PJM LMP Visualization Application Infrastructure Setup

This document provides supplemental instructions for developers wishing to go beyond a basic repository clone and transition this project into a fully functional analytics tool. To achieve this, users must establish the underlying data infrastructure and external service accounts.

1. Required Accounts & Services

To fully replicate the environment, the following external services are required:

- PJM Data Miner (E-Suite) Required to access wholesale market APIs.
  [PJM Account Setup Link]
- MySQL Database Instance Required for data warehousing (e.g., AWS RDS or local instance).
  [Amazon RDS MYSQL Setup Link]
- MapTiler Cloud Required for vector basemap rendering.
  [MapTiler API Setup Link]

2. Infrastructure & Credentials

Before running the application, users must acquire valid API keys from the services listed above. These credentials must be mapped to the .env file located in the project root directory.
This configuration allows the backend to authenticate external requests and store the massive datasets required for analysis.
Note: Once the credentials are updated in the .env file, the application's internal logic automatically detects and utilizes them.

3. Data Ingestion

Once a MySQL database is connected, it must be populated to power the visualizations.Navigate to the src/data/ directory.

- Execute the Python ingestion scripts (e.g., pjm_query_rt_lmp.py).
- Modify the date ranges within these scripts to fetch the desired historical periods.

These scripts automatically structure and ingest raw API data into the DB tables detailed in src/img/db_schema.png, utilizing the credentials defined in the previous step.

4. Launch

Only after this data foundation is laid is the application ready for interaction. Running the command below in your terminal will launch a dashboard capable of querying real historical data, allowing users to visualize congestion risks and price behavior.

Type `npm run dev` in your terminal to start the App.

On the following page you will find the directory architecture for the application.

PJM LMP Visualization App Directory Structure:

Please refer to the tree structure below to locate the specific configuration files and scripts mentioned in the previous section.

```
project-root/
├── src/
|   ├── components/
|   |   ├── config.js                 # Global constants, API endpoints, and default settings
|   |   ├── controller.js             # The "Brain": Manages app state and orchestrates updates
|   |   ├── filter.js                 # Logic for filtering datasets by zone or time
|   |   ├── math.js                   # Helper functions for color interpolation and calculations
|   |   ├── picker.js                 # Logic handling the date/time range selection inputs
|   |   ├── style.css                 # Custom CSS styling for map overlays and UI elements
|   |   ├── ui.js                     # Manages DOM elements, popups, and sidebar interactions
|   |   └── utils.js                  # General utility functions (formatting dates, cleaning text)
|   ├── data/
|   |   ├── pjm_query_da_lmp.py       # Script to ingest Day-Ahead market prices
|   |   ├── pjm_query_forecast_load.py # Script to ingest load forecast data
|   |   ├── pjm_query_inst_load.py    # Script to ingest instantaneous load data
|   |   ├── pjm_query_metered_load.py # Script to ingest historical metered load
|   |   ├── pjm_query_rt_constraints.py # Script to ingest real-time transmission constraints
|   |   ├── pjm_query_rt_lmp.py       # Script to ingest Real-Time market prices
|   |   ├── pjm_query_temp_set.py     # Script to ingest temperature/weather data
|   |   ├── pjm_transmission_lines.py # Script to ingest transmission line geo-data
|   |   ├── test_connection.py        # Utility to verify database connectivity
|   |   ├── test_pjm_data_retrieval.py # Utility to test PJM API response and ingestion
|   |   └── PJM_zones.geojson         # Geospatial polygons defining PJM load zones
|   ├── img/
|   |   ├── db_schema.png             # Diagram of the SQL database structure
|   |   ├── lmp_icon.png              # Application icon
|   |   ├── observable.png            # Framework logo
|   |   ├── PJM_Map.png               # Screenshot of the main map interface
|   |   └── Query_Tool.png            # Screenshot of the data query interface
|   ├── backend.py                    # FastAPI server handling SQL queries from the frontend
|   ├── index.md                      # Main entry point: Contains the Map View
|   ├── overview.md                   # User Guide: Narrative explanation of market concepts
|   └── picker.md                     # Query Tool: Interface for selecting historical data
├── .env                              # Environment variables (API keys, DB creds) - Ignored
├── .gitignore                        # Specifies files to exclude from version control
├── eda.md                            # Exploratory Data Analysis and preliminary findings
├── lmp_env.yml                       # Conda environment file for Python dependencies
├── observablehq.config.js            # Configuration for the Observable Framework
├── package.json                      # Node.js dependencies and build scripts
├── proposal.md                       # Original academic project proposal
├── README.md                         # Main project documentation and setup guide
└── System_Architecture_and_Deployment_Guide.pdf # Instructions for infrastructure setup and data ingestion
```