Full Stack Developer Task: eGauge Web Application Development

Objective

You have been tasked with developing a web application that interfaces with an eGauge energy meter (specifically, eGauge91511, provided by Eenovators) to allow users to connect to the meter, input a tariff, and generate a bill report based on energy consumption data. The goal is to create a functional, user-friendly web app that demonstrates your full stack development skills, including research, API integration, frontend design, and backend processing. The task must be completed and presented by **June 2, 2025** (4 days from May 29, 2025).

Task Overview

Your objective is to build a web application that:

1. Connects to the eGauge energy meter (eGauge91511) using the provided credentials.
2. Allows users to input a tariff (cost per kWh).
3. Retrieves energy consumption data from the eGauge meter.
4. Generates a bill report based on the consumption data and the user-provided tariff.
5. Presents the results in a clear, professional, and user-friendly interface.

You are encouraged to conduct independent research to understand the eGauge system, its API, and best practices for building the web app. Below, you will find specific details and pointers to guide your work, but you are expected to explore additional resources to ensure a robust solution.

eGauge System Details

- **Meter ID**: eGauge91511
- **Provider**: Eenovators
- **Access Credentials**:
  - **Username**: owner
  - **Password**: Default@123

- **API Access**: The eGauge system provides an API for retrieving energy consumption data. You can access the eGauge API documentation at eGauge API Documentation or by exploring the device's interface at `**http://<meter-ip-or-hostname>/api/**`.

- **Key Information**:

  - The eGauge meter tracks real-time and historical energy consumption data, typically in kWh.

  - Data can be retrieved in formats such as JSON or XML via HTTP requests.

  - The meter supports queries for specific time ranges, which you will need to implement to generate the bill report.

  - Authentication may be required for API access; use the provided credentials to authenticate requests.

- **Research Pointers**:

  - Investigate the eGauge API for endpoints related to retrieving energy usage data (e.g., `/api/register` or `/api/data`).

  - Explore how to handle time-series data for energy consumption.

  - Review best practices for securing API credentials in your application (e.g., avoid hardcoding the username and password).

Task Requirements

Functional Requirements

1. **Web Application:**

  - Develop a web app using modern web technologies (e.g., HTML, JavaScript, CSS for the frontend; Node.js, Python, or similar for the backend).

  - The app should have a clean, responsive user interface built with a framework like React, Vue.js, or similar.

  - Ensure the app is accessible via a web browser and works on both desktop and mobile devices.

2. **Meter Connection**:

  - Implement functionality to connect to the eGauge91511 meter using the provided credentials.

  - Handle API authentication securely and retrieve energy consumption data for a user-specified time period.

3. **Tariff Input**:

- Provide a form for users to input a tariff (cost per kWh, e.g., $0.15/kWh).

- Validate the input to ensure it is a positive number.

4. **Bill Report Generation**:

- Retrieve energy consumption data for a specified period (e.g., daily, weekly, or monthly, as selected by the user).

- Calculate the total cost by multiplying the total kWh consumed by the user-provided tariff.

- Display the bill report in a clear format, including:

  - Total energy consumed (kWh)

  - Tariff used

  - Total cost

  - Time period of the data

- Allow users to download the report as a PDF or view it in the browser.

5. **Error Handling**:

- Handle cases where the meter is unreachable, credentials fail, or API requests return errors.

- Provide user-friendly error messages in the interface.

Non-Functional Requirements

- **Performance**: The app should load quickly and handle API requests efficiently.

- **Security**: Securely manage API credentials and user inputs to prevent vulnerabilities (e.g., XSS, SQL injection).

- **Documentation**: Provide a brief README or documentation explaining how to set up and run the application, including any dependencies.

- **Presentation**: Prepare a short presentation (5-10 minutes) to demonstrate the application, explain your approach, and highlight key features. Include a discussion of your research process and any challenges faced.

Development Guidelines

- **Technology Stack**: Choose a stack you are comfortable with, but ensure it is suitable for a production-grade web application. Recommended technologies include:

  - **Frontend: React**, Vue.js, or Angular with Tailwind CSS or Bootstrap for styling.

-**Backend**: Node.js with Express, Python with Flask/Django, or similar.

- **API Interaction**: Use libraries like `axios` or `fetch` for JavaScript, or `requests` for Python to interact with the eGauge API.

- **Research**: Spend time exploring the eGauge API and documentation. Look for community forums, GitHub repositories, or blog posts that discuss integrating with eGauge meters. For example, search for "eGauge API integration" or "eGauge energy meter web app" to find relevant resources.

- **Testing**: Test the application thoroughly, including edge cases (e.g., invalid tariff inputs, no data available for the selected period).

- **Code Quality**: Write clean, modular, and well-commented code. Follow best practices for version control (e.g., use Git).

Deliverables

1. **Source Code**: A complete, working web application with all source code provided in a Git repository or ZIP file.

2. **Documentation**: A README file or document explaining:

- How to set up and run the application.

- Any dependencies or prerequisites.

- A brief overview of the code structure.

3. **Presentation**: A 5-10 minute presentation (slides or live demo) showcasing the application, your research process, and key features.

4. **Bill Report Example**: Include at least one sample bill report generated by the app, either as a screenshot or a downloadable file.

Submission

- Submit all deliverables by **June 2, 2025, 16:00 PM EAT**.

- Provide a link to the Git repository or a ZIP file containing the source code and documentation.

- Schedule a time to present your work (details to be provided).

Additional Notes

- **Previous Work Reference**: This task builds on a previous project where a similar web app was developed to connect to an eGauge meter and generate bill reports. Use this as inspiration but aim to improve the implementation based on your research and expertise.

- **Resources**:

  - eGauge API Documentation: https://www.egauge.net/docs/

  - eGauge Support: https://www.egauge.net/support/

  - General web development resources (e.g., MDN Web Docs, Stack Overflow) for frontend and backend best practices.

- **Evaluation Criteria**:

  - Functionality: Does the app meet all requirements (connection, tariff input, bill generation)?

  - Code Quality: Is the code clean, modular, and well-documented?

  - User Experience: Is the interface intuitive and responsive?

  - Research Effort: Did you demonstrate an understanding of the eGauge system through independent research?

  - Presentation: Is the demo clear and professional?

We look forward to seeing your solution and your approach to tackling this challenge. Good luck