Methodology

Chapter 1: Data Acquisition and Storage

1.1 Data Collection and Storage Process

The foundation of the data acquisition process is in a Python script that runs in a serverless environment on AWS Lambda. This script runs at a 15-minute interval to access real-time energy consumption and production data from Transelectrica. Using the publicly accessible interface provided by Transelectrica, the script efficiently extracts the data points required for analysis.

Once the data is acquired, it is systematically stored in a MongoDB database. The script is designed to store this information in designated collections in the database structure. A primary collection named "energy_reports" serves as a repository for real-time energy production and consumption data. In addition, separate collections are designated to store predictive data obtained from Transelectrica's consumption and production forecasts, namely "production_energy_forecast" and "energy_forecast_consumption".

In addition to consumption and production data, the script records information on various energy sources. This includes different types of energy such as coal, hydrocarbons, water, nuclear, wind, solar and biomass. The script is configured to capture the proportional contribution of each energy source, thereby facilitating a comprehensive analysis of the energy mix and its environmental impacts.

1.2 Data Structure and Types

The MongoDB database structure is designed to accommodate the various datasets from Transelectrica. Using a NoSQL database framework, it offers the flexibility to accommodate changing data types and structures. The database architecture revolves around collections built to efficiently store and isolate real-time consumption, production, predictive data and energy resource details. These collections are schema-driven, facilitating organized data storage and retrieval.

In the "energy_reports" collection, the data reflects the current energy consumption and production values. Each entry is accompanied by timestamps that allow for temporal analysis and pattern recognition. For predictive data collections ("energy_forecasting_production" and "energy_forecasting_sumption"), a similar approach is used to catalog the forecasted values, allowing comparative analysis with real-time data.

1.4 Environmental Impact Assessment

A key element of the project is a comprehensive assessment of the environmental effects caused by different energy sources. By recording the percentage contribution of carbon, hydrocarbons, water, nuclear energy, wind, solar energy and biomass, quantitative assessment of their ecological footprints becomes feasible. This endeavor is in line with the overall objective to reduce dependence on environmentally harmful energy sources and pave the way for sustainable energy strategies.

Chapter 2: Backend Development and Database Interaction

2.1 Development of Node JS Express App

Creating a Node.js Express application is crucial for seamless interaction with the MongoDB database and effective retrieval of data. This back-end framework is designed to act as an intermediary between the database and front-end components, simplifying data transfer and data management.

Node.js Express includes functions for creating secure connections to the MongoDB database. This includes configuring connection settings, authentication procedures, and leveraging appropriate libraries or middleware for data interaction.

2.2 Handling Data Retrieval

A fundamental aspect of backend development revolves around the creation of robust APIs responsible for facilitating the retrieval, modification and manipulation of data.Using appropriate query mechanisms and efficient data access methods, the Node.js Express application uses designated API endpoints to retrieve real-time energy consumption and production data from the different MongoDB collections.

The GET/energy_reports endpoint serves as a gateway to retrieve real-time energy consumption and production data stored in the "energy_reports" collection. When receiving requests from the frontend, this endpoint facilitates the transmission of instantaneous energy indicators accompanied by timestamps for temporal analysis.

The GET/energy_forecasting_production and GET/energy_forecasting_consumption endpoints are specifically used to access forecasting energy consumption and production data in the respective MongoDB collections. These endpoints allow the frontend to retrieve forecasted energy metrics based on selected dates, facilitating predictive analysis and planning.

When invoked, these GET endpoints efficiently retrieve data in a structured format, ensuring that the information transmitted is organized and formatted according to defined schemas. The acquired data is then sent to the frontend, ready for seamless integration and display within the Angular application.

The designated GET endpoints act as intermediaries between the back-end database and the Angular front-end. They embed the logic to perform database queries, retrieve relevant data, and package the responses into a standard format for easy consumption by front-end components.

Error handling mechanisms are integrated into the back-end infrastructure to reduce potential problems during data retrieval or transmission. Additionally, logging features are embedded to facilitate monitoring and tracking of system activities and potential anomalies.

Chapter 3: Frontend Development and User Interface

3.1 Angular App Development

The frontend aspect of the project is based on the development of an Angular application serving as an admin panel. Angular's comprehensive framework allows you to create a modular and responsive user interface, promoting a seamless user experience.

The application is built around a component-based architecture that divides the user interface into reusable and independent components. This modularity makes the application easy to maintain, reusable and scalable.

The Angular application is connected to the backend Node.js Express APIs, making it easy to retrieve real-time data from MongoDB collections. Specifically it uses APIs designed to access and retrieve real-time energy consumption and production data.

3.2 Real-Time Data Display

Utilizing the ChartJS library, the app incorporates line chart components for visually depicting real-time energy consumption and production trends. These graphs dynamically update as new data is fetched, offering users a comprehensive visualization of the energy metrics.

In addition to displaying consumption and production trends, an imbalance graph is integrated into the UI. This graph illustrates the difference between production and consumption, highlighting instances where surplus energy production could be stored in EV batteries.

Building on the existing two line chart graphs depicting real-time consumption and production imbalances, I have introduced two new indicators below the graphs to enhance the user's understanding of energy dynamics.

The indicator placed on the left showcases the estimated potential energy available for storage. This value corresponds to the surplus energy produced but not consumed in the current day. By subtracting consumption from production, any excess energy that could potentially be stored in EV batteries is visualized, promoting insights into effective energy utilization strategies.

Complementing the display, the indicator on the right computes the value of unsold energy. This calculation is derived by evaluating the surplus energy at a rate of 0.2 euros per kilowatt-hour (kWh) — representing the hypothetical market value of surplus energy that remains unconsumed.

These additional indicators serve as decision-making aids, offering a comprehensive view of energy imbalances. They highlight the potential for energy storage solutions while providing a speculative market value for surplus energy, aiding in strategic considerations for surplus energy utilization or potential market transactions.

3.3 Energy Forecasting Page

Similar to the real-time data display, the forecasting page showcases line graphs portraying forecasted consumption and production trends. These graphs dynamically update based on the selected date, enabling users to visualize future energy metrics.

Complementing the consumption and production predictions, an imbalance graph delineates the projected difference between forecasted production and consumption. This visualization aids in understanding potential energy surplus or deficit scenarios.

3.4 Energy Report Page

The energy report page contains a comprehensive breakdown of the types of energy produced in the last 24 hours. Using data from MongoDB, the Angular application creates a pie chart that illustrates the distribution of coal, hydrocarbon, water, nuclear, wind, solar, and biomass energy sources.

The pie chart offers interactivity, displaying energy values in megawatts (MW) and percentages when you hover over each segment. Also, text labels and corresponding values are shown below the chart for detailed understanding.

Chapter 4: Integration and Functionality

4.1 Integration of Backend and Frontend

The successful integration of backend and frontend components remains a fundamental goal in this phase. The focus is on creating robust connections to facilitate data flow and communication between the Node.js Express backend and the Angular frontend.

The backend APIs developed in Chapter 2 are integrated into the Angular application. RESTful API endpoints act as mediators, allowing real-time energy consumption, production, and resource data to be retrieved and forwarded to the frontend.

Leveraging Angular's data binding mechanisms, the frontend easily connects to backend APIs. This mapping enables real-time and predictive energy indicators to be dynamically displayed and updated within the user interface, promoting an interactive and informative experience for users.

4.2 Usability and User Experience

Angular's framework helps to create a user-friendly interface based on intuitive design principles. The emphasis is on smooth navigation, clear presentation of data and responsive design, fostering an environment conducive to user interaction and understanding.

Efforts are aimed at a clear and comprehensible presentation of energy indicators and forecasts. The use of graphical representations, such as line charts and pie charts, helps users quickly understand consumption, production trends and energy source breakdowns, making interpretation easier.

The application incorporates responsive design elements to cater to various devices and screen sizes. The user interface dynamically adapts to different resolutions, ensuring consistent usability and readability across multiple platforms, including desktops, tablets, and mobile devices.

Interactive components within the user interface promote user engagement and ease of interaction. Features like dropdowns for date selection and hover-over tooltips on charts provide users with additional contextual information, enriching the overall experience.

4.3 Deployment Strategies

The deployment strategy for the developed systems involves leveraging specific platforms tailored to host different components of the project.

The Angular application is deployed using Netlify, providing a hosting environment for the frontend interface. Netlify offers a user-friendly platform for effortless deployment, ensuring accessibility and scalability of the Angular app.

The backend Node.js Express API is hosted on an AWS EC2 instance, enabling robust and scalable deployment. The API is accessed via the URL http://3.79.231.199, and specific endpoints, including /reports for energy reports, /production-forecast for production forecasts, and /consumption-forecast for consumption forecasts, offer access to the stored data.

The EC2 instance serves as the hosting environment for the API, with configurations ensuring continuous availability.An SSH key makes it easy to access your GitHub repository from a Linux terminal inside your EC2 instance, allowing you to easily retrieve and update the code.

The PM2 process manager is used on the virtual machine to maintain API functionality beyond the shutdown of the VM. This ensures that the API file (app.js) continues to run even when the virtual machine is not actively accessed, ensuring uninterrupted service availability.

To address browser security concerns associated with insecure connections, an SSL certificate was acquired from zerossl.com. The SSL certificate was integrated into the Node.js Express app deployed on AWS EC2, enabling the API's transition to a secure HTTPS protocol accessible via https://3.79.231.199, ensuring secure communication between clients and the API.

Implementing SSL effectively resolved potential browser limitations that could have prevented API requests due to insecure connections. By securing the API endpoint with SSL, it meets modern browser security standards, enabling secure and uninterrupted data retrieval from the backend.

Chapter 5: Conclusion and Future Directions

5.1 Summary of Achievements

The culmination of this thesis represents a significant milestone in the utilization of real-time and predictive energy data for grid stabilization and environmental impact assessment. A number of key achievements were achieved during the development and implementation phases:

1. Robust Data Acquisition and Storage: The utilization of a Python script operating on AWS Lambda to fetch real-time energy data from Transelectrica, systematically stored in MongoDB collections. This ensured a structured repository for immediate and predictive energy metrics.
2. Backend and Frontend Development: The successful creation of a Node.js Express backend application and Angular frontend interface facilitated seamless communication between databases and the user interface. ChartJS integration for real-time and predictive data visualization enhanced user insights.
3. Comprehensive Functionality and Usability: A user-friendly admin panel equipped with features to display real-time data, predictive analytics, and energy type breakdowns.

5.2 Future Scope and Enhancements

Future enhancements may include the development and integration of predictive algorithms within the system. By leveraging machine learning or statistical modeling, the project can evolve to offer more accurate forecasting of energy consumption and production.

Future enhancements may also involve the development of a dedicated mobile application, targeting EV owners. This application would aim to offer real-time insights into periods of energy overproduction or underproduction, providing users with valuable information to optimize their EV charging behaviors. It would actively notify users during peak energy production or shortage hours, encouraging them to plug in their EVs, effectively contributing to grid stability by utilizing their vehicle batteries as a temporary energy storage solution. Connected to the user's EV, the application would facilitate participation in supporting the grid during peak hours. Users who plug in their EVs during these critical periods and provide surplus energy to the grid would be rewarded with redeemable points. These points could serve as a form of compensation for the users. Companies supporting the initiative might offer discounts or benefits to users accumulating these points. For instance, discounts at partner stores or the ability to reduce energy costs using acquired points could incentivize and reward active grid support.

Depending on the growth of the project and the participation of the stakeholders, there are many possibilities. The initiative can develop into a robust ecosystem that promotes energy efficiency, encourages sustainable practices, and creates beneficial collaboration between energy consumers and providers.

Expanding the scope by incorporating additional data sources beyond Transelectrica could enrich the system. Integrating data from renewable energy sources, weather patterns, or societal factors can offer comprehensive insights into energy production and consumption patterns.

Continual improvements in the frontend interface can enhance user interaction and experience. Implementing more interactive data visualization tools, customizable user preferences, and responsive design elements can further elevate usability.

Going beyond mere energy type breakdowns, future iterations of the project may incorporate advanced environmental impact assessment tools. This could involve sophisticated models to quantify and visualize the ecological footprint associated with various energy sources.

The ultimate aspiration lies in deploying and implementing the developed system in practical settings. Collaborations with utility providers or governmental agencies could pave the way for the application's real-world utilization, contributing to sustainable energy practices.