COMP5339 Assignment 2

System Description

This system aims to build a real-time streaming processing and visualization platform for electricity generation and carbon emissions data, showcasing the dynamic changes of various power generation facilities in the Australian National Electricity Market (NEM). The system periodically retrieves the latest data from the OpenElectricity API, cleans and integrates it, and then uses the MQTT protocol for message publishing and subscription, providing real-time visualization updates on an interactive map dashboard.

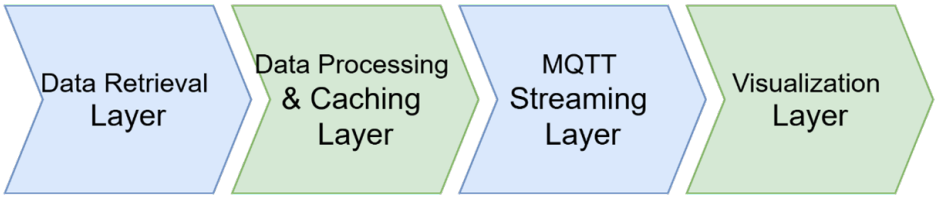


Figure 1: System Architecture

Figure 1 shows the system architecture of this project, and it has four modules:

1. **Data Retrieval Module:** This module retrieves power generation and carbon dioxide emission data in batches using the official OpenElectricity SDK. Performs preliminary parsing and timestamp correction on the returned JSON results to ensure data format and time sequence consistency. The retrieval is performed in a one-time batch mode rather than through periodic API polling.
2. **Data Processing & Caching Module:** Based on `pandas`, this module cleans the raw data, handles missing values, standardizes units, and aggregates data from multiple generator units. The results are stored as structured CSV files to reduce redundant API calls and speed up subsequent access. The generated CSV files uniformly adopt ISO 8601 UTC timestamps to ensure time consistency during the streaming phase.
3. **MQTT Streaming Module: Publishing:** This module sequentially reads cleaned records from the cached CSV files, encapsulates each data point (including power and emission indicators) into a JSON message, and publishes it to the MQTT Broker at fixed intervals following the chronological order, thereby simulating the real-time streaming data push process. **Subscription:** The publisher enforces a 0.1-second interval between messages and checks each message (facility, timestamp) based on the last sent status - only new or updated values are published, and all records are sorted by event time to maintain order. **Continuous Execution:** The streaming media publisher runs in continuous playback loops. After a complete pass in the historical dataset, the system waits for 60 seconds before restarting to simulate continuous operation. Due to only releasing incremental updates and IO being non-blocking, the system runs continuously without any stagnation or crashes. **Optimization:** Retrieve and run once, and concretize into CSV. All subsequent processing, publishing, and visualization are carried out on local data. Only state changes will trigger MQTT messages, and the UI will incrementally update the affected tags instead of redrawing the entire mapping, thereby improving response speed and reducing load.

4. **Visualization Module:** The front-end uses Dash to build an interactive map dashboard, receiving real-time data aggregated via MQTT through the back-end WebSocket, and dynamically refreshing charts and markers. Each facility marker displays the power plant name, energy type, and latest power and emission information, and users can filter and switch by region or fuel type. The dashboard displays the latest market price and network-level demand alongside the map, updating in real time based on incoming MQTT messages.

Data Acquisition

In data acquisition, data collection is based on the official OpenElectricity OEClient. First, the API key is read from environment variables, and time windows, sampling intervals, network status filters, and batch parameters are loaded. The system retrieves a list of NEM network facilities in the states of NSW, QLD, VIC, SA, and TAS. Then, in batches within the set sampling intervals, the system requests the power generation and CO2 emission indicators of each facility. Finally, it obtains the market price and demand for the same period. During implementation, since the official documentation does not fully provide the structure of the "data" field, we validated and aligned the parsing logic by batching sample responses and manually confirming field paths. SDK paths rely on their built-in strongly typed model for parsing, while REST paths are read according to fixed fields after basic validation.

Data Cleaning

To ensure the validity of the disseminated data, we performed the following cleanup:

(1) **Validity check:** Processing starts only when the API returns success=true and data is non-empty. The facility list is validated before extraction, failures are logged, and the facility is skipped to avoid contaminating downstream steps.

(2) **Time consistency check:** A whitelist check is applied to the sampling interval. If the input is invalid or missing, the system rolls back to the default and emits a warning, keeping the timeline and granularity consistent.

(3) **Pre-aggregation outlier handling:** None values are ignored in aggregation, and no additional filtering is applied to NaN outliers. Records missing or not found in the mapping are skipped with a warning to prevent isolated units from entering plant-level totals.

Data Transform

After data cleaning, the system aggregates the cleaned results to form reusable analytical input. First, based on the facility-unit mapping, the power and emission indicators of multiple units under the same power plant are summed at each timestamp to generate a power plant-level time series. The aggregation result is written in JSON format, with metadata such as aggregated_at, aggregation_method, facility_count, and date_range, serving as a unified input for subsequent transformation and analysis. Subsequently, the transformation script reads the aggregated JSON and converts it into various types of CSV files, including a main table of power plant-level time series, a list of facilities and units, and a wide index table facility_metrics_wide.csv. All outputs use a unified column order for direct use in data association and MQTT streaming publishing in Assignment 1.

Data Integration

After cleaning and renovation, the system provides real-time streaming media through multi-level integration of unified data. The collection/aggregation module outputs factory-level JSON, and the conversion module converts it into a structured long table for visualization. Each line

records the power and emissions of a facility at the timestamp and merges the metadata. This wide table (facility-metrics-wide. csv) is used as input for simulating real-time streams. During the publishing process, records are read in order, timestamps are standardized to UTC, and sorted stably by event time and facility code; JSON payloads with core metrics are sent out at fixed intervals. Because facility metadata is pre-integrated, publishers do not require additional connections, and subscribers can directly present messages - integrating the static facility data of task 1 with the streaming indicator of task 2 in terms of structure and time.

Data Subscription

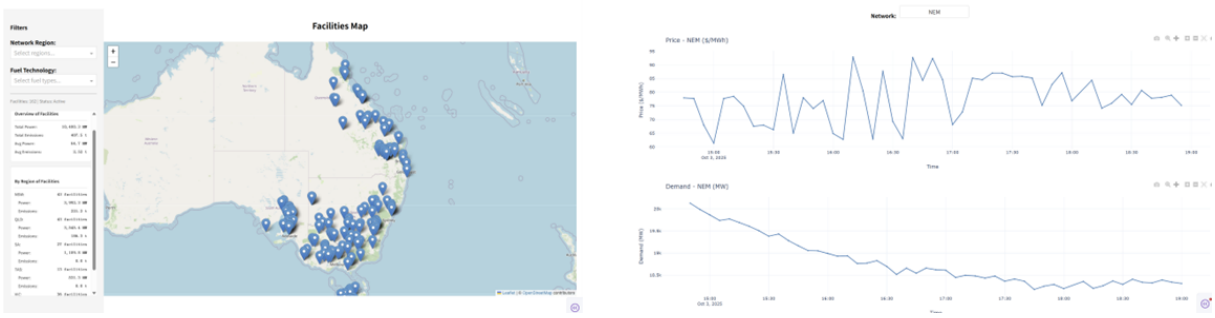
The subscriber connects to the same MQTT broker and listens to the facilities and market metrics topics. The incoming message is decoded from JSON into a structured Python object and stored in a memory state table indexed by (facility id, timestamp). The visualization backend exposes this state to the Dash frontend through WebSocket, enabling incremental UI refresh without the need for complete re-rendering. This allows the dashboard to dynamically reflect real-time changes in facility production and emissions.

Data Visualisation

The system uses a real-time architecture with front-end and back-end separation. Front end: Dash builds the user interface, Dash flyers render facility maps, Plotly draws rolling price/demand maps. Backend: FastAPI (REST+WebSocket) subscribes to MQTT facility/marketplace messages, updates memory status, and then pushes changes to the frontend through WebSocket. When offline, the data is standardized into CSV, and the publisher streams the records to MQTT in chronological order. Merge facility metadata in the backend to form a unified view. The status manager triggers incremental map/chart updates - marking switching power and emissions, and supporting regional/fuel filters. The chart displays the latest prices and demand windows. This creates a low-latency interactive loop that does not directly rely on external real-time APIs.

Dashboard

The visualization dashboard presents the operational status and market dynamics of power generation facilities in Australia's NEM across both spatial and temporal dimensions. One dashboard uses a map to indicate the distribution of power plants, while the other uses a line graph to represent the trends in market prices and demand.



(a) Facilities Map: spatial distribution of generation facilities across NEM regions.

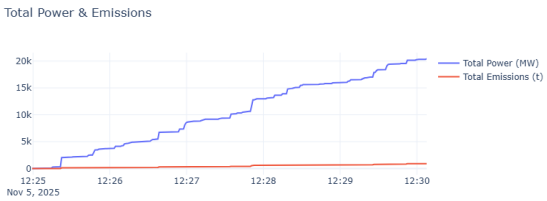
(b) Market Trends: price and demand variation over time.

Figure 2: Dashboard Visualization

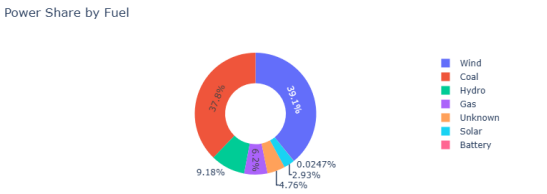
Figure 2 (a) depicts the geographical distribution and real-time status of NEM power facilities. Each blue mark is a plant. Click to display name, fuel type, real-time output, and emissions. The left panel filters by network area and fuel technology, and the results are up-

dated in real-time for dynamic exploration. The following statistical section summarizes the number of factories, total production capacity, total emissions, and average emission intensity by region for quick cross-state comparisons.

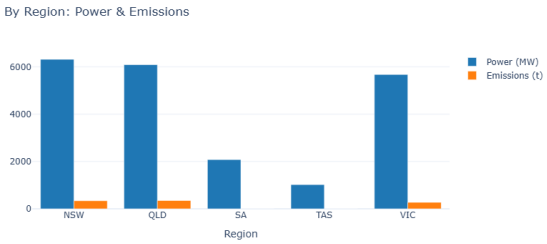
Figure 2(b) shows the price and demand of NEM on the same timeline: the top graph tracks real-time prices, and the bottom graph shows total demand. Prices fluctuate significantly during the window period, while demand overall shows a downward trend. This opposite pattern is typical: when renewable energy production is high and demand is low, oversupply can push prices down or even turn negative. On the contrary, prices rebound at peak or when renewable energy production weakens.



(a) Total Power & Emissions



(b) Power Share by Fuel



(c) By Region: Power & Emissions

(d) Top-10 Facilities: Power & Emissions

Figure 3: Overview of power generation and emissions in the NEM.

To further demonstrate the real-time operating status and emission characteristics of the power system, the system will visualize the power plant-level power and emission information.

Figure 3 (a) shows the changes in total power generation and emissions over time: as the units go online or increase, the power generation increases in a stepped manner, while the emissions fluctuate less, indicating a smaller change in intensity.

Figure 3 (b) shows the power generation combination: wind energy and coal dominate. The shareholding ratios of hydropower, natural gas, and solar energy are relatively small, and the contribution of batteries is the smallest. Overall, supply is still driven by fossil fuels and large-scale renewable energy, with limited access to new energy and energy storage.

Figure 3 (c) compares the regional total: New South Wales and Queensland have significantly higher power generation, reflecting a larger system. Emissions tracking of power generation in various regions indicates that the level is mainly driven by output scale rather than the main differences in clean energy structure.

Figure 3 (d) shows the top ten factories ranked by power generation and their emissions, indicating that high-yield facilities also contribute the most emissions. Although the emission intensity of each factory varies, the overall trend is a positive correlation between production and emissions.

Appendix

Individual Contribution

Kehao Liu: Major contributor to the group's submission. Completed most of the coding tasks and participated in data acquisition, data cleaning, and data visualization. Attended every group meeting.

Ziqi Zheng: Major contributor to the group's submission. Mainly responsible for writing the report, data transformation and analyzing the data by EDA. Attended every group meeting.

Program Start-up Command

1. Full Pipeline Execution: python data_pipeline.py –mode full
2. Data Acquisition Only: python data_pipeline.py –mode –mode acquisition
3. Data Transformation Only: python data_pipeline.py –mode –mode transformation
4. Data Publication Only: python data_pipeline.py –mode –mode publication
5. Execution of dashboard application: python src/dashboard/monolithic_dashboard.py

- Web interface: ‘http://localhost:8050‘ (default Dash port)
- API endpoints: ‘http://localhost:8000‘ (default FastAPI port)
- WebSocket endpoint: ‘ws://localhost:8000/ws‘

More Figures Distribution

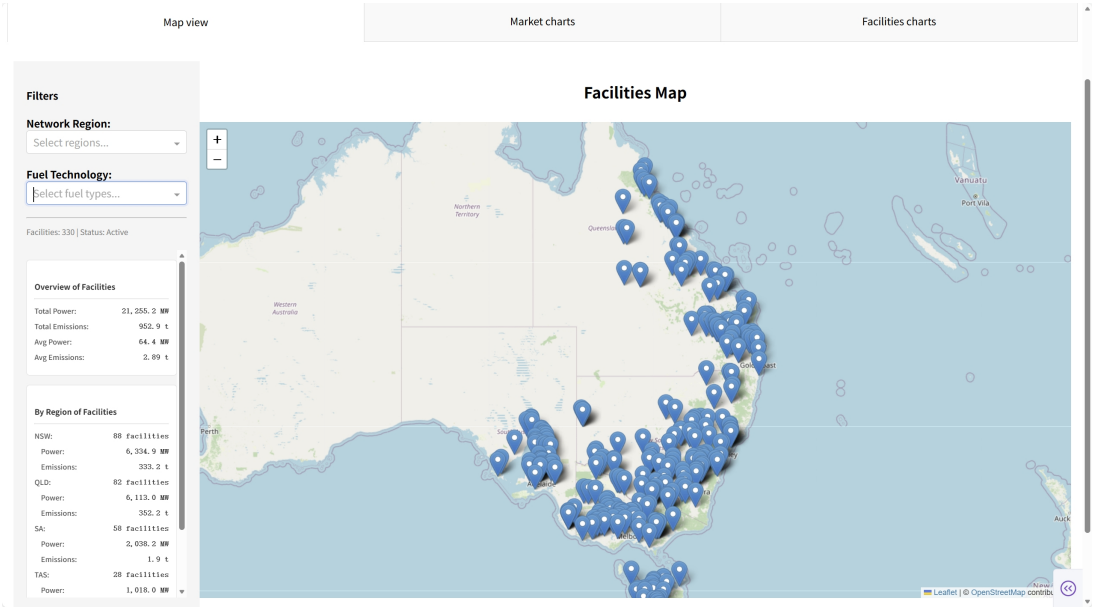


Figure 4: Distribution of all active power generation facilities across Australia.

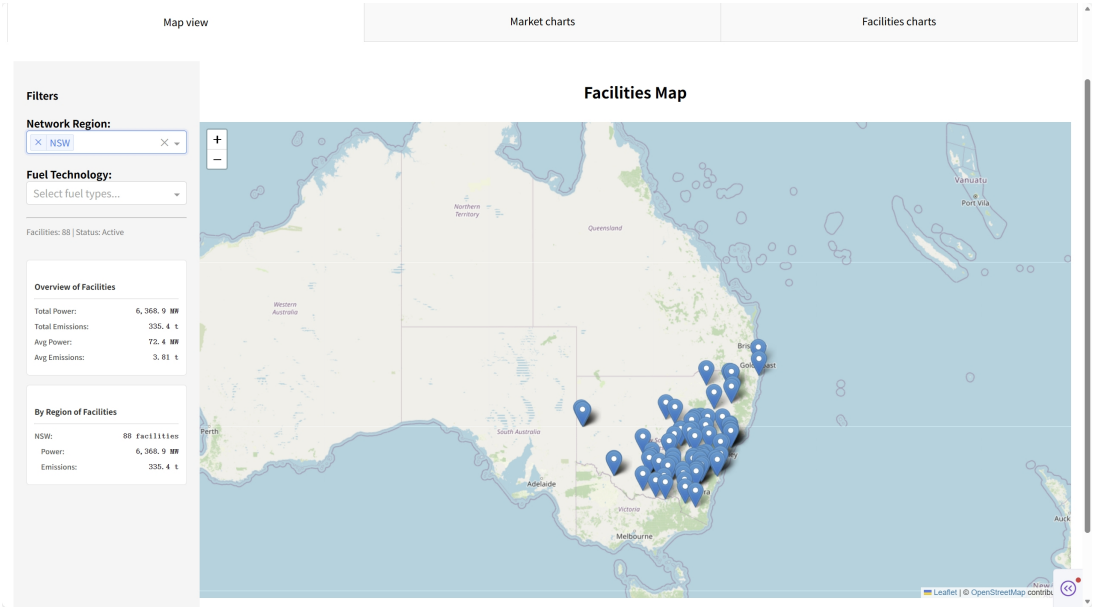


Figure 5: Facility distribution filtered by the NSW network region.

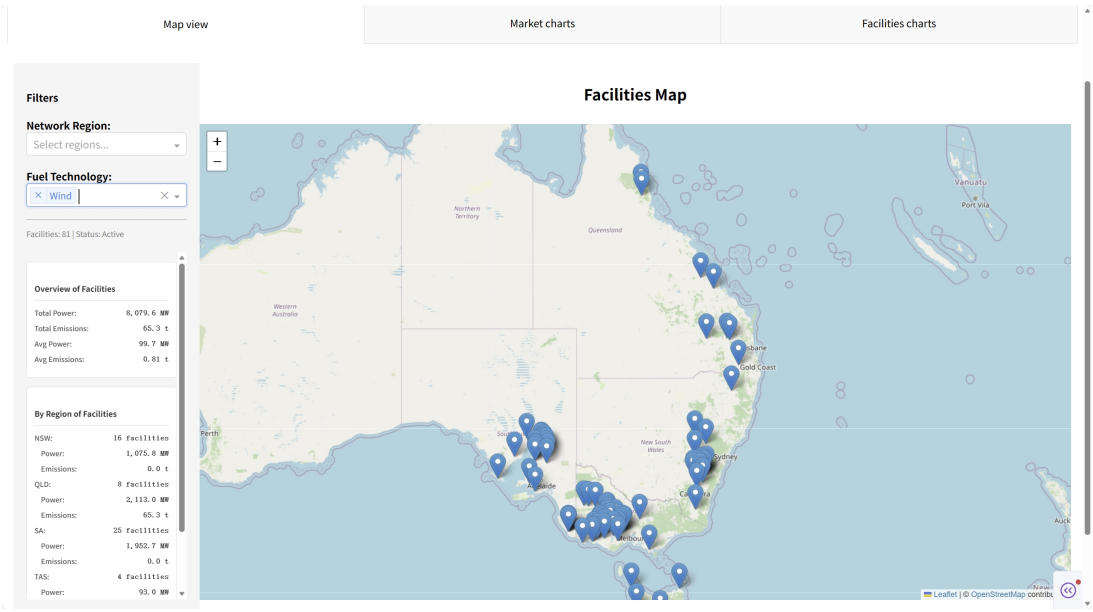


Figure 6: Facility distribution filtered by Wind-based generation technology.

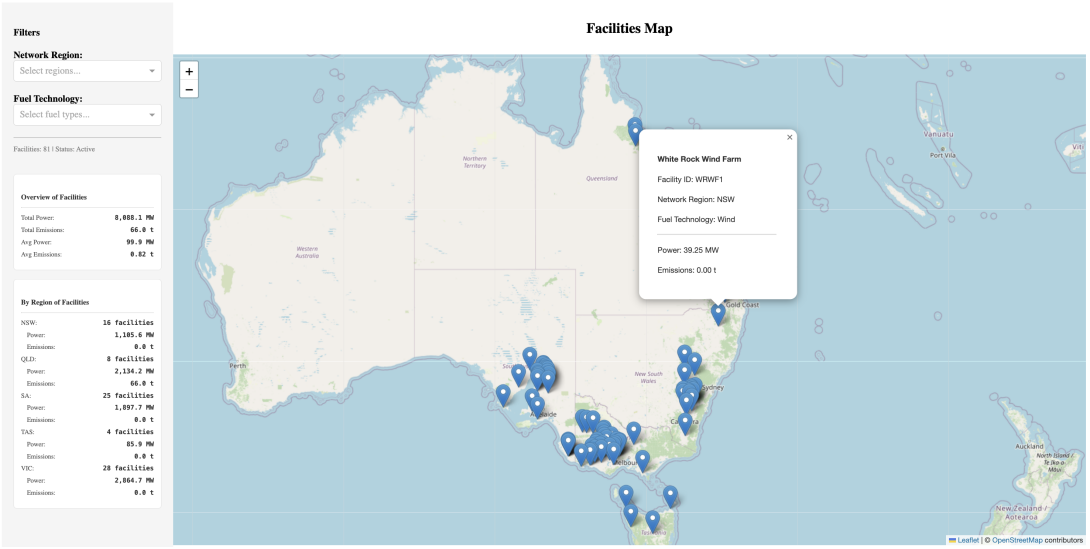


Figure 7: Facility Distribution Map with On-Hover Operational Details.

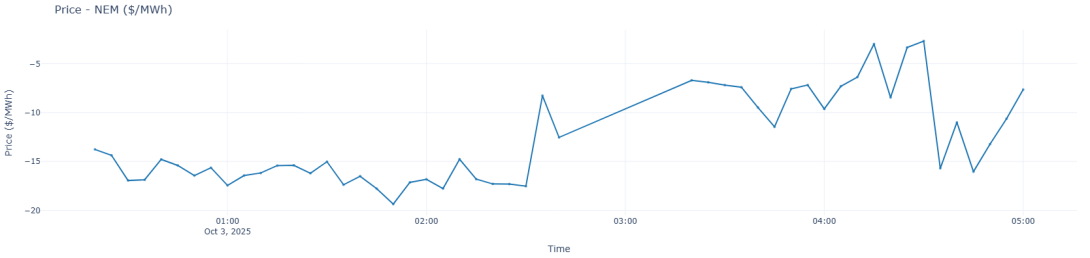


Figure 8: NEM electricity demand over time. The curve shows a gradual increase in demand during the observed period.

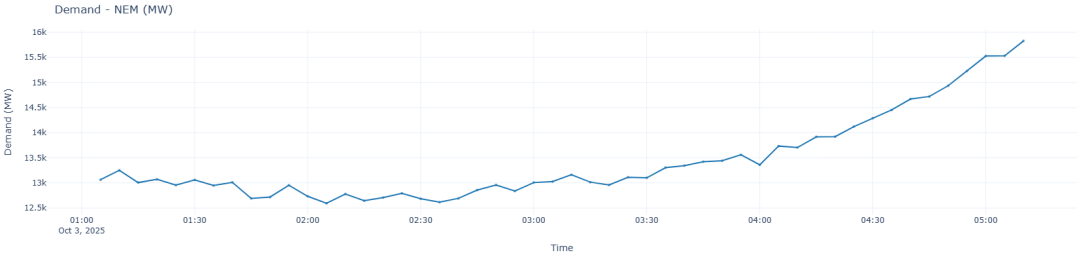


Figure 9: NEM electricity market price over time. Price volatility is observed, including occurrences of negative pricing.