Group Members/Roles:

Dacillo, Nicole – IoT Device Developer

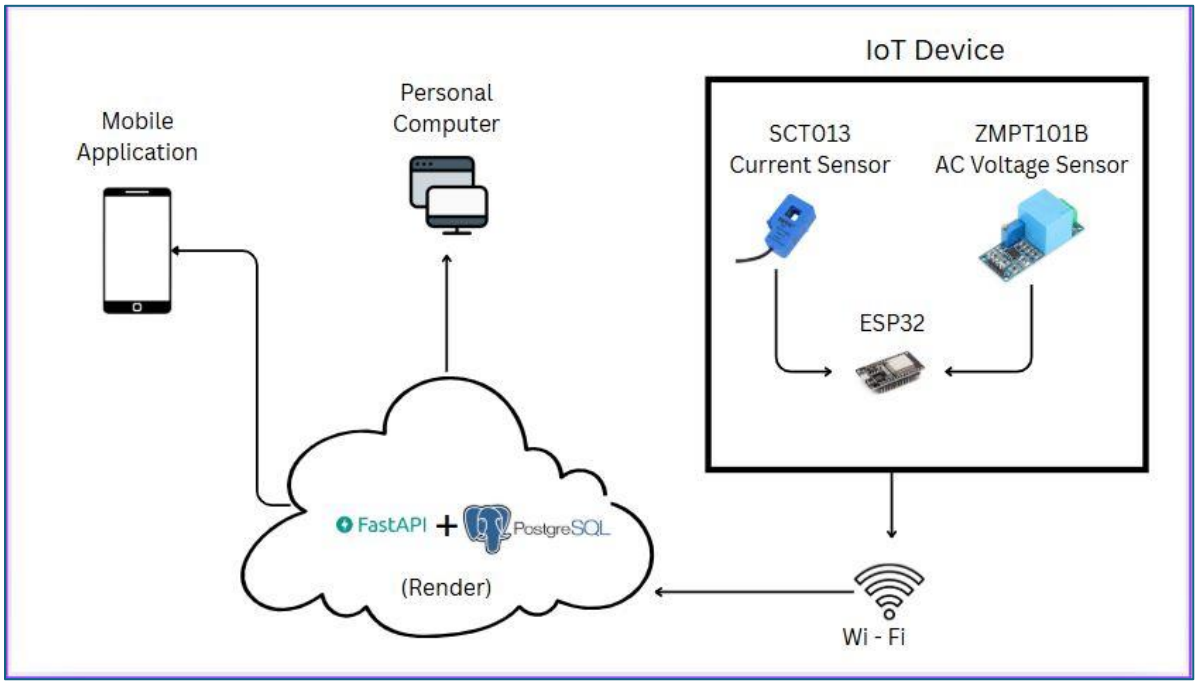
Principio, Joice – Database Administrator

Yamson, Junefree – Web Developer and API Developer

Electricity Consumption Monitoring with IoT Integration

Documentation

I. System architecture diagram



The system architecture describes an IoT-based electricity consumption monitoring system. Current and voltage sensors (SCT013 and ZMPT101B) collect data, which is processed by an ESP32 microcontroller and transmitted wirelessly via Wi-Fi to a cloud server. The server, using FastAPI for data management and PostgreSQL for storage, acts as a central repository. Users access this data through a web application, which is also

accessible via a mobile app. Both interfaces provide real-time monitoring and analytical tools for energy consumption patterns.

II. Challenges faced during development and how they were overcome.

The development of our application presented several significant challenges. Firstly, the entire team was unfamiliar and new with Flutter, leading us to rely heavily on self-directed learning using Youtube tutorials, Flutter documentations, and chatbots or AI-powered assistance, a process in which impacted and take up most of our project timeline and required significant time investment to learn the framework from scratch. Secondly, our team developer struggled on how to make the web and mobile responsive, requiring extensive testing and adjustments. We also struggled to make the app work well on both phones and computers, early versions weren't very mobile-friendly, requiring a complete redesign. Additionally, the app sometimes failed to send data in real-time due to unstable Wi-Fi, needing manual refreshes. We fixed this by adding automatic retries and offline data storage. Thirdly, our IoT Integration also has its set of difficulties, particularly in sensor integration. With three groups sharing limited sensor and device resources, required careful collaboration, meticulous wiring, and reliance on experienced team members to prevent damage and ensure accurate readings. Additionally, the risk of electrical shock during wiring added difficulty and caution to hardware setup. Despite these obstacles, our collaboration with our group members and other groups, individual skills, and with the help of external resources, allowed us to overcome the challenges and achieve significant progress.

III. Technologies and tools used.

IoT Hardware:

- ESP32 Microcontroller
- SCT013 Current Sensor / ZMPT101B Voltage Sensor
- Jumper wires, breadboard, resistors

Backend & API:

- Django Fast API
- Hosted on Render

Frontend:

- Flutter

Database:

- PostgreSQL

Deployment & DevOps:

- GitHub
- Cloud Hosting (Render)

I. Future improvements or features to add.

- Enable users to receive notifications when electricity consumption crosses a defined threshold.

- Add a feature that shows which appliances are consuming the most electricity, helping users identify and manage high-usage devices.

- Equip IoT devices with SD card storage to locally store data in case of network outages, then sync automatically when the connection is restored.

- Strengthen security measures to protect user data and prevent unauthorized access. This could include multi-factor authentication and more robust encryption.

- Integrate smart plugs or power meters that can identify the type of appliance consuming power