for electrical parameters such as voltage, current, power factor, energy, power, and frequency at various substations. This was achieved by utilizing energy meters connected to a serial-to-WiFi converter, which enabled the transmission of data to a centralized platform. The website was designed to provide live updates on the performance of each transformer through individual Streamlit-based websites. These websites displayed the collected data along with corresponding visualizations to facilitate easy and intuitive monitoring. The results of the project were largely consistent with the expected outcomes. The real-time data collection from the energy meters was successfully integrated with the online platform, allowing for immediate access to key electrical parameters. The data was transmitted reliably via the serial-to- WiFi converter, ensuring that the information remained accurate and up-to-date across all substations. The use of Streamlit dashboards proved to be an effective tool for visualizing the data, presenting it in a clear and accessible manner for monitoring purposes.A notable observation from the implementation was the significance of continuous monitoring in enhancing the overall efficiency of energy management at substations. By tracking electrical parameters in real-time, it became possible to detect anomalies or inefficiencies promptly, enabling quick responses to potential issues before they escalate. Moreover, the ability to monitor these parameters remotely contributes to better decision-making and more informed management of energy resources.In conclusion, the project demonstrated the viability and effectiveness of using IoT-based solutions for real-time monitoring of electrical parameters. The website not only met the initial objectives but also highlighted the potential for further improvements, such as integrating predictive maintenance features and expanding the range of parameters monitored. This project serves as a foundation for more advanced energy management systems, promoting both operational efficiency and the proactive management of electrical networks.

# REFERENCES

Al-Mashaqbeh, R. A., Salameh, M. M. A., & Al-Najjar, M. R. (2022). Smart Metering Systems for Energy Monitoring and Management. Energy Reports, 8, 2150– 2163.

Jadhav, S. D. M., Patil, S. S., & Deshmukh, A. K. (2020). Design and Implementation of Smart Metering System Using Wi-Fi for Real-Time Energy Monitoring. *International Journal of Engineering Research & Technology (IJERT), 9*(5).

Lewis, D. R. W. (2020). Streamlit: A New Approach for Building Data Science Web Apps. *Journal of Open Source Software, 5*(50), 1797

Ali, M. S. M., Khan, M. A. N., & Khan, M. A. H. (2021). A Review of Smart Grid Architecture and Smart Metering. *IEEE Access, 9*, 18899–18915.

Hussain, S., & Kiani, A. (2020). Integration of IoT and Smart Metering for Energy Consumption Monitoring in Industrial Automation. Sensors, 20(21), 6055.

Gonzalez, R., & Garcia, E. (2019). Real-Time Data Monitoring and Control System for Energy Management Using IoT. *IEEE Internet of Things Journal, 6*(3), 4520– 4527.

Aghaei, J., & Alizadeh, M. (2020). A Smart Grid Design Using Wireless Communication for Energy Metering and Monitoring. *Sustainable Energy Technologies and Assessments, 37*, 100557.

Song, H., & Yu, Z. (2021). Cloud-Based Real-Time Monitoring of Electrical Parameters Using IoT. *International Journal of Electrical Power & Energy Systems, 123*, 106156

Abdullah, M. M. A., & Mohammad, M. (2021). Smart Metering and IoT: Data Collection and Real-Time Energy Management. *Energy and Buildings, 226*, 110378.

Yadav, M., & Gupta, A. (2021). Development of an IoT-Based Smart Metering System for Energy Monitoring. *Journal of Electrical Engineering & Technology, 16*(2),865–872.

## Curriculum Vitae

|   |   |
|---|---|
| Name: | Sham Ganesh K |
| Father 's name: | Kumarasan R |
| Date of Birth: | 31/08/2003 |
| Nationality: | Indian |
| Sex: | Male |
| Permanent Address: | No. 25/14, Kalathi Annamalai street, Arani, Tiruvannamalai, Tamil Nadu 632301. |
| Phone number: | 9080518057 |
| E-mail ID: | shamganesh.k2021@vitstudent.ac.in |

## Curriculum Vitae

| | |
|---|---|
| Name: | Thavanesh R |
| Father 's name: | Rajakumar G |
| Date of Birth: | 05/03/2004 |
| Nationality: | Indian |
| Sex: | Male |
| Permanent Address: | HIG B-484, TNHB, Seekarajapuram, Ranipet, Tamilnadu-632515. |
| Phone number: | 9342687768 |
| E-mail ID: | Thavanesh.r2021@vitstudent.ac.in |

## Curriculum Vitae

| | |
|---|---|
| Name: | Yuvan Shankar M |
| Father 's name: | Mohan G |
| Date of Birth: | 07/05/2004 |
| Nationality: | Indian |
| Sex: | Male |
| Permanent Address: | No. 4/7 BaskalNaidu street, Saidapet, Vellore, Tamilnadu- 632012. |
| Phone number: | 9342687768 |
| E-mail ID: | Yuvan.shankar2021@vitstudent.ac.in |

## Capstone Project

| | |
|---|---|
| **Project title:** | Electrical Energy Monitoring and Visualization |
| **Team Members:** | Sham Ganesh, Thavanesh, Yuvan Shankar |
| **Faculty Guide:** | Dr. Meikandasivam S |
| **Semester/Year:** | Winter semester 2024-2025 |
| **Project Abstract:** (Not more than 200 words) | This project focuses on the development of a web-based interface for real-time visualization of electrical parameters such as power, voltage, current, frequency, energy, and power factor. The data is collected from smart meters installed across VIT substations using serial-to-WiFi converters and stored in a MySQL database by our seniors. Our contribution involved designing and developing the frontend using the Streamlit library in Python. The website enables users to select a substation, choose parameters, define date ranges, and visualize data through interactive graphs. These graphs help in analyzing energy usage patterns and identifying irregularities in system performance. To enhance operational safety, voltage graphs include dotted lines representing ±5% voltage regulation limits. This visual cue helps ensure the voltage stays within acceptable operating boundaries. The web application provides an intuitive and scalable solution for monitoring and managing electrical data, supporting energy efficiency and reliability across the campus. |
| **Project Title:** | Electrical Energy Monitoring and Visualization |

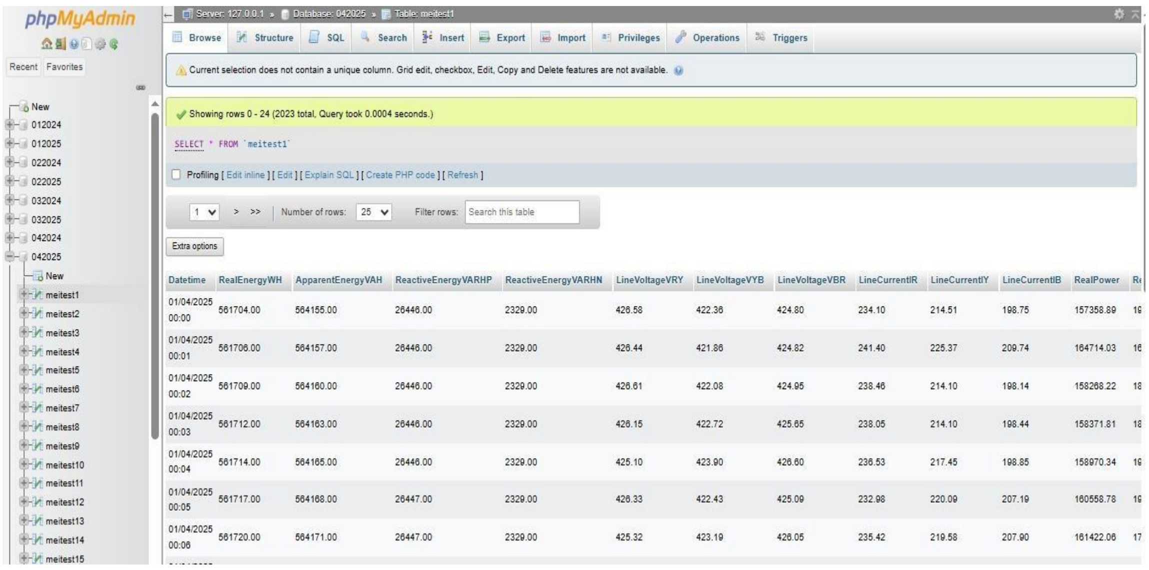
| | |
|---|---|
| List **codes** and **standards** that significantly affect your project (Must) | 1. **IEEE 519 – Harmonic Control in Electric Power Systems:** This standard ensures that the power quality measurements (including power factor and voltage levels) monitored by the system remain within acceptable harmonic distortion limits. It influenced the design of our data monitoring and filtering components.<br>2. **IEC 61010 – Safety Requirements for Electrical Equipment for Measurement, Control, and Laboratory Use:** This standard outlines safety protocols for the devices used in collecting electrical parameters from transformers and smart meters. It guided the selection and handling of hardware to ensure compliance with safety regulations.<br>3. **Modbus Communication Protocol Standard:** Since smart meters typically use Modbus protocol for serial communication, the project architecture and data extraction logic were developed in alignment with this widely adopted standard.<br>4. **IEEE 802.11 (Wi-Fi Standard):** The use of serial-to-WiFi converters requires adherence to this standard to ensure reliable wireless data transmission from substations to the server. |
| List at least two significant realistic design constraints that are applied to your project. (Must) | 1. **Real-time Data Handling and Storage Efficiency:** The system is designed to collect data from multiple smart meters at one-minute intervals and store it continuously in monthly MySQL databases. This imposes constraints on database performance, requiring efficient data management, optimized queries, and periodic archival strategies to ensure the system remains responsive and scalable.<br>2. **Network Reliability and Communication Protocols:** Since data collection relies on serial-to-WiFi converters for transmitting information from smart meters, any fluctuations in WiFi connectivity can disrupt data flow. This creates a dependency on stable and secure network infrastructure, influencing decisions in designing fault-tolerant systems and ensuring data accuracy. |

| | |
|---|---|
| Briefly explain two significant trade-offs considered in your design, including options considered and the solution chosen (Must) | 1. **Real-Time Data Acquisition and Storage:** The system collects energy data from multiple smart meters every minute, requiring continuous, uninterrupted operation. This imposes constraints on processing speed, database efficiency, and system uptime to ensure data is accurately captured and stored in real time without loss.<br>2. **Limited Network Stability and Bandwidth:** The use of serial-to-WiFi converters for communication between meters and the server introduces a dependency on WiFi connectivity. Any signal drops or interference can impact data transmission, so the system must be designed to handle potential communication lags or retries effectively. |
| Describe the computing aspects, if any, of your project. Specifically identifying hardware-software trade-offs, interfaces, and/or interactions | The project enables real-time monitoring of electrical parameters across substations using smart meters. Data is transmitted via serial-to-WiFi converters to a centralized MySQL database on a XAMPP server for easier management and scalability. A Streamlit-based web application visualizes this data, allowing users to view electrical parameters for specific powerhouses. The system balances reliability, cost, and user accessibility with a focus on scalability |

# Appendices

# Appendix A

## MySQL Table Structure



phpMyAdmin

Recent Favorites

- New
- 012024
- 012025
- 022024
- 022025
- 032024
- 032025
- 042024
- 042025
  - New
  - meitest1
  - meitest2
  - meitest3
  - meitest4
  - meitest5
  - meitest6
  - meitest7
  - meitest8
  - meitest9
  - meitest10
  - meitest11
  - meitest12
  - meitest13
  - meitest14
  - meitest15

Server: 127.0.0.1 » Database: 042025 » Table: meitest1

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 24 (2023 total, Query took 0.0004 seconds.)

```
SELECT * FROM `meitest1`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

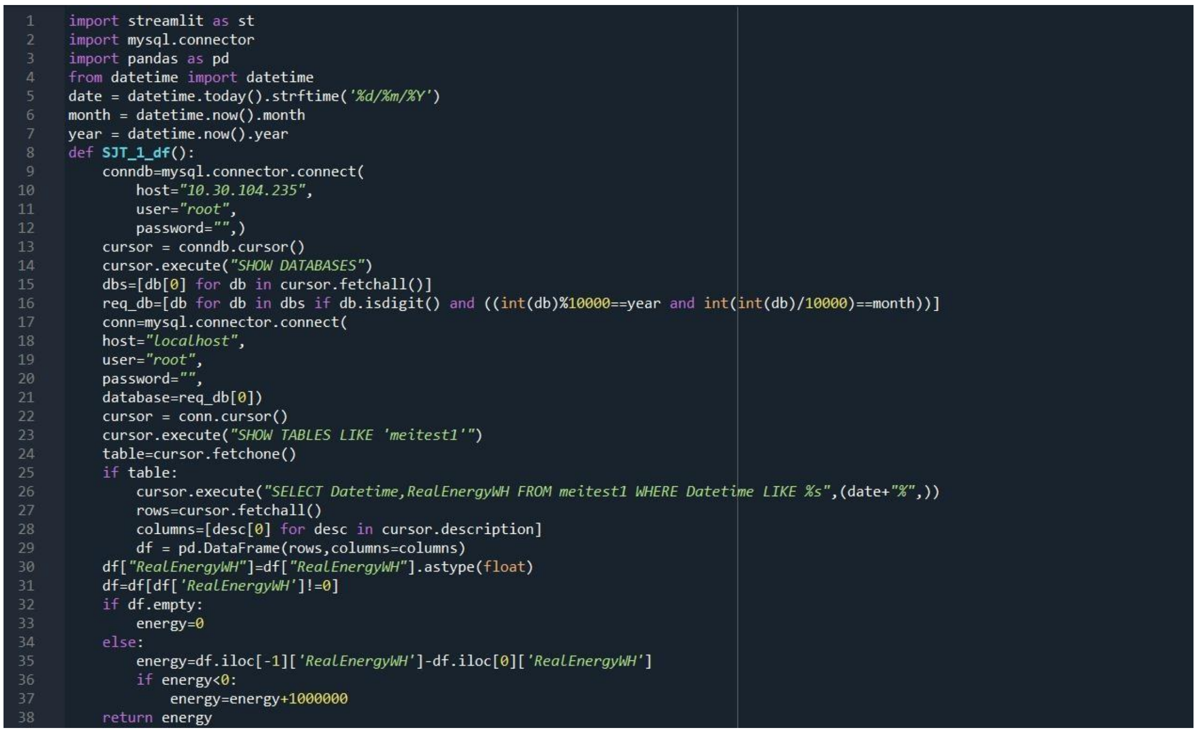
1 > >> | Number of rows: 25 Filter rows: Search this table

Extra options

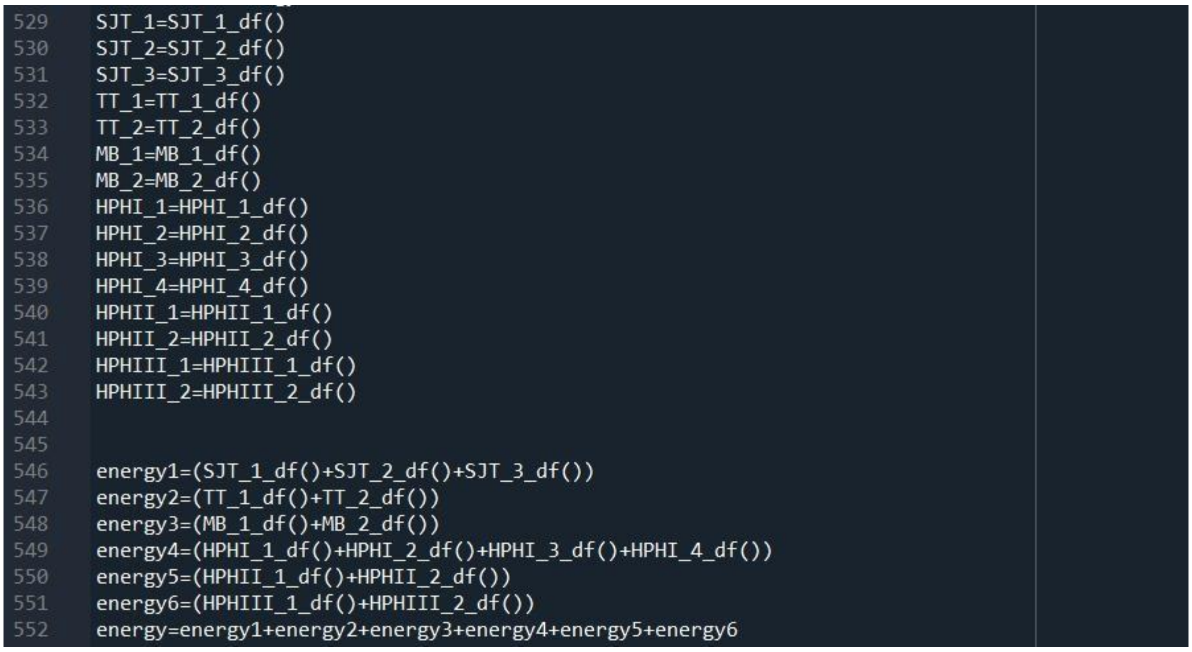
| Datetime | RealEnergyWH | ApparentEnergyVAH | ReactiveEnergyVARHP | ReactiveEnergyVARHN | LineVoltageVRY | LineVoltageVYB | LineVoltageVBR | LineCurrentIR | LineCurrentIY | LineCurrentIB | RealPower | Re |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01/04/2025 00:00 | 561704.00 | 564155.00 | 26446.00 | 2329.00 | 426.58 | 422.36 | 424.80 | 234.10 | 214.51 | 198.75 | 157358.89 | 19 |
| 01/04/2025 00:01 | 561706.00 | 564157.00 | 26446.00 | 2329.00 | 426.44 | 421.86 | 424.82 | 241.40 | 225.37 | 209.74 | 164714.03 | 16 |
| 01/04/2025 00:02 | 561709.00 | 564160.00 | 26446.00 | 2329.00 | 426.61 | 422.08 | 424.95 | 238.46 | 214.10 | 198.14 | 158268.22 | 18 |
| 01/04/2025 00:03 | 561712.00 | 564163.00 | 26446.00 | 2329.00 | 426.15 | 422.72 | 425.65 | 238.05 | 214.10 | 198.44 | 158371.81 | 18 |
| 01/04/2025 00:04 | 561714.00 | 564165.00 | 26446.00 | 2329.00 | 425.10 | 423.90 | 426.60 | 236.53 | 217.45 | 198.85 | 158970.34 | 19 |
| 01/04/2025 00:05 | 561717.00 | 564168.00 | 26447.00 | 2329.00 | 426.33 | 422.43 | 425.09 | 232.98 | 220.09 | 207.19 | 160558.78 | 19 |
| 01/04/2025 00:06 | 561720.00 | 564171.00 | 26447.00 | 2329.00 | 425.32 | 423.19 | 426.05 | 235.42 | 219.58 | 207.90 | 161422.06 | 17 |

## Appendix B

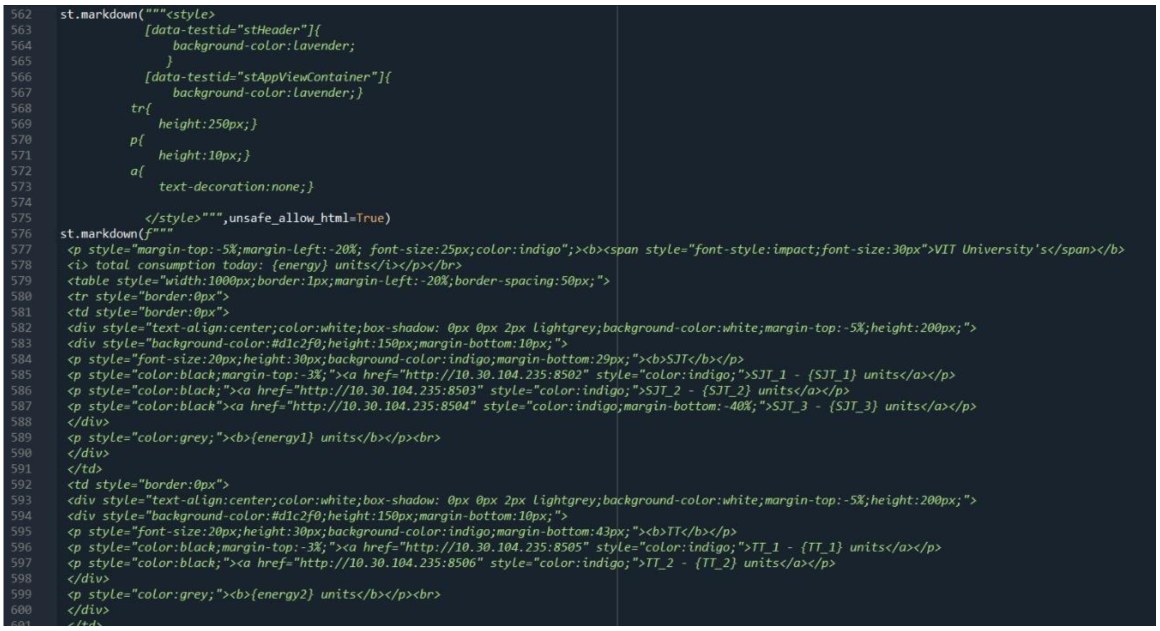
## Streamlit Code Snippets



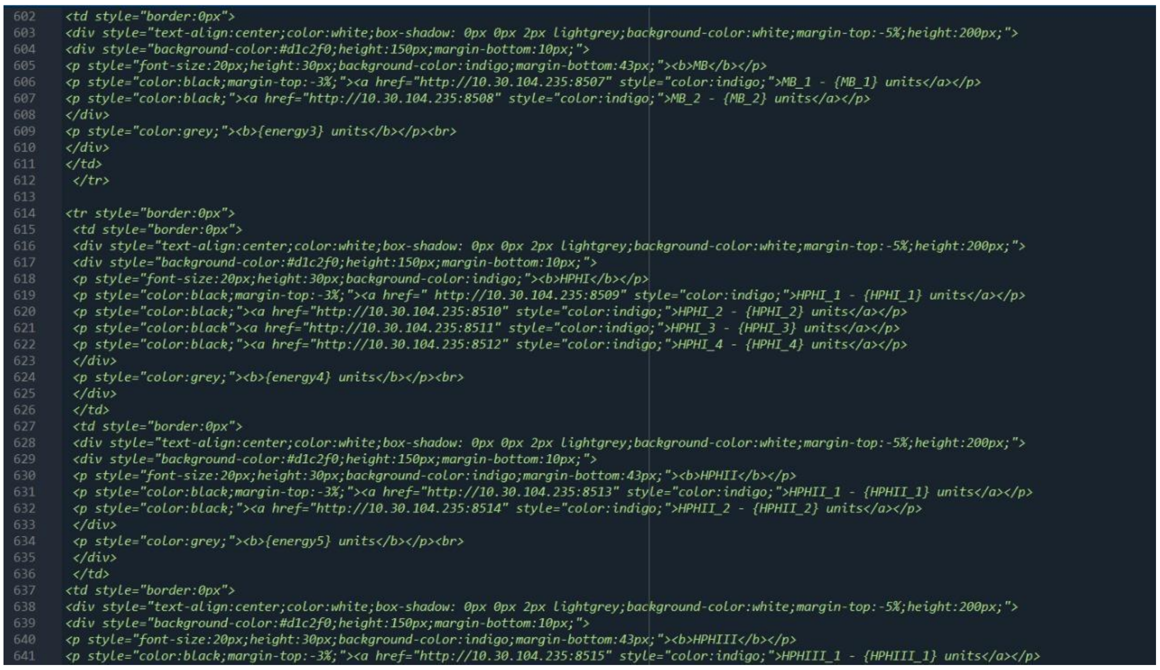
```python
import streamlit as st
import mysql.connector
import pandas as pd
from datetime import datetime
date = datetime.today().strftime('%d/%m/%Y')
month = datetime.now().month
year = datetime.now().year
def SJT_1_df():
    conndb=mysql.connector.connect(
        host="10.30.104.235",
        user="root",
        password="",)
    cursor = conndb.cursor()
    cursor.execute("SHOW DATABASES")
    dbs=[db[0] for db in cursor.fetchall()]
    req_db=[db for db in dbs if db.isdigit() and ((int(db)%10000==year and int(int(db)/10000)==month))]
    conn=mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database=req_db[0])
    cursor = conn.cursor()
    cursor.execute("SHOW TABLES LIKE 'meitest1'")
    table=cursor.fetchone()
    if table:
        cursor.execute("SELECT Datetime,RealEnergyWH FROM meitest1 WHERE Datetime LIKE %s",(date+"%",))
        rows=cursor.fetchall()
        columns=[desc[0] for desc in cursor.description]
        df = pd.DataFrame(rows,columns=columns)
    df["RealEnergyWH"]=df["RealEnergyWH"].astype(float)
    df=df[df['RealEnergyWH']!=0]
    if df.empty:
        energy=0
    else:
        energy=df.iloc[-1]['RealEnergyWH']-df.iloc[0]['RealEnergyWH']
        if energy<0:
            energy=energy+1000000
    return energy
```



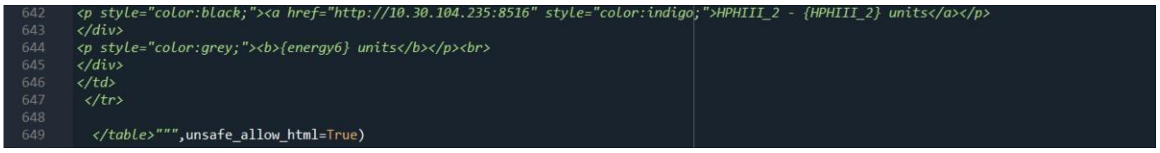
```python
SJT_1=SJT_1_df()
SJT_2=SJT_2_df()
SJT_3=SJT_3_df()
TT_1=TT_1_df()
TT_2=TT_2_df()
MB_1=MB_1_df()
MB_2=MB_2_df()
HPHI_1=HPHI_1_df()
HPHI_2=HPHI_2_df()
HPHI_3=HPHI_3_df()
HPHI_4=HPHI_4_df()
HPHII_1=HPHII_1_df()
HPHII_2=HPHII_2_df()
HPHIII_1=HPHIII_1_df()
HPHIII_2=HPHIII_2_df()


energy1=(SJT_1_df()+SJT_2_df()+SJT_3_df())
energy2=(TT_1_df()+TT_2_df())
energy3=(MB_1_df()+MB_2_df())
energy4=(HPHI_1_df()+HPHI_2_df()+HPHI_3_df()+HPHI_4_df())
energy5=(HPHII_1_df()+HPHII_2_df())
energy6=(HPHIII_1_df()+HPHIII_2_df())
energy=energy1+energy2+energy3+energy4+energy5+energy6
```



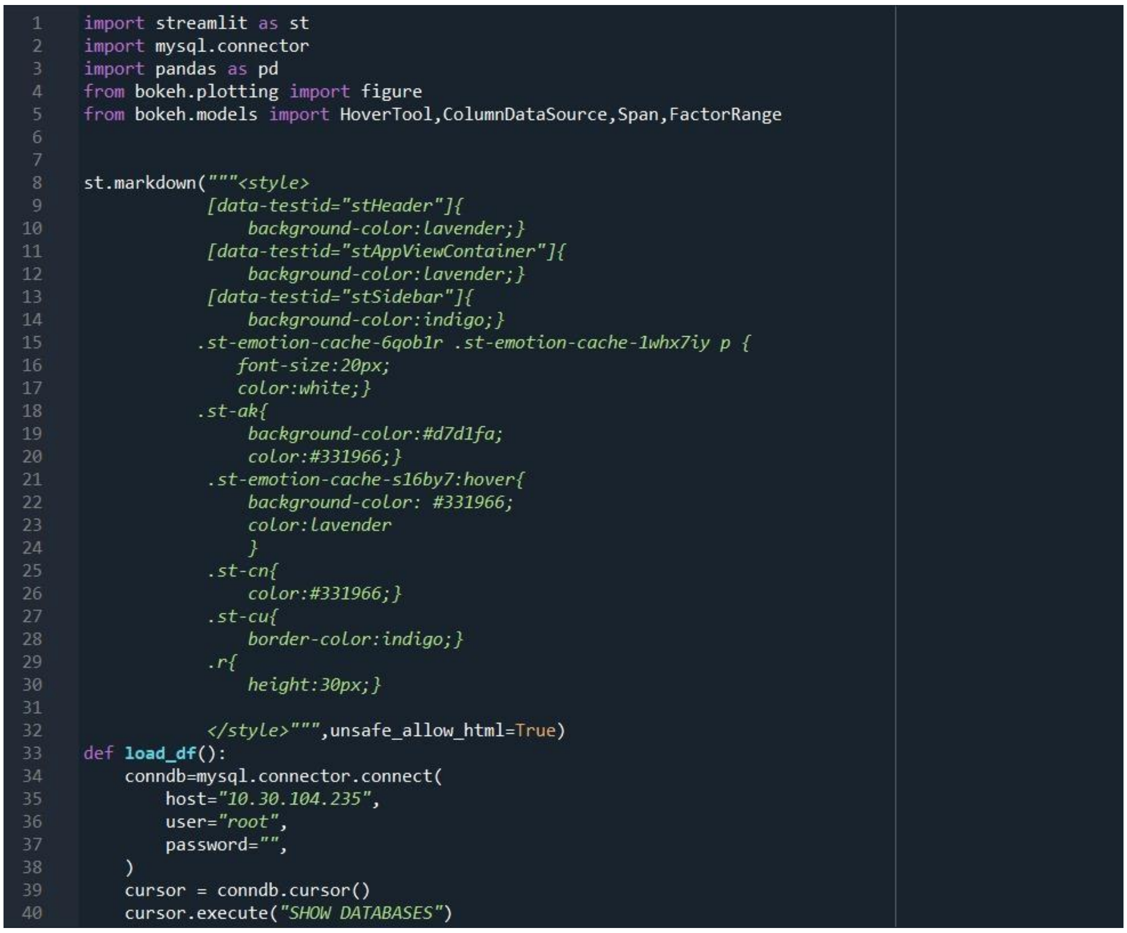
```
st.markdown("""<style>
            [data-testid="stHeader"]{
                background-color:lavender;
                  }
            [data-testid="stAppViewContainer"]{
                background-color:lavender;}
          tr{
              height:250px;}
          p{
              height:10px;}
          a{
              text-decoration:none;}

            </style>""",unsafe_allow_html=True)
st.markdown(f"""
 <p style="margin-top:-5%;margin-left:-20%; font-size:25px;color:indigo";><b><span style="font-style:impact;font-size:30px">VIT University's</span></b>
 <i> total consumption today: {energy} units</i></p></br>
 <table style="width:1000px;border:1px;margin-left:-20%;border-spacing:50px;">
 <tr style="border:0px">
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;height:200px;">
 <div style="background-color:#d1c2f0;height:150px;margin-bottom:10px;">
 <p style="font-size:20px;height:30px;background-color:indigo;margin-bottom:29px;"><b>SJT</b></p>
 <p style="color:black;margin-top:-3%;"><a href="http://10.30.104.235:8502" style="color:indigo;">SJT_1 - {SJT_1} units</a></p>
 <p style="color:black;"><a href="http://10.30.104.235:8503" style="color:indigo;">SJT_2 - {SJT_2} units</a></p>
 <p style="color:black"><a href="http://10.30.104.235:8504" style="color:indigo;margin-bottom:-40%;">SJT_3 - {SJT_3} units</a></p>
 </div>
 <p style="color:grey;"><b>{energy1} units</b></p><br>
 </div>
 </td>
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;height:200px;">
 <div style="background-color:#d1c2f0;height:150px;margin-bottom:10px;">
 <p style="font-size:20px;height:30px;background-color:indigo;margin-bottom:43px;"><b>TT</b></p>
 <p style="color:black;margin-top:-3%;"><a href="http://10.30.104.235:8505" style="color:indigo;">TT_1 - {TT_1} units</a></p>
 <p style="color:black;"><a href="http://10.30.104.235:8506" style="color:indigo;">TT_2 - {TT_2} units</a></p>
 </div>
 <p style="color:grey;"><b>{energy2} units</b></p><br>
 </div>
```



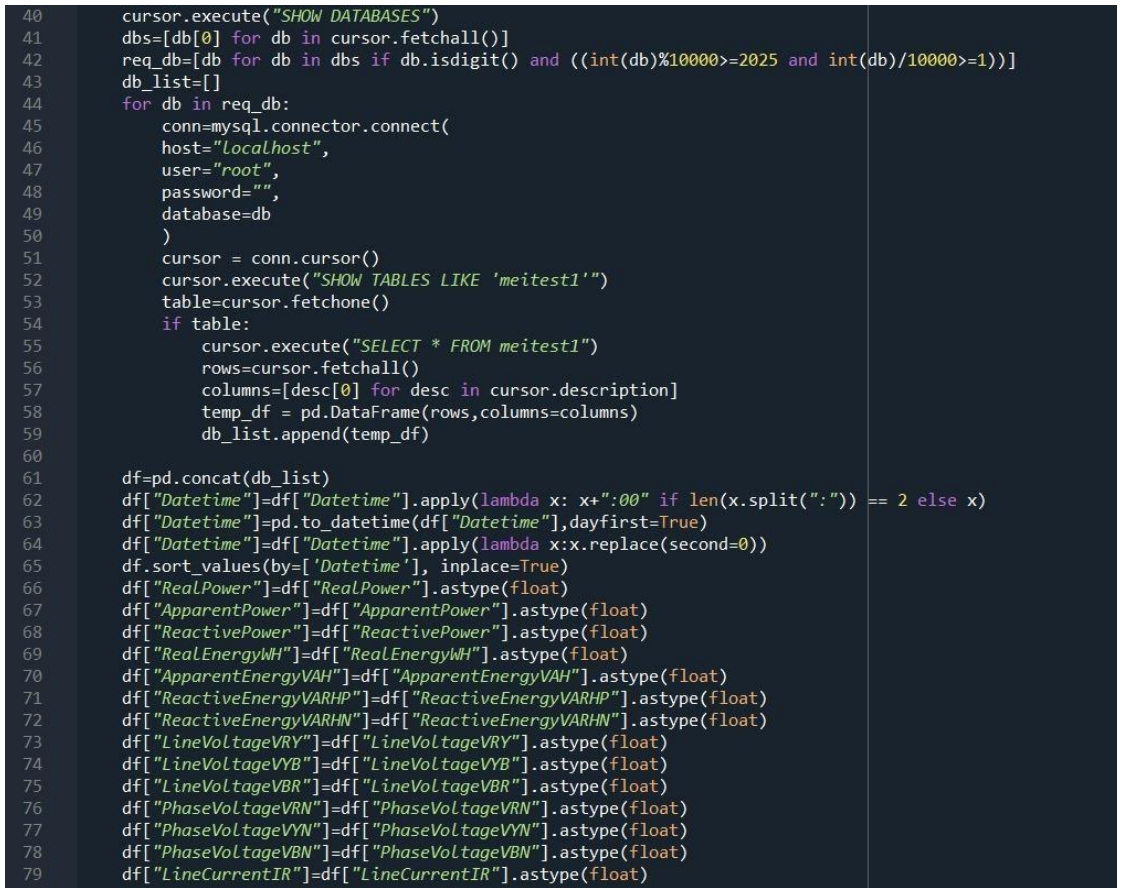
```
<td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;height:200px;">
<div style="background-color:#d1c2f0;height:150px;margin-bottom:10px;">
<p style="font-size:20px;height:30px;background-color:indigo;margin-bottom:43px;"><b>MB</b></p>
<p style="color:black;margin-top:-3%;"><a href="http://10.30.104.235:8507" style="color:indigo;">MB_1 - {MB_1} units</a></p>
<p style="color:black;"><a href="http://10.30.104.235:8508" style="color:indigo;">MB_2 - {MB_2} units</a></p>
</div>
<p style="color:grey;"><b>{energy3} units</b></p><br>
</div>
</td>
 </tr>

<tr style="border:0px">
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;height:200px;">
 <div style="background-color:#d1c2f0;height:150px;margin-bottom:10px;">
 <p style="font-size:20px;height:30px;background-color:indigo;"><b>HPHI</b></p>
 <p style="color:black;margin-top:-3%;"><a href=" http://10.30.104.235:8509" style="color:indigo;">HPHI_1 - {HPHI_1} units</a></p>
 <p style="color:black;"><a href="http://10.30.104.235:8510" style="color:indigo;">HPHI_2 - {HPHI_2} units</a></p>
 <p style="color:black"><a href="http://10.30.104.235:8511" style="color:indigo;">HPHI_3 - {HPHI_3} units</a></p>
 <p style="color:black;"><a href="http://10.30.104.235:8512" style="color:indigo;">HPHI_4 - {HPHI_4} units</a></p>
 </div>
 <p style="color:grey;"><b>{energy4} units</b></p><br>
 </div>
 </td>
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;height:200px;">
 <div style="background-color:#d1c2f0;height:150px;margin-bottom:10px;">
 <p style="font-size:20px;height:30px;background-color:indigo;margin-bottom:43px;"><b>HPHII</b></p>
 <p style="color:black;margin-top:-3%;"><a href="http://10.30.104.235:8513" style="color:indigo;">HPHII_1 - {HPHII_1} units</a></p>
 <p style="color:black;"><a href="http://10.30.104.235:8514" style="color:indigo;">HPHII_2 - {HPHII_2} units</a></p>
 </div>
 <p style="color:grey;"><b>{energy5} units</b></p><br>
 </div>
 </td>
<td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;height:200px;">
<div style="background-color:#d1c2f0;height:150px;margin-bottom:10px;">
<p style="font-size:20px;height:30px;background-color:indigo;margin-bottom:43px;"><b>HPHIII</b></p>
<p style="color:black;margin-top:-3%;"><a href="http://10.30.104.235:8515" style="color:indigo;">HPHIII_1 - {HPHIII_1} units</a></p>
```



```
<p style="color:black;"><a href="http://10.30.104.235:8516" style="color:indigo;">HPHIII_2 - {HPHIII_2} units</a></p>
</div>
<p style="color:grey;"><b>{energy6} units</b></p><br>
</div>
</td>
 </tr>

  </table>""",unsafe_allow_html=True)
```

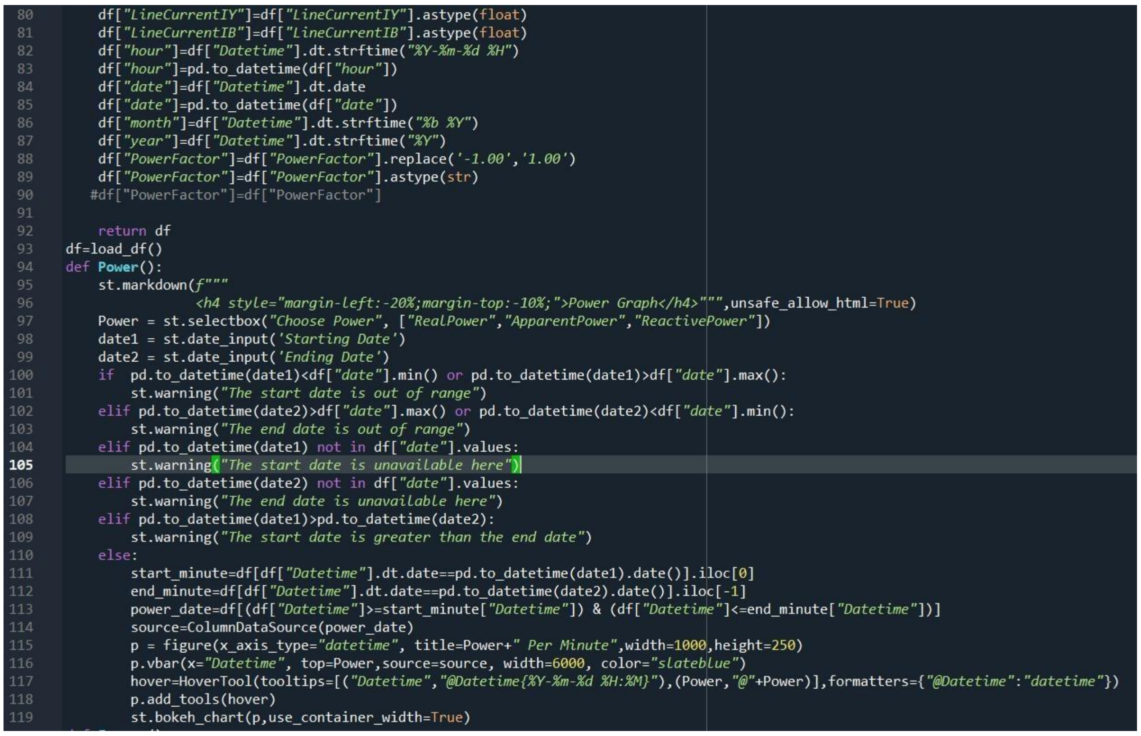


```python
import streamlit as st
import mysql.connector
import pandas as pd
from bokeh.plotting import figure
from bokeh.models import HoverTool,ColumnDataSource,Span,FactorRange


st.markdown("""<style>
            [data-testid="stHeader"]{
                background-color:lavender;}
            [data-testid="stAppViewContainer"]{
                background-color:lavender;}
            [data-testid="stSidebar"]{
                background-color:indigo;}
           .st-emotion-cache-6qob1r .st-emotion-cache-1whx7iy p {
               font-size:20px;
               color:white;}
           .st-ak{
                background-color:#d7d1fa;
                color:#331966;}
            .st-emotion-cache-s16by7:hover{
                background-color: #331966;
                color:lavender
                }
            .st-cn{
                color:#331966;}
            .st-cu{
                border-color:indigo;}
            .r{
                height:30px;}

            </style>""",unsafe_allow_html=True)
def load_df():
    conndb=mysql.connector.connect(
        host="10.30.104.235",
        user="root",
        password="",
    )
    cursor = conndb.cursor()
    cursor.execute("SHOW DATABASES")
```

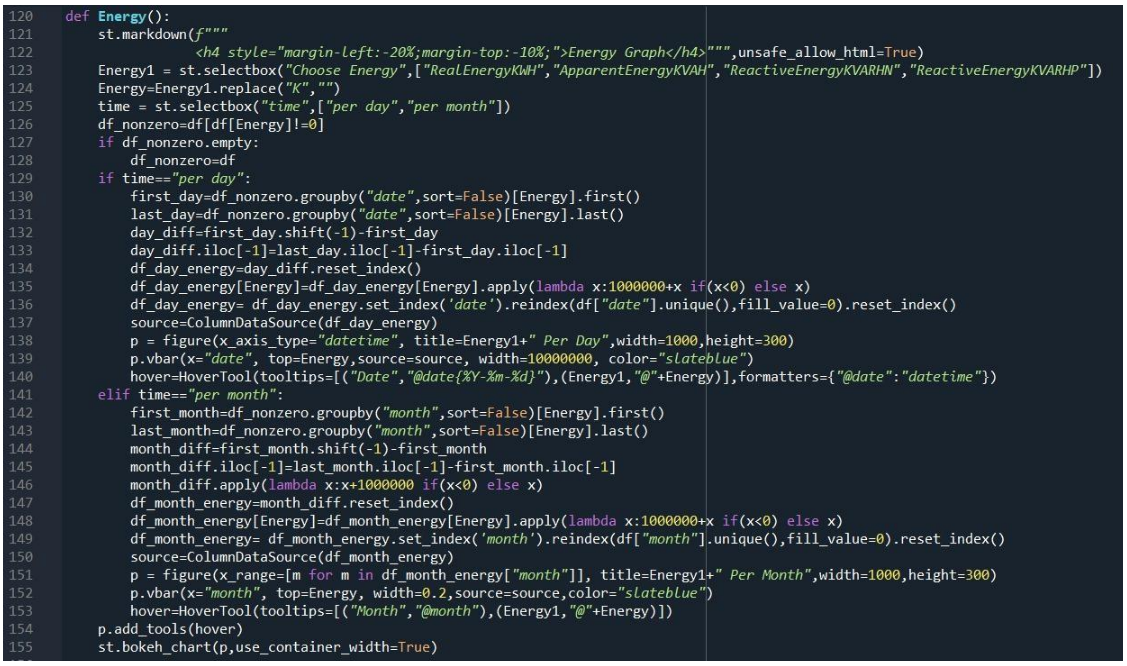


```python
    cursor.execute("SHOW DATABASES")
```

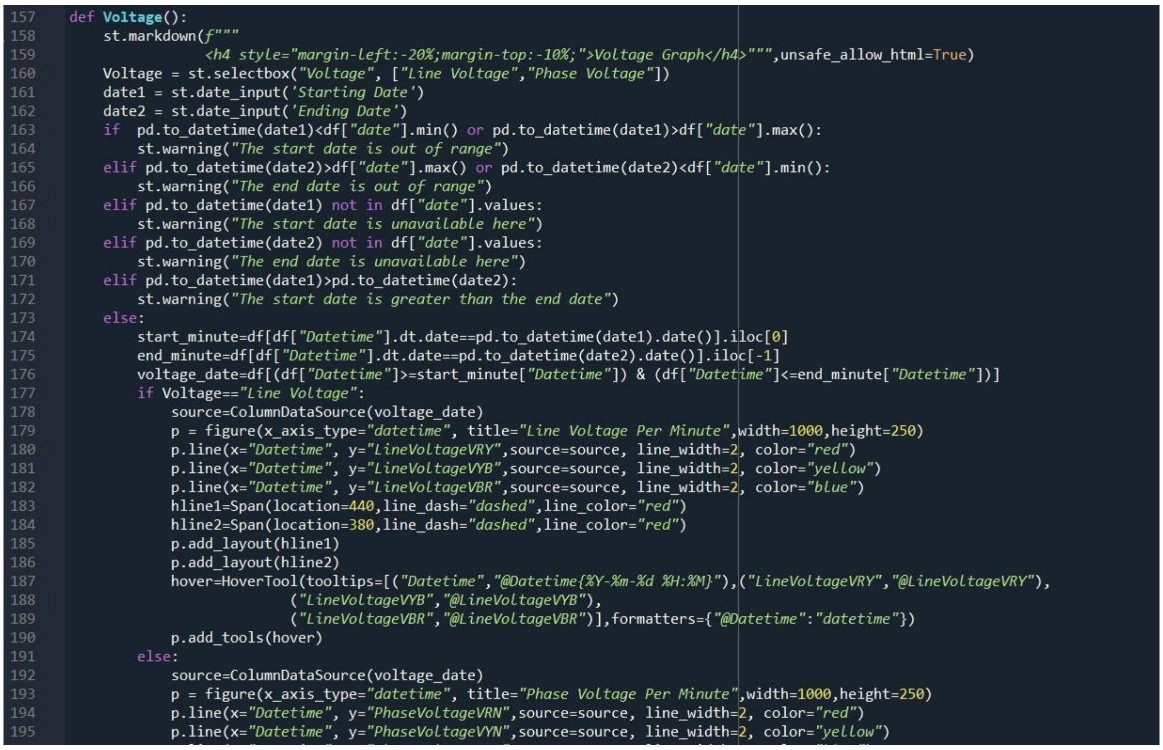
```python
    dbs=[db[0] for db in cursor.fetchall()]
    req_db=[db for db in dbs if db.isdigit() and ((int(db)%10000>=2025 and int(db)/10000>=1))]
    db_list=[]
    for db in req_db:
        conn=mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database=db
        )
        cursor = conn.cursor()
        cursor.execute("SHOW TABLES LIKE 'meitest1'")
        table=cursor.fetchone()
        if table:
            cursor.execute("SELECT * FROM meitest1")
            rows=cursor.fetchall()
            columns=[desc[0] for desc in cursor.description]
            temp_df = pd.DataFrame(rows,columns=columns)
            db_list.append(temp_df)

    df=pd.concat(db_list)
    df["Datetime"]=df["Datetime"].apply(lambda x: x+":00" if len(x.split(":")) == 2 else x)
    df["Datetime"]=pd.to_datetime(df["Datetime"],dayfirst=True)
    df["Datetime"]=df["Datetime"].apply(lambda x:x.replace(second=0))
    df.sort_values(by=['Datetime'], inplace=True)
    df["RealPower"]=df["RealPower"].astype(float)
    df["ApparentPower"]=df["ApparentPower"].astype(float)
    df["ReactivePower"]=df["ReactivePower"].astype(float)
    df["RealEnergyWH"]=df["RealEnergyWH"].astype(float)
    df["ApparentEnergyVAH"]=df["ApparentEnergyVAH"].astype(float)
    df["ReactiveEnergyVARHP"]=df["ReactiveEnergyVARHP"].astype(float)
    df["ReactiveEnergyVARHN"]=df["ReactiveEnergyVARHN"].astype(float)
    df["LineVoltageVRY"]=df["LineVoltageVRY"].astype(float)
    df["LineVoltageVYB"]=df["LineVoltageVYB"].astype(float)
    df["LineVoltageVBR"]=df["LineVoltageVBR"].astype(float)
    df["PhaseVoltageVRN"]=df["PhaseVoltageVRN"].astype(float)
    df["PhaseVoltageVYN"]=df["PhaseVoltageVYN"].astype(float)
    df["PhaseVoltageVBN"]=df["PhaseVoltageVBN"].astype(float)
    df["LineCurrentIR"]=df["LineCurrentIR"].astype(float)
```



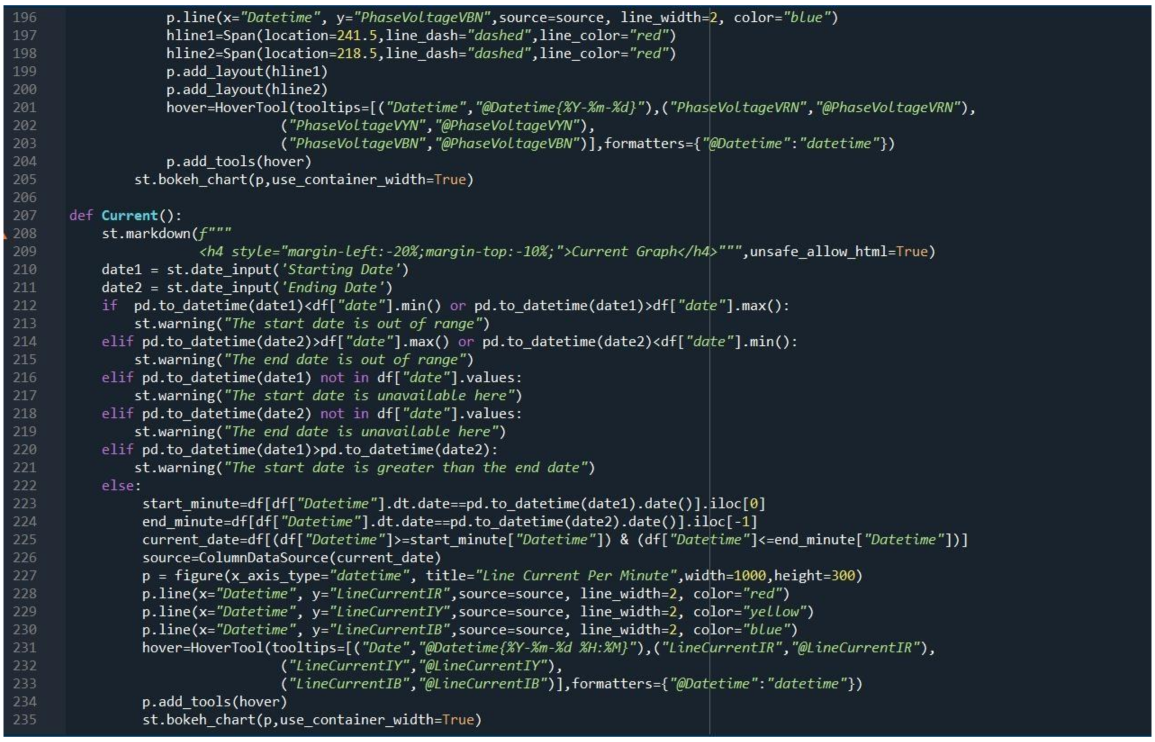
```python
    df["LineCurrentIY"]=df["LineCurrentIY"].astype(float)
    df["LineCurrentIB"]=df["LineCurrentIB"].astype(float)
    df["hour"]=df["Datetime"].dt.strftime("%Y-%m-%d %H")
    df["hour"]=pd.to_datetime(df["hour"])
    df["date"]=df["Datetime"].dt.date
    df["date"]=pd.to_datetime(df["date"])
    df["month"]=df["Datetime"].dt.strftime("%b %Y")
    df["year"]=df["Datetime"].dt.strftime("%Y")
    df["PowerFactor"]=df["PowerFactor"].replace('-1.00','1.00')
    df["PowerFactor"]=df["PowerFactor"].astype(str)
   #df["PowerFactor"]=df["PowerFactor"]

    return df
df=load_df()
def Power():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Power Graph</h4>""",unsafe_allow_html=True)
    Power = st.selectbox("Choose Power", ["RealPower","ApparentPower","ReactivePower"])
    date1 = st.date_input('Starting Date')
    date2 = st.date_input('Ending Date')
    if  pd.to_datetime(date1)<df["date"].min() or pd.to_datetime(date1)>df["date"].max():
        st.warning("The start date is out of range")
    elif pd.to_datetime(date2)>df["date"].max() or pd.to_datetime(date2)<df["date"].min():
        st.warning("The end date is out of range")
    elif pd.to_datetime(date1) not in df["date"].values:
        st.warning("The start date is unavailable here")
    elif pd.to_datetime(date2) not in df["date"].values:
        st.warning("The end date is unavailable here")
    elif pd.to_datetime(date1)>pd.to_datetime(date2):
        st.warning("The start date is greater than the end date")
    else:
        start_minute=df[df["Datetime"].dt.date==pd.to_datetime(date1).date()].iloc[0]
        end_minute=df[df["Datetime"].dt.date==pd.to_datetime(date2).date()].iloc[-1]
        power_date=df[(df["Datetime"]>=start_minute["Datetime"]) & (df["Datetime"]<=end_minute["Datetime"])]
        source=ColumnDataSource(power_date)
        p = figure(x_axis_type="datetime", title=Power+" Per Minute",width=1000,height=250)
        p.vbar(x="Datetime", top=Power,source=source, width=6000, color="slateblue")
        hover=HoverTool(tooltips=[("Datetime","@Datetime{%Y-%m-%d %H:%M}"),(Power,"@"+Power)],formatters={"@Datetime":"datetime"})
        p.add_tools(hover)
        st.bokeh_chart(p,use_container_width=True)
```



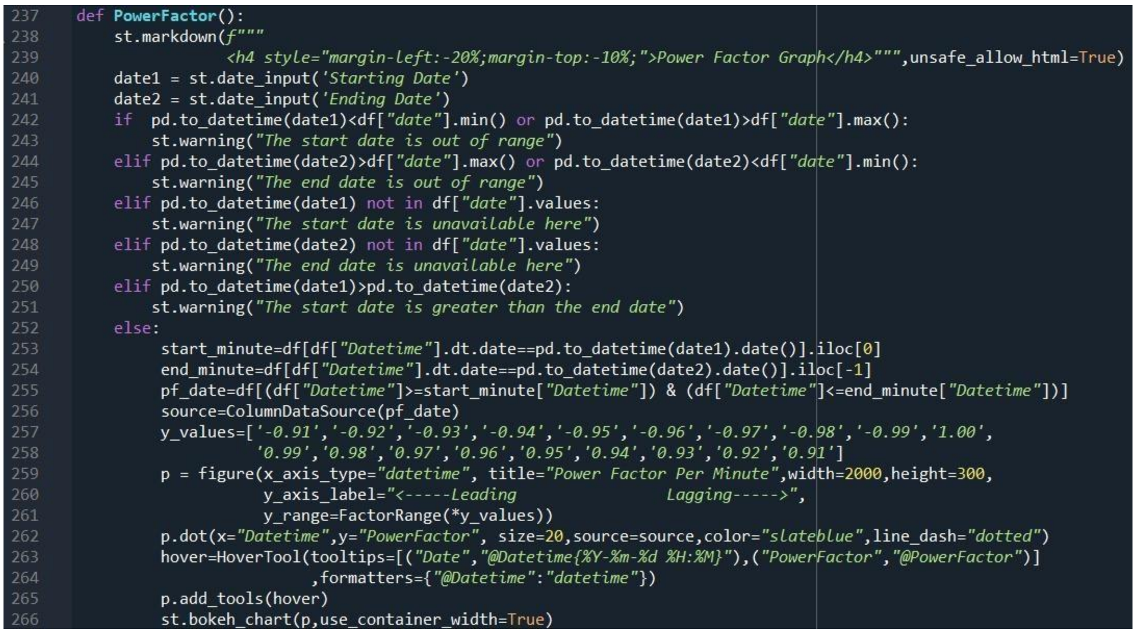
```python
def Energy():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Energy Graph</h4>""",unsafe_allow_html=True)
    Energy1 = st.selectbox("Choose Energy",["RealEnergyKWH","ApparentEnergyKVAH","ReactiveEnergyKVARHN","ReactiveEnergyKVARHP"])
    Energy=Energy1.replace("K","")
    time = st.selectbox("time",["per day","per month"])
    df_nonzero=df[df[Energy]!=0]
    if df_nonzero.empty:
        df_nonzero=df
    if time=="per day":
        first_day=df_nonzero.groupby("date",sort=False)[Energy].first()
        last_day=df_nonzero.groupby("date",sort=False)[Energy].last()
        day_diff=first_day.shift(-1)-first_day
        day_diff.iloc[-1]=last_day.iloc[-1]-first_day.iloc[-1]
        df_day_energy=day_diff.reset_index()
        df_day_energy[Energy]=df_day_energy[Energy].apply(lambda x:1000000+x if(x<0) else x)
        df_day_energy= df_day_energy.set_index('date').reindex(df["date"].unique(),fill_value=0).reset_index()
        source=ColumnDataSource(df_day_energy)
        p = figure(x_axis_type="datetime", title=Energy1+" Per Day",width=1000,height=300)
        p.vbar(x="date", top=Energy,source=source, width=10000000, color="slateblue")
        hover=HoverTool(tooltips=[("Date","@date{%Y-%m-%d}"),(Energy1,"@"+Energy)],formatters={"@date":"datetime"})
    elif time=="per month":
        first_month=df_nonzero.groupby("month",sort=False)[Energy].first()
        last_month=df_nonzero.groupby("month",sort=False)[Energy].last()
        month_diff=first_month.shift(-1)-first_month
        month_diff.iloc[-1]=last_month.iloc[-1]-first_month.iloc[-1]
        month_diff.apply(lambda x:x+1000000 if(x<0) else x)
        df_month_energy=month_diff.reset_index()
        df_month_energy[Energy]=df_month_energy[Energy].apply(lambda x:1000000+x if(x<0) else x)
        df_month_energy= df_month_energy.set_index('month').reindex(df["month"].unique(),fill_value=0).reset_index()
        source=ColumnDataSource(df_month_energy)
        p = figure(x_range=[m for m in df_month_energy["month"]], title=Energy1+" Per Month",width=1000,height=300)
        p.vbar(x="month", top=Energy, width=0.2,source=source,color="slateblue")
        hover=HoverTool(tooltips=[("Month","@month"),(Energy1,"@"+Energy)])
    p.add_tools(hover)
    st.bokeh_chart(p,use_container_width=True)
```



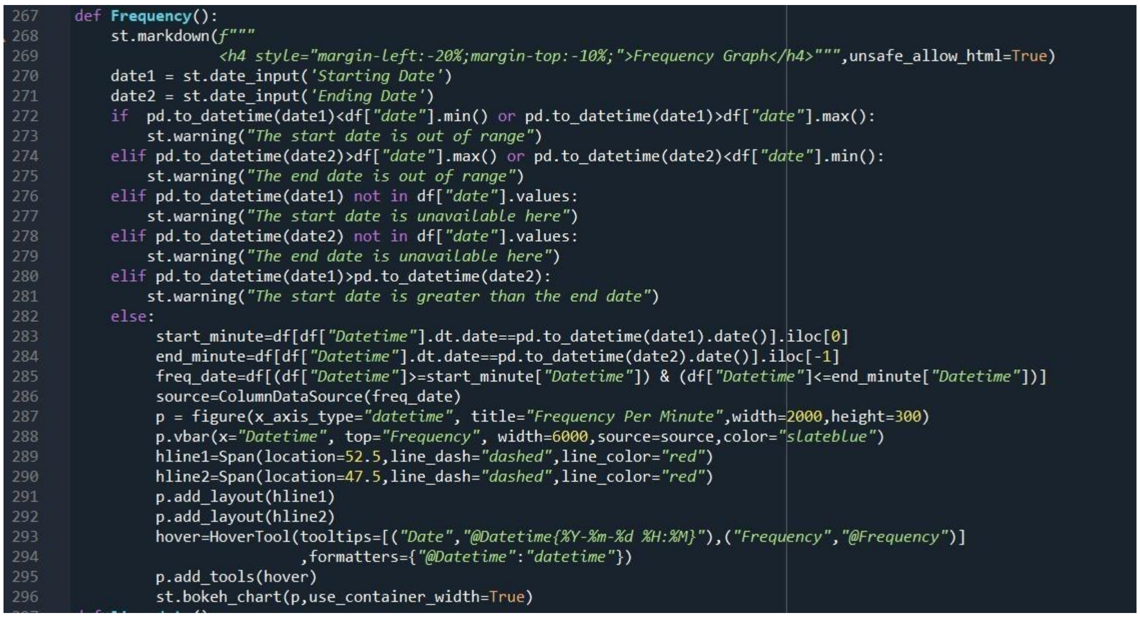
```python
def Voltage():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Voltage Graph</h4>""",unsafe_allow_html=True)
    Voltage = st.selectbox("Voltage", ["Line Voltage","Phase Voltage"])
    date1 = st.date_input('Starting Date')
    date2 = st.date_input('Ending Date')
    if  pd.to_datetime(date1)<df["date"].min() or pd.to_datetime(date1)>df["date"].max():
        st.warning("The start date is out of range")
    elif pd.to_datetime(date2)>df["date"].max() or pd.to_datetime(date2)<df["date"].min():
        st.warning("The end date is out of range")
    elif pd.to_datetime(date1) not in df["date"].values:
        st.warning("The start date is unavailable here")
    elif pd.to_datetime(date2) not in df["date"].values:
        st.warning("The end date is unavailable here")
    elif pd.to_datetime(date1)>pd.to_datetime(date2):
        st.warning("The start date is greater than the end date")
    else:
        start_minute=df[df["Datetime"].dt.date==pd.to_datetime(date1).date()].iloc[0]
        end_minute=df[df["Datetime"].dt.date==pd.to_datetime(date2).date()].iloc[-1]
        voltage_date=df[(df["Datetime"]>=start_minute["Datetime"]) & (df["Datetime"]<=end_minute["Datetime"])]
        if Voltage=="Line Voltage":
            source=ColumnDataSource(voltage_date)
            p = figure(x_axis_type="datetime", title="Line Voltage Per Minute",width=1000,height=250)
            p.line(x="Datetime", y="LineVoltageVRY",source=source, line_width=2, color="red")
            p.line(x="Datetime", y="LineVoltageVYB",source=source, line_width=2, color="yellow")
            p.line(x="Datetime", y="LineVoltageVBR",source=source, line_width=2, color="blue")
            hline1=Span(location=440,line_dash="dashed",line_color="red")
            hline2=Span(location=380,line_dash="dashed",line_color="red")
            p.add_layout(hline1)
            p.add_layout(hline2)
            hover=HoverTool(tooltips=[("Datetime","@Datetime{%Y-%m-%d %H:%M}"),("LineVoltageVRY","@LineVoltageVRY"),
                          ("LineVoltageVYB","@LineVoltageVYB"),
                          ("LineVoltageVBR","@LineVoltageVBR")],formatters={"@Datetime":"datetime"})
            p.add_tools(hover)
        else:
            source=ColumnDataSource(voltage_date)
            p = figure(x_axis_type="datetime", title="Phase Voltage Per Minute",width=1000,height=250)
            p.line(x="Datetime", y="PhaseVoltageVRN",source=source, line_width=2, color="red")
            p.line(x="Datetime", y="PhaseVoltageVYN",source=source, line_width=2, color="yellow")
```



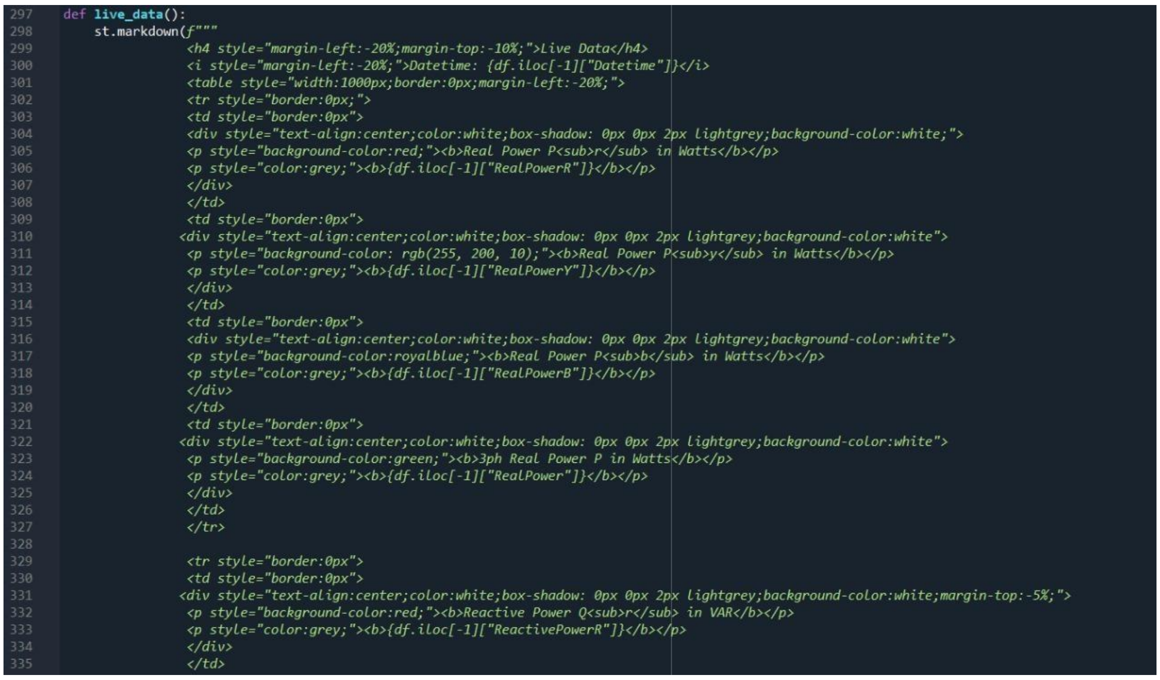
```python
            p.line(x="Datetime", y="PhaseVoltageVBN",source=source, line_width=2, color="blue")
            hline1=Span(location=241.5,line_dash="dashed",line_color="red")
            hline2=Span(location=218.5,line_dash="dashed",line_color="red")
            p.add_layout(hline1)
            p.add_layout(hline2)
            hover=HoverTool(tooltips=[("Datetime","@Datetime{%Y-%m-%d}"),("PhaseVoltageVRN","@PhaseVoltageVRN"),
                          ("PhaseVoltageVYN","@PhaseVoltageVYN"),
                          ("PhaseVoltageVBN","@PhaseVoltageVBN")],formatters={"@Datetime":"datetime"})
            p.add_tools(hover)
        st.bokeh_chart(p,use_container_width=True)

def Current():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Current Graph</h4>""",unsafe_allow_html=True)
    date1 = st.date_input('Starting Date')
    date2 = st.date_input('Ending Date')
    if  pd.to_datetime(date1)<df["date"].min() or pd.to_datetime(date1)>df["date"].max():
        st.warning("The start date is out of range")
    elif pd.to_datetime(date2)>df["date"].max() or pd.to_datetime(date2)<df["date"].min():
        st.warning("The end date is out of range")
    elif pd.to_datetime(date1) not in df["date"].values:
        st.warning("The start date is unavailable here")
    elif pd.to_datetime(date2) not in df["date"].values:
        st.warning("The end date is unavailable here")
    elif pd.to_datetime(date1)>pd.to_datetime(date2):
        st.warning("The start date is greater than the end date")
    else:
        start_minute=df[df["Datetime"].dt.date==pd.to_datetime(date1).date()].iloc[0]
        end_minute=df[df["Datetime"].dt.date==pd.to_datetime(date2).date()].iloc[-1]
        current_date=df[(df["Datetime"]>=start_minute["Datetime"]) & (df["Datetime"]<=end_minute["Datetime"])]
        source=ColumnDataSource(current_date)
        p = figure(x_axis_type="datetime", title="Line Current Per Minute",width=1000,height=300)
        p.line(x="Datetime", y="LineCurrentIR",source=source, line_width=2, color="red")
        p.line(x="Datetime", y="LineCurrentIY",source=source, line_width=2, color="yellow")
        p.line(x="Datetime", y="LineCurrentIB",source=source, line_width=2, color="blue")
        hover=HoverTool(tooltips=[("Date","@Datetime{%Y-%m-%d %H:%M}"),("LineCurrentIR","@LineCurrentIR"),
                      ("LineCurrentIY","@LineCurrentIY"),
                      ("LineCurrentIB","@LineCurrentIB")],formatters={"@Datetime":"datetime"})
        p.add_tools(hover)
        st.bokeh_chart(p,use_container_width=True)
```



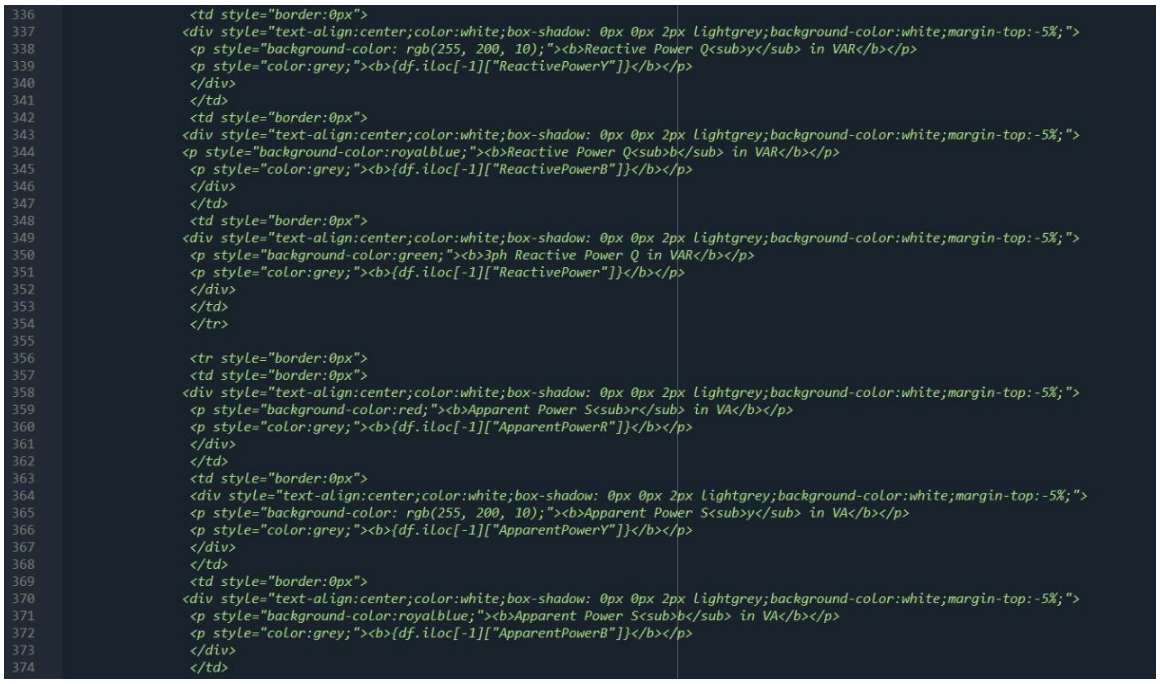
```python
def PowerFactor():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Power Factor Graph</h4>""",unsafe_allow_html=True)
    date1 = st.date_input('Starting Date')
    date2 = st.date_input('Ending Date')
    if  pd.to_datetime(date1)<df["date"].min() or pd.to_datetime(date1)>df["date"].max():
        st.warning("The start date is out of range")
    elif pd.to_datetime(date2)>df["date"].max() or pd.to_datetime(date2)<df["date"].min():
        st.warning("The end date is out of range")
    elif pd.to_datetime(date1) not in df["date"].values:
        st.warning("The start date is unavailable here")
    elif pd.to_datetime(date2) not in df["date"].values:
        st.warning("The end date is unavailable here")
    elif pd.to_datetime(date1)>pd.to_datetime(date2):
        st.warning("The start date is greater than the end date")
    else:
         start_minute=df[df["Datetime"].dt.date==pd.to_datetime(date1).date()].iloc[0]
         end_minute=df[df["Datetime"].dt.date==pd.to_datetime(date2).date()].iloc[-1]
         pf_date=df[(df["Datetime"]>=start_minute["Datetime"]) & (df["Datetime"]<=end_minute["Datetime"])]
         source=ColumnDataSource(pf_date)
         y_values=['-0.91','-0.92','-0.93','-0.94','-0.95','-0.96','-0.97','-0.98','-0.99','1.00',
                   '0.99','0.98','0.97','0.96','0.95','0.94','0.93','0.92','0.91']
         p = figure(x_axis_type="datetime", title="Power Factor Per Minute",width=2000,height=300,
                    y_axis_label="<-----Leading                 Lagging----->",
                    y_range=FactorRange(*y_values))
         p.dot(x="Datetime",y="PowerFactor", size=20,source=source,color="slateblue",line_dash="dotted")
         hover=HoverTool(tooltips=[("Date","@Datetime{%Y-%m-%d %H:%M}"),("PowerFactor","@PowerFactor")]
                         ,formatters={"@Datetime":"datetime"})
         p.add_tools(hover)
         st.bokeh_chart(p,use_container_width=True)
```



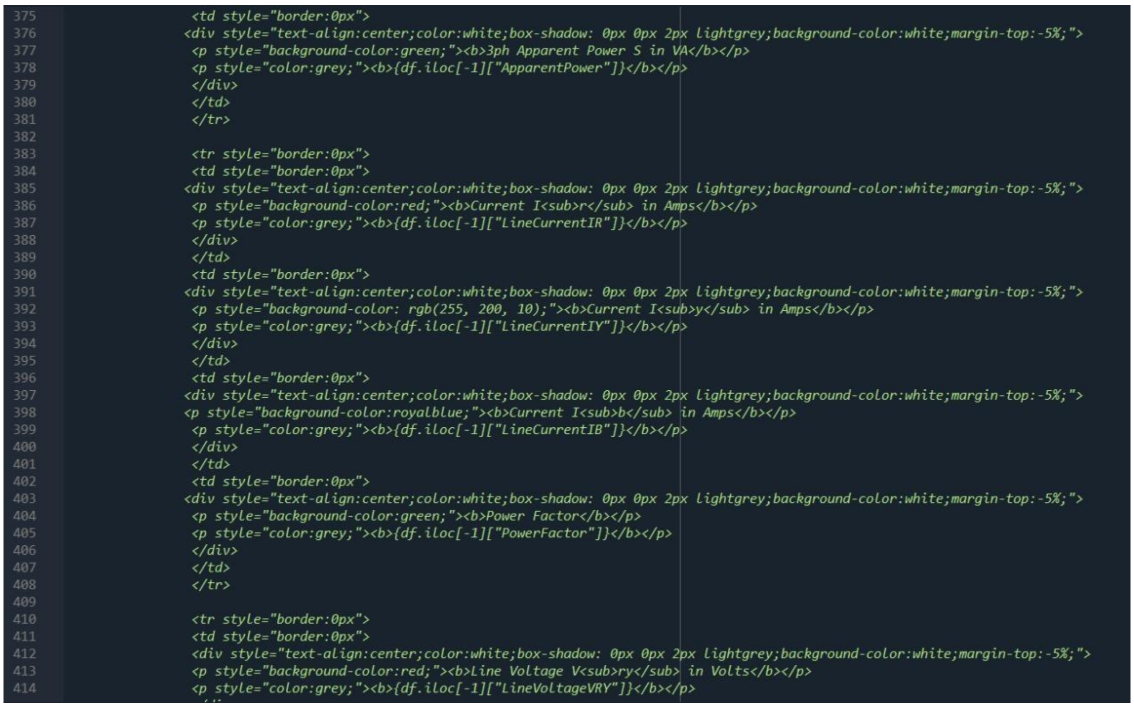
```python
def Frequency():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Frequency Graph</h4>""",unsafe_allow_html=True)
    date1 = st.date_input('Starting Date')
    date2 = st.date_input('Ending Date')
    if  pd.to_datetime(date1)<df["date"].min() or pd.to_datetime(date1)>df["date"].max():
        st.warning("The start date is out of range")
    elif pd.to_datetime(date2)>df["date"].max() or pd.to_datetime(date2)<df["date"].min():
        st.warning("The end date is out of range")
    elif pd.to_datetime(date1) not in df["date"].values:
        st.warning("The start date is unavailable here")
    elif pd.to_datetime(date2) not in df["date"].values:
        st.warning("The end date is unavailable here")
    elif pd.to_datetime(date1)>pd.to_datetime(date2):
        st.warning("The start date is greater than the end date")
    else:
         start_minute=df[df["Datetime"].dt.date==pd.to_datetime(date1).date()].iloc[0]
         end_minute=df[df["Datetime"].dt.date==pd.to_datetime(date2).date()].iloc[-1]
         freq_date=df[(df["Datetime"]>=start_minute["Datetime"]) & (df["Datetime"]<=end_minute["Datetime"])]
         source=ColumnDataSource(freq_date)
         p = figure(x_axis_type="datetime", title="Frequency Per Minute",width=2000,height=300)
         p.vbar(x="Datetime", top="Frequency", width=6000,source=source,color="slateblue")
         hline1=Span(location=52.5,line_dash="dashed",line_color="red")
         hline2=Span(location=47.5,line_dash="dashed",line_color="red")
         p.add_layout(hline1)
         p.add_layout(hline2)
         hover=HoverTool(tooltips=[("Date","@Datetime{%Y-%m-%d %H:%M}"),("Frequency","@Frequency")]
                         ,formatters={"@Datetime":"datetime"})
         p.add_tools(hover)
         st.bokeh_chart(p,use_container_width=True)
```



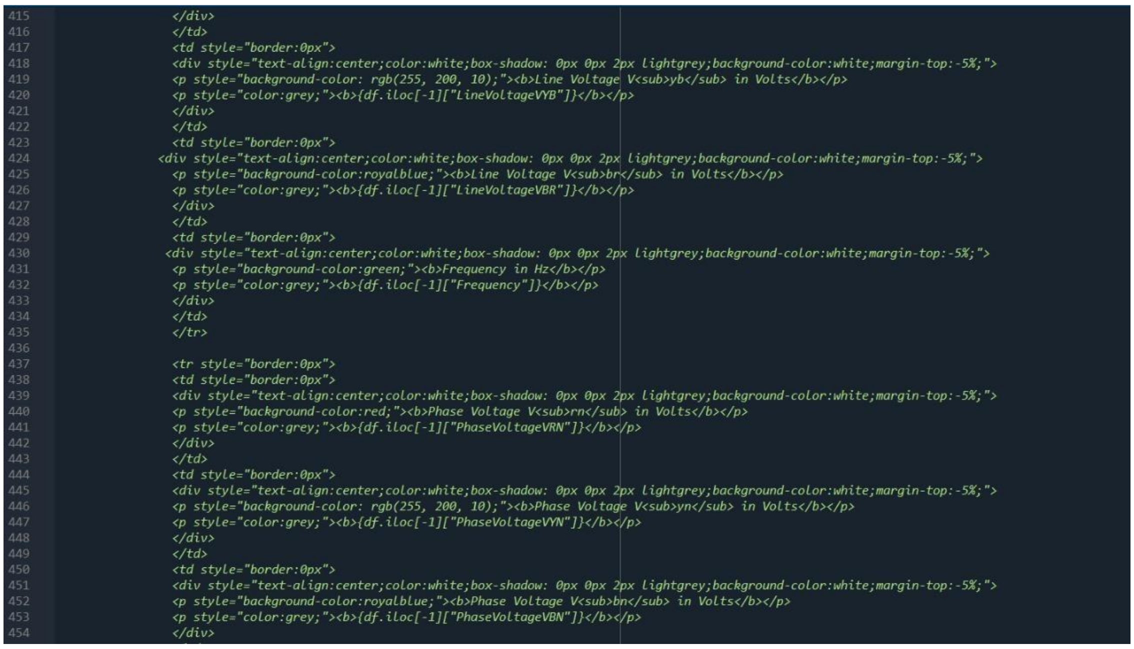
```
def live_data():
    st.markdown(f"""
                <h4 style="margin-left:-20%;margin-top:-10%;">Live Data</h4>
                <i style="margin-left:-20%;">Datetime: {df.iloc[-1]["Datetime"]}</i>
                <table style="width:1000px;border:0px;margin-left:-20%;">
                <tr style="border:0px;">
                <td style="border:0px">
                <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;">
                <p style="background-color:red;"><b>Real Power P<sub>r</sub> in Watts</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["RealPowerR"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white">
                <p style="background-color: rgb(255, 200, 10);"><b>Real Power P<sub>y</sub> in Watts</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["RealPowerY"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
                <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white">
                <p style="background-color:royalblue;"><b>Real Power P<sub>b</sub> in Watts</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["RealPowerB"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white">
                <p style="background-color:green;"><b>3ph Real Power P in Watts</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["RealPower"]}</b></p>
                </div>
                </td>
                </tr>

                <tr style="border:0px">
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
                <p style="background-color:red;"><b>Reactive Power Q<sub>r</sub> in VAR</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ReactivePowerR"]}</b></p>
                </div>
                </td>
```



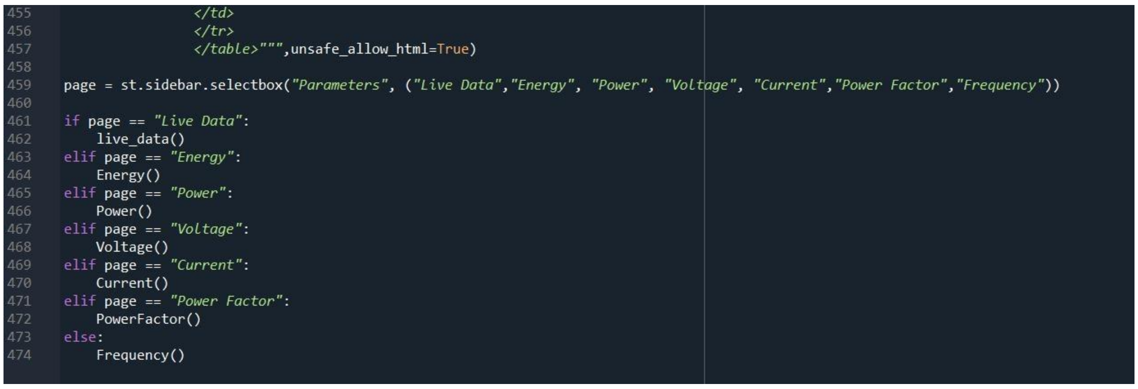
```
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
                <p style="background-color: rgb(255, 200, 10);"><b>Reactive Power Q<sub>y</sub> in VAR</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ReactivePowerY"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
               <p style="background-color:royalblue;"><b>Reactive Power Q<sub>b</sub> in VAR</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ReactivePowerB"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
                <p style="background-color:green;"><b>3ph Reactive Power Q in VAR</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ReactivePower"]}</b></p>
                </div>
                </td>
                </tr>

                <tr style="border:0px">
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
                <p style="background-color:red;"><b>Apparent Power S<sub>r</sub> in VA</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ApparentPowerR"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
                <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
                <p style="background-color: rgb(255, 200, 10);"><b>Apparent Power S<sub>y</sub> in VA</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ApparentPowerY"]}</b></p>
                </div>
                </td>
                <td style="border:0px">
               <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
                <p style="background-color:royalblue;"><b>Apparent Power S<sub>b</sub> in VA</b></p>
                <p style="color:grey;"><b>{df.iloc[-1]["ApparentPowerB"]}</b></p>
                </div>
                </td>
```



```
 <td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:green;"><b>3ph Apparent Power S in VA</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["ApparentPower"]}</b></p>
 </div>
 </td>
 </tr>

 <tr style="border:0px">
 <td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:red;"><b>Current I<sub>r</sub> in Amps</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["LineCurrentIR"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color: rgb(255, 200, 10);"><b>Current I<sub>y</sub> in Amps</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["LineCurrentIY"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
<p style="background-color:royalblue;"><b>Current I<sub>b</sub> in Amps</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["LineCurrentIB"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:green;"><b>Power Factor</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["PowerFactor"]}</b></p>
 </div>
 </td>
 </tr>

 <tr style="border:0px">
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:red;"><b>Line Voltage V<sub>ry</sub> in Volts</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["LineVoltageVRY"]}</b></p>
```



```
 </div>
 </td>
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color: rgb(255, 200, 10);"><b>Line Voltage V<sub>yb</sub> in Volts</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["LineVoltageVYB"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
<div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:royalblue;"><b>Line Voltage V<sub>br</sub> in Volts</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["LineVoltageVBR"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:green;"><b>Frequency in Hz</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["Frequency"]}</b></p>
 </div>
 </td>
 </tr>

 <tr style="border:0px">
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:red;"><b>Phase Voltage V<sub>rn</sub> in Volts</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["PhaseVoltageVRN"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color: rgb(255, 200, 10);"><b>Phase Voltage V<sub>yn</sub> in Volts</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["PhaseVoltageVYN"]}</b></p>
 </div>
 </td>
 <td style="border:0px">
 <div style="text-align:center;color:white;box-shadow: 0px 0px 2px lightgrey;background-color:white;margin-top:-5%;">
 <p style="background-color:royalblue;"><b>Phase Voltage V<sub>bn</sub> in Volts</b></p>
 <p style="color:grey;"><b>{df.iloc[-1]["PhaseVoltageVBN"]}</b></p>
 </div>
```



```
                </td>
                </tr>
                </table>""",unsafe_allow_html=True)

page = st.sidebar.selectbox("Parameters", ("Live Data","Energy", "Power", "Voltage", "Current","Power Factor","Frequency"))

if page == "Live Data":
    live_data()
elif page == "Energy":
    Energy()
elif page == "Power":
    Power()
elif page == "Voltage":
    Voltage()
elif page == "Current":
    Current()
elif page == "Power Factor":
    PowerFactor()
else:
    Frequency()
```