

**CHHATTISGARH STATE POWER GENERATION
COMPANY LIMITED**



CSPGCL KORBA (WEST)

A VOCATIONAL TRAINING PROJECT

ON

THERMAL POWER PLANT

(FROM - 04/06/2025 TO 04/07/2025)

ON

HASDEO THERMAL POWER STATION KORBA (WEST)

GUIDED BY:

1. Pratibha Sonwani (SE)
2. Anjlus Tirkey (EE)
3. Kajal Yadav (AE)
4. Sunil Kumar Pradhan (JE)
5. Twinkle Sao (JE)

SUBMITTED BY:

Name: ARUSHI DEWANGAN
Branch: CSE
Semester: 6th
College : KALINGA
INSTITUTE OF INDUSTRIAL
TECHNOLOGY, BBSR

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the authorities of Hasdeo Thermal Power Station, Korba West, for granting me the invaluable opportunity to undertake my vocational training at their esteemed facility. This experience has been instrumental in enhancing my practical knowledge and understanding of the industry.

I am particularly indebted to the MIS-IT Division for their exceptional support and guidance throughout my training period. The engineers and staff members of the division generously shared their expertise, provided insightful explanations, and patiently addressed my queries. Their mentorship not only deepened my technical skills but also inspired me to pursue excellence in my professional journey.

Finally, I am profoundly grateful to my family and friends for their continuous encouragement, understanding, and moral support, which have been my pillars of strength throughout this endeavor.

Name: ARUSHI DEWANGAN

Enrollment No.:

Institute Name: KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY, BBSR

Branch: COMPUTER SCIENCE AND ENGINEERING

INDEX

S.No	TOPICS	PAGE NO.
1.	INTRODUCTION OF CSPGCL KORBA WEST	4,5
2.	OBJECTIVES AND PROBLEM STATEMENT	6
3.	TECHNOLOGY STACK USED	7
4.	SYSTEM ARCHITECTURE	8
5.	DATABASE DESIGN AND DATA MANAGEMENT	9,10
6.	FRONTEND DESIGN AND IMPLEMENTATION	11
7.	DATA ANALYSIS AND VISUALISATION	12,13,14
8.	BACKEND DEVELOPMENT	15,16,17
9.	CONCLUSION AND FUTURE WORK	18
10.	EXPANSION POTENTIAL AND PRACTICAL IMPACT	19
11.	REFERENCES	20

INTRODUCTION OF CSPGCL KORBA WEST

The Hasdeo Thermal Power Station, also known as the Korba West Thermal Power Station, is a significant coal-fired power facility located in Korba, Chhattisgarh, India. Owned and operated by the Chhattisgarh State Power Generation Company Limited (CSPGCL), this plant plays a vital role in meeting the energy demands of the region.

Historical Development:

Commissioned between 1983 and 1986, the original setup of the Hasdeo Thermal Power Station comprised four units, each with a capacity of 210 MW, totaling 840 MW. These units were instrumental in bolstering the state's power infrastructure during that period.

Commissioning Dates:

Unit 1 – June 21, 1983

Unit 2 – March 31, 1984

Unit 3 – March 26, 1985

Unit 4 – March 13, 1986

Expansion and Modernization:

In response to growing energy needs, a significant expansion was undertaken with the commissioning of a 500 MW unit in 2013, known as the Korba West Extension. This addition increased the plant's total capacity to 1,340 MW.

Further modernization efforts are underway. In March 2025, Bharat Heavy Electricals Limited (BHEL) secured an order from CSPGCL to set up a 2x660 MW supercritical thermal power project at the Hasdeo site. This project aims to replace the older 4x210

MW units with more efficient and environmentally friendly technology, incorporating advanced flue gas desulphurization systems to control emissions.

Fuel and Environmental Considerations:

The power station sources its coal primarily from the nearby Kusmunda coal mine, operated by South Eastern Coalfields Limited (SECL). While the plant is crucial for meeting the region's energy needs, its geographical proximity to sensitive habitats like the Lemru Elephant Reserve is an elephant corridor. This proximity necessitates careful environmental management and sustainable operational practices to mitigate potential ecological impacts.

In summary, the Hasdeo Thermal Power Station has evolved from its initial setup in the 1980s to a modern facility incorporating advanced technologies to meet contemporary energy and environmental standards.



Figure 1: Hasdeo Thermal Power Station, Korba West

OBJECTIVES AND PROBLEM STATEMENT

OBJECTIVES:-

The primary objectives of this project are:

1. To enable effective data analysis and visualization by providing users with interactive tools to explore, interpret, and gain insights from operational datasets, with a particular emphasis on short-term trends and time series patterns.
2. To facilitate easy identification of hidden trends and patterns through dynamic dashboards and visual representations, empowering users to make informed decisions based on recent data.
3. To design a user-friendly web application that allows intuitive data selection—such as choosing specific dates via a calendar interface—for targeted analysis.
4. To integrate backend and frontend technologies that ensure seamless data retrieval, processing, and presentation, making the analytics accessible and actionable for end users.
5. To provide automated suggestions and highlights based on analyzed data, enhancing the user's ability to quickly understand key findings and potential actions.

PROBLEM STATEMENT:

Plant employees log operational data in spreadsheets, but lack a dedicated, user-friendly platform for visualizing trends and extracting insights. This makes it hard to quickly spot patterns or make data-driven decisions.

How Is This Different from Traditional Excel Charts?

- Excel charts are static and require manual updates.
- No automated insights or trend detection—users must interpret everything themselves.
- Comparing multiple KPIs or time periods is cumbersome and often involves juggling several files.
- No central dashboard for all key metrics.
- The proposed web app solves these issues by providing:
- Interactive, automated visualizations that update instantly.
- Built-in analytics like calendar filtering, multi-metric dashboards, and suggestive recommendations.
- Centralized access to all KPIs in one place.
- Easy navigation and quick trend comparisons.

This empowers employees to move beyond static charts and gain fast, actionable insights for better decision-making.

TECHNOLOGY STACK USED

Frontend:-

HTML5: Structured the content and layout of the web application.

CSS3: Provided styling and ensured a responsive, visually appealing interface.

JavaScript: Enabled interactivity and dynamic behavior on the client side.

Chart.js: Integrated for rendering interactive and customizable data visualizations within the browser.

Backend:-

Node.js: Served as the runtime environment for executing server-side JavaScript code.

Express.js: Used as the web application framework to handle routing, API creation, and server logic.

Database:-

MySQL: Managed the storage, retrieval, and organization of operational data in a relational database format.

Git: Used for version control and public code hosting.

Technology Stack at a Glance



Figure 2: Technology Stack at a Glance

SYSTEM ARCHITECTURE

The system architecture of this web application is based on a robust three-tier model, ensuring clear separation of concerns and efficient data flow between components. The architecture comprises the following layers:

1. Presentation Layer (Frontend):

The frontend is developed using HTML, CSS, JavaScript, and Chart.js. This layer is responsible for rendering the user interface, including interactive elements such as the calendar and dynamic data visualizations. It manages all user interactions and communicates with the backend server to request and display data.

2. Application Layer (Backend):

The backend is implemented with Node.js and Express.js. It acts as the intermediary between the frontend and the database, handling all business logic, processing user requests, and managing API endpoints. The backend receives input from the frontend, performs necessary computations or analytics, and retrieves relevant data from the database.

3. Data Layer (Database):

MySQL serves as the relational database for the application. It stores all operational data in a structured format, enabling efficient querying and data management. The backend interacts with the database to fetch, insert, or update data as required by user actions.

Data Flow:

When a user selects a specific date on the calendar interface, the frontend sends a request to the backend server. The backend processes this request, queries the MySQL database for the relevant data, performs any required analysis, and returns the results to the frontend. The frontend then visualizes this data for the user through interactive charts and dashboards.

This layered architecture enhances the modularity, scalability, and maintainability of the application, allowing each component to be developed, tested, and updated independently.

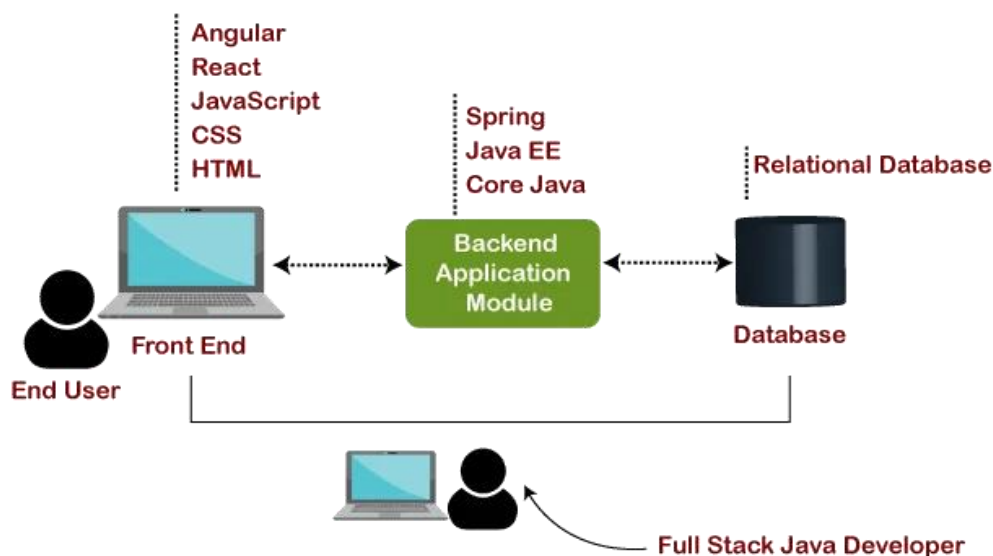


Figure 3: System Architecture of the Web Application

DATABASE DESIGN AND DATA MANAGEMENT

Overview

The database design and data management process form the backbone of the application, ensuring that operational data is accurately stored, efficiently retrieved, and ready for analysis and visualization. The system uses a structured relational database (MySQL) to manage key performance indicators (KPIs) derived from raw daily performance reports (DPRs).

Data Cleaning and Preparation

Raw Data Extraction:

The initial data was sourced from raw DPRs, which often contained unstructured or inconsistent entries.

KPI Selection:

To focus the analysis, essential KPIs were identified, such as:

- Generation (MU)
- Coal Consumption (MT)
- Heat Rate (kcal/kWh)
- Specific Coal Consumption (kg/kWh)

Data Cleaning:

The raw data was cleaned and standardized in **Excel**. This involved:

1. Removing irrelevant or duplicate entries
2. Correcting inconsistencies in date formats and units
3. Handling missing or anomalous values
4. Structuring the data into columns for each KPI

First Cleaned Dataset:

The cleaned and organized data was saved as a new Excel sheet, forming the basis for database import and further analysis.

Database Schema Design:-

Table Structure:

A single main table, `cleaned_cspgcl`, was created with the following fields:

Date (DATE): The operational date

Generation (MU) (FLOAT): Daily power generation

Coal Cons. (MT) (FLOAT): Daily coal consumption

Heat Rate (kcal/kWh) (FLOAT): Daily heat rate

Spec. Coal (kg/kWh) (FLOAT): Specific coal consumption

Data Import and Management

Import Process:

The cleaned Excel sheet was imported into the MySQL database using standard import tools or scripts, ensuring all KPIs were mapped to their respective columns.

Data Validation:

After import, sample queries and checks were performed to verify data integrity and consistency.

Ongoing Data Management:

New data can be appended to the table following the same cleaning and validation process.

The database supports efficient querying for date-based analysis and visualization.

Data Access and Security

Backend Integration:

The backend server connects to the MySQL database using secure credentials managed via environment variables.

Query Optimization:

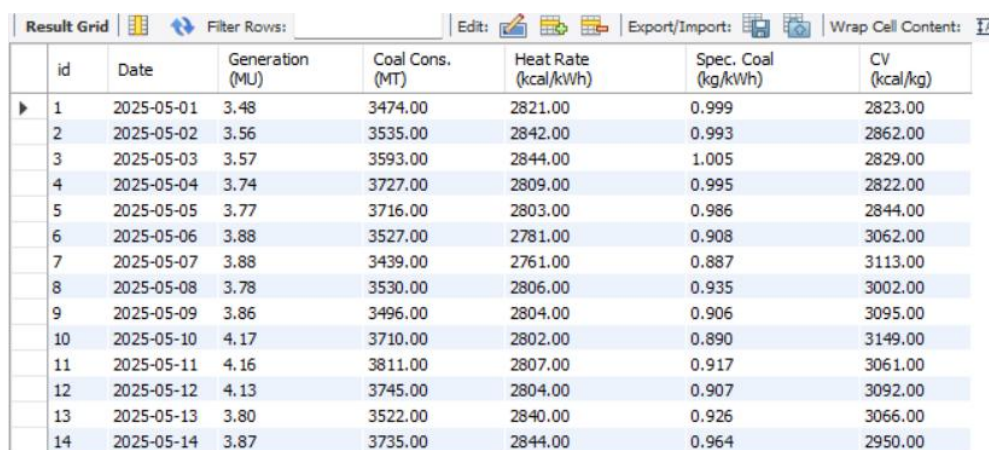
Indexes on the **Date** field and other relevant columns ensure fast retrieval for analytics and visualization.

Data Security:

Access to the database is restricted to authorized backend processes, and sensitive information is not exposed to the frontend.

Summary

The database design and data management workflow ensures that only clean, relevant, and well-structured data is used for analysis. By carefully selecting KPIs, cleaning raw DPRs, and implementing a robust relational schema, the system provides a reliable foundation for accurate analytics and visualization.



	id	Date	Generation (MU)	Coal Cons. (MT)	Heat Rate (kcal/kWh)	Spec. Coal (kg/kWh)	CV (kcal/kg)
▶	1	2025-05-01	3.48	3474.00	2821.00	0.999	2823.00
	2	2025-05-02	3.56	3535.00	2842.00	0.993	2862.00
	3	2025-05-03	3.57	3593.00	2844.00	1.005	2829.00
	4	2025-05-04	3.74	3727.00	2809.00	0.995	2822.00
	5	2025-05-05	3.77	3716.00	2803.00	0.986	2844.00
	6	2025-05-06	3.88	3527.00	2781.00	0.908	3062.00
	7	2025-05-07	3.88	3439.00	2761.00	0.887	3113.00
	8	2025-05-08	3.78	3530.00	2806.00	0.935	3002.00
	9	2025-05-09	3.86	3496.00	2804.00	0.906	3095.00
	10	2025-05-10	4.17	3710.00	2802.00	0.890	3149.00
	11	2025-05-11	4.16	3811.00	2807.00	0.917	3061.00
	12	2025-05-12	4.13	3745.00	2804.00	0.907	3092.00
	13	2025-05-13	3.80	3522.00	2840.00	0.926	3066.00
	14	2025-05-14	3.87	3735.00	2844.00	0.964	2950.00

Figure 4: Successfully Fetched and Cleaned Data Displayed in Database Viewer

FRONTEND DESIGN AND IMPLEMENTATION

Overview

The frontend of the web application is designed to provide users with an intuitive and interactive platform for analyzing operational data. Built using HTML, CSS, JavaScript, and Chart.js, the interface allows users to easily select dates, view key metrics, and interpret trends through dynamic visualizations.

User Interface Layout

The main interface consists of:

Navigation Bar: Displays the application title and provides a consistent header.

Calendar Widget: Allows users to select specific dates for data retrieval and analysis.

Performance Tabs: Users can switch between different analytic views such as Performance, Efficiency, and Analysis.

Data Visualization Area: Key metrics (like Generation Trend and Coal Consumption) are visualized using interactive charts.

Selected Day Info Panel: Displays detailed statistics for the chosen date, including generation, coal usage, heat rate, and specific coal consumption.

Suggestive Measures & Analysis: Offers users the ability to select questions and receive automated suggestions or analyses based on the data.

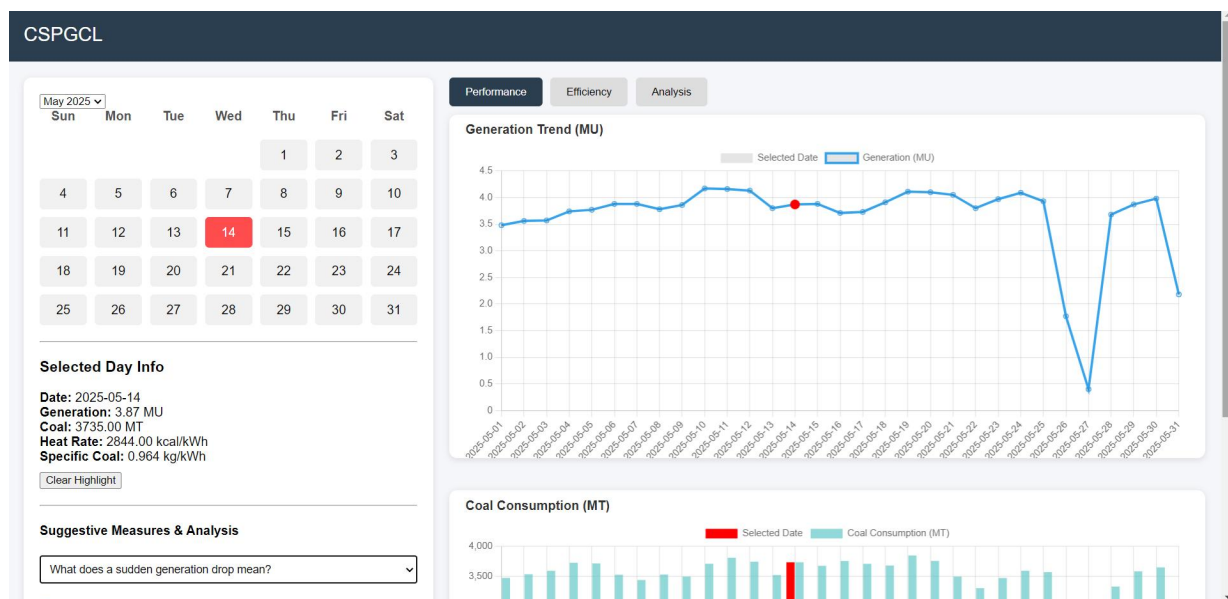


Figure 5: Main Dashboard Interface of the Data Visualization Web Application

Suggestive Measures & Analysis

What does a sudden generation drop mean? ▾

-- Select a Question --

Suggestive Measures

- How can I improve a high Heat Rate?
- What causes consistently low generation?
- What to do about high coal consumption?
- What does a sudden generation drop mean?

Data Analysis

- Which day had the highest generation?
- Which day was most fuel-efficient?
- What was the average heat rate?
- What was the total coal consumed?

Suggestive Measures & Analysis

What does a sudden generation drop mean? ▾

A large, sudden generation drop almost always signifies a Unit Trip (an automatic protective shutdown). Review the station's event logs for that date to find the root cause (e.g., boiler, turbine, or generator fault).

Figure 6 : Suggestive Measures & Analysis Panel

Displays the application's interactive suggestion feature, where users can select a question and receive automated explanations or recommendations based on operational data.

DATA ANALYSIS AND VISUALISATION

Overview

Data analysis and visualization are at the core of this project, transforming raw operational data into actionable insights through systematic examination and interactive visual outputs. The goal is to help users identify trends, patterns, and anomalies in the data, supporting informed decision-making.

Data Analysis Approach :-

Data Preparation:

The collected operational data is first cleaned and organized in the MySQL database to ensure accuracy and consistency. This step includes handling missing values, validating data types, and structuring records for efficient querying and analysis.

Descriptive Analysis:

The application computes key summary statistics such as daily totals, averages, and trends for metrics like power generation and coal consumption. This helps users understand the overall distribution and central tendencies in the dataset.

Trend and Comparative Analysis:

By enabling users to select specific dates or periods, the system allows for comparative analysis across different timeframes. Users can observe fluctuations, peaks, and anomalies in operational performance.

Automated Insights:

The suggestion feature analyzes the selected data and provides automated recommendations or highlights, helping users quickly interpret results and take action.

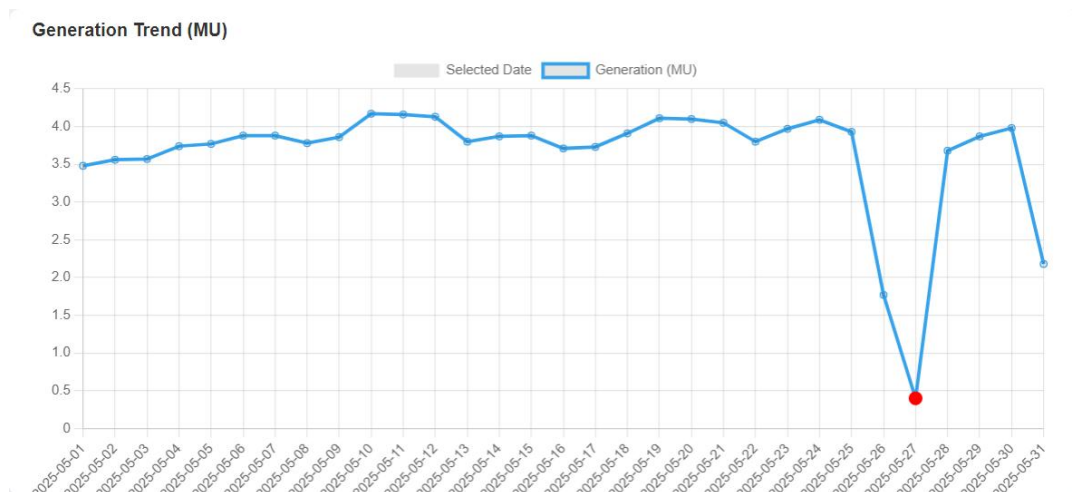


Figure 7: Generation Trend (MU) Chart

Displays the daily power generation trend, highlighting patterns and anomalies over time.

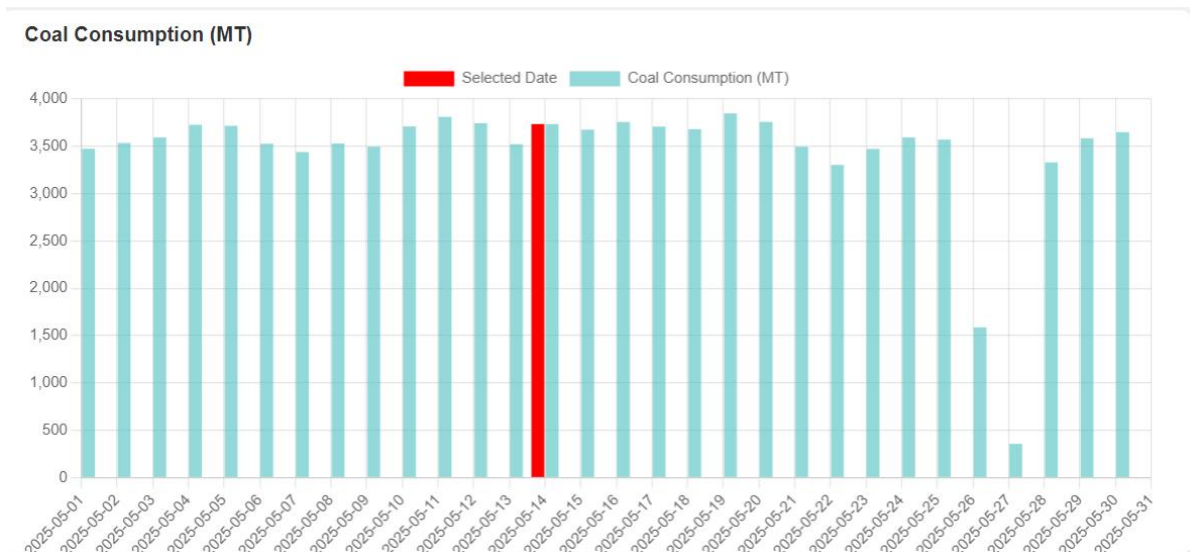


Figure 8: Coal Consumption (MT) Visualization
Shows daily coal usage, enabling users to monitor resource efficiency.

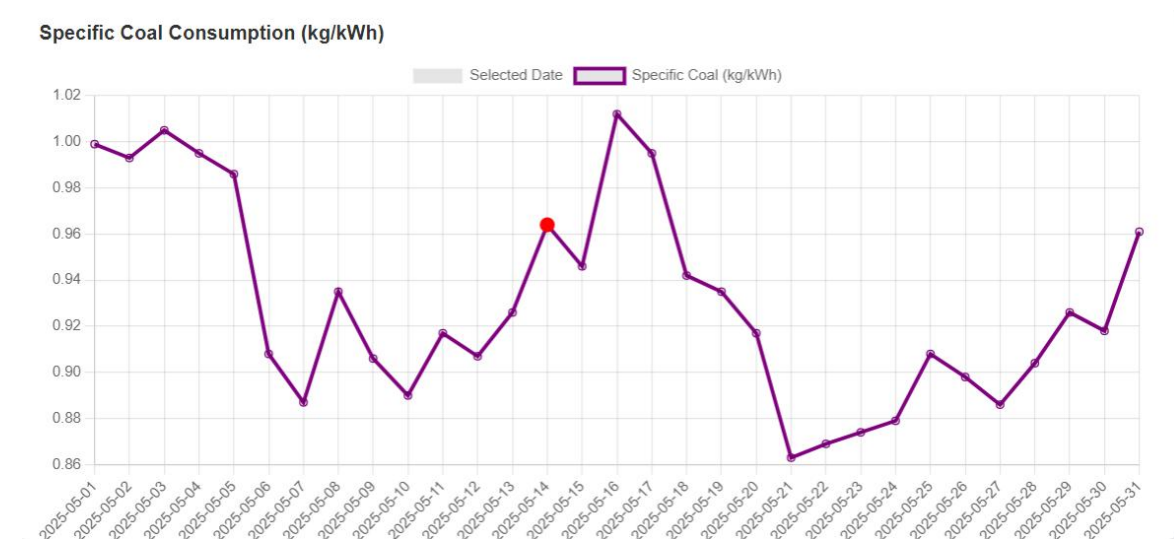


Figure 9: Heat Rate Trend (kcal/kWh)
Displays the daily heat rate trend, highlighting efficiency fluctuations and selected date emphasis



Figure 10: Specific Coal Consumption (kg/kWh)
Shows the daily specific coal consumption, illustrating variations and the highlighted selected date.

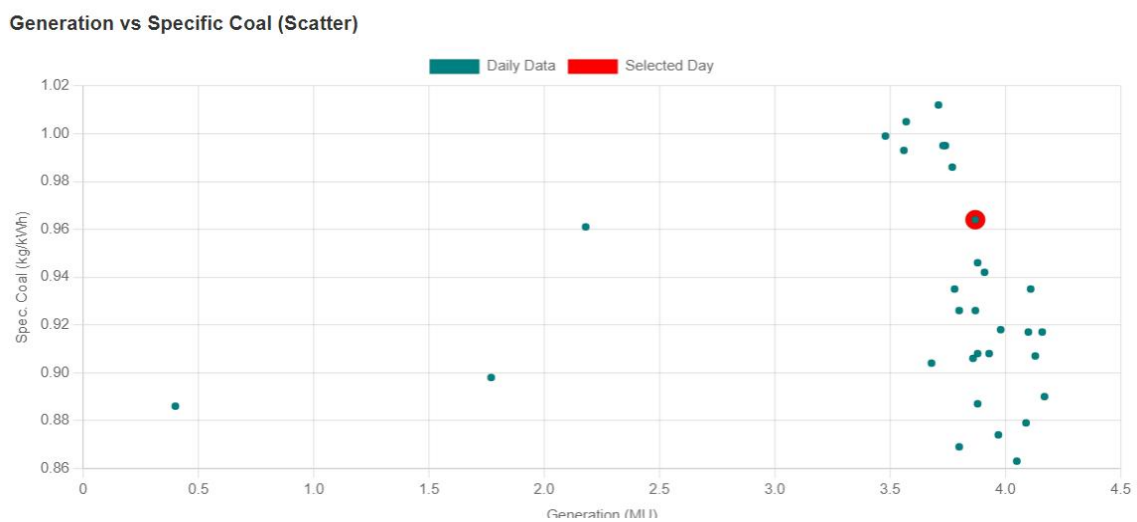


Figure 11: Generation vs Specific Coal Consumption (Scatter Plot)
Visualizes the relationship between daily power generation and specific coal consumption, with the selected day marked distinctly.

BACKEND DEVELOPMENT

The backend of the web application is designed to efficiently handle data processing, business logic, and communication between the frontend interface and the database. Built using Node.js and Express.js, the backend ensures secure, reliable, and scalable operations.

Architecture and Key Components:-

Server Implementation:

The backend server is implemented in Node.js using the Express.js framework. It listens for HTTP requests from the frontend, processes them, and returns appropriate responses.

API Endpoints:

RESTful API endpoints are created to handle various operations such as retrieving operational data for selected dates, processing analytics, and serving suggestion outputs. These endpoints ensure structured and secure data exchange between the frontend and backend.

Database Integration:

The backend connects to a MySQL database, executing queries to fetch, insert, or update operational records as required. Parameterized queries and proper error handling are used to maintain data integrity and security.

Environment Configuration:

Sensitive information such as database credentials is managed using environment variables stored in a .env file, following best security practices.

Project Structure:

The backend codebase is organized for maintainability, with separate files for server configuration (server.js), route handling, and utility functions

Below is a screenshot of the backend project structure and a portion of the code as seen in Visual Studio Code. This illustrates the organization of files and demonstrates hands-on development:

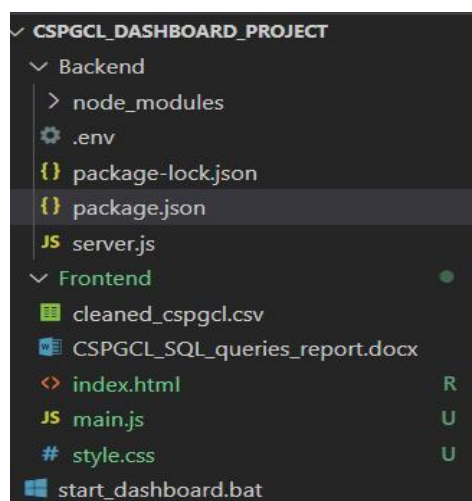


Figure 12: Visual Studio Code editor showing the organized backend project directory and main server file (server.js), demonstrating hands-on backend development and code structure

Environment Configuration and Server Initialization

The backend server is initialized in `server.js`, where environment variables are loaded using the `dotenv` package to securely manage sensitive credentials. Express.js is used to create the server, and CORS is enabled to allow cross-origin requests from the frontend.

```
1 // At the very top, load the variables from the .env file
2 require('dotenv').config();
3
4 const express = require('express');
5 const mysql = require('mysql2');
6 const cors = require('cors');
7
8 const app = express();
9 // Use the PORT from the .env file, or default to 3000
10 const port = process.env.PORT || 3000;
11
12 // Enable CORS for all origins
13 app.use(cors());
```

Database Connection

A connection to the MySQL database is established using credentials from the `.env` file. The application exits gracefully if the connection fails, ensuring robust error handling.

```
14
15 // Create MySQL connection USING the variables from .env
16 const connection = mysql.createConnection({
17   host: process.env.DB_HOST,
18   port: process.env.DB_PORT,
19   user: process.env.DB_USER,
20   password: process.env.DB_PASSWORD,
21   database: process.env.DB_NAME
22 });
23
24 // Connect to MySQL
25 connection.connect(err => {
26   if (err) {
27     console.error('✖ MySQL connection error:', err);
28     // If the database connection fails, the app can't do anything.
29     // It's often better to exit gracefully.
30     process.exit(1);
31   }
32   console.log('✔ Connected to MySQL!');
33 });
34
```

API Endpoint Example

A RESTful API endpoint is defined to fetch operational data from the database, returning the results as JSON to the frontend.

```
34
35 // API endpoint to fetch data
36 app.get('/api/data', (req, res) => {
37   const query = `
38     SELECT
39       DATE_FORMAT(Date, '%Y-%m-%d') AS Date,
40       \Generation (MU)\`,
41       \Coal Cons. (MT)\`,
42       \Heat Rate (kcal/kWh)\`,
43       \Spec. Coal (kg/kWh)\`
44   FROM
45     cleaned_cspgcl
46   ORDER BY
47     \Date\`
48   `;
49
50   connection.query(query, (err, results) => {
51     if (err) {
52       console.error('✖ SQL QUERY FAILED:', err);
53       res.status(500).json({
54         message: 'Failed to execute database query.',
55         error: err.sqlMessage
56       });
57       return;
58     }
59     res.json(results);
60   });
61 });
62
```


Server Startup

The server listens on the specified port and logs a message when running successfully.

```
62
63 // Start the server
64 app.listen(port, () => {
65   console.log(`🚀 Server running at http://localhost:\${port}`);
66 });
```

Testing and Validation

A focused testing process ensured the application's reliability and usability:

Unit Testing: Verified backend functions and API endpoints for correct data handling.

Integration Testing: Checked seamless data flow between frontend, backend, and database.

Functional Testing: Confirmed all user features—calendar selection, chart updates, and suggestions—work as intended.

Database Testing: Ensured data integrity and accurate queries.

UI & Performance Testing: Assessed layout, responsiveness, and speed across browsers.

Security Testing: Validated input handling to prevent vulnerabilities.

User Acceptance: Incorporated user feedback to refine usability.

Outcome:

All major features passed testing, the interface is intuitive, and analytics are accurate and actionable.

CONCLUSION AND FUTURE WORK

Conclusion

This project successfully developed a user-friendly web application for data analytics and visualization of operational KPIs. By integrating a calendar-based interface, interactive charts, and automated suggestions, the system enables users to efficiently explore and interpret daily performance data. The backend, built with Node.js and Express.js, ensures secure and reliable data retrieval from a well-structured MySQL database, while the frontend provides clear and accessible insights through dynamic visualizations.

The application meets its core objectives of simplifying data analysis, supporting operational decision-making, and making complex datasets accessible to a wider range of users. The systematic approach to data cleaning and KPI selection further enhances the quality and relevance of the analytics provided.

Future Work

While the current system provides robust descriptive analytics and visualization, several enhancements can be considered for future development:

Advanced Analytics:

Integrate predictive analytics or machine learning models to forecast trends and detect anomalies automatically.

Real-Time Data Integration:

Enable real-time data streaming and visualization for up-to-the-minute monitoring and faster response to operational changes.

Enhanced User Features:

Add customizable dashboards, export options (PDF/Excel), and user authentication for personalized experiences and improved data security.

Mobile Optimization:

Develop a mobile-responsive or dedicated mobile app version to increase accessibility for users on different devices.

Scalability and Extensibility:

Expand the system to handle larger datasets, additional KPIs, or integration with other data sources and reporting tools.

By pursuing these directions, the application can evolve into a comprehensive decision-support platform, further empowering users with actionable insights and advanced analytics capabilities.

EXPANSION POTENTIAL AND PRACTICAL IMPACT

Scalability and Future Analysis

While the current version of the web application is designed to analyze and visualize data for a single month, its architecture and workflow are inherently scalable. The system can be readily expanded to handle data for the entire 12 months of a year. With a full year's dataset, users will be able to:

- Conduct comprehensive annual analyses, identifying long-term trends, seasonal patterns, and operational anomalies.
- Compare performance across different months, quarters, or seasons for deeper insights.
- Generate summary reports and visualizations that reflect the plant's performance over extended periods.
- Looking further ahead, the application can be adapted to manage and analyze multi-year datasets. With sufficient historical data, advanced features such as trend forecasting and predictive analytics can be introduced, enabling:
 - Year-over-year performance comparisons.
 - Long-term forecasting for up to 10 years, supporting strategic planning and resource allocation.
 - Early detection of emerging issues or opportunities based on historical patterns.

Value for Plant Employees

This web application addresses a significant gap in the current workflow of plant employees. While daily data entry and analysis are often performed in Excel, there is typically a lack of dedicated tools for intuitive data visualization and trend analysis. The application offers:

- A centralized platform for visualizing operational KPIs, making it easier to interpret complex datasets at a glance.
- Automated trend detection and suggestive analytics, reducing manual effort and the risk of oversight.
- Enhanced decision support, allowing employees to quickly grasp what's happening in the plant and respond proactively.
- Time savings and improved accuracy compared to manual charting and analysis in spreadsheets.
- By providing these capabilities, the web app empowers plant staff to move beyond routine data entry and gain actionable insights, ultimately supporting better operational management and continuous improvement.

Summary

The foundation laid with one month's data demonstrates the system's effectiveness and sets the stage for broader adoption. As the dataset grows, the application's value will increase, offering plant employees a powerful tool for both day-to-day monitoring and long-term strategic analysis.

REFERENCES

- 1)The raw dataset for this project was provided by my instructor in the form of a folder containing daily performance reports (DPRs) for one month.
- 2)Core Technology Documentation
Node.js Documentation (Node.js Foundation):
<https://nodejs.org/docs/latest/api/>
- 3)Express.js Documentation (Express.js Foundation):
<https://expressjs.com>
- 4)MySQL 8.0 Reference Manual (Oracle Corporation):
https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/
- 5)Chart.js Documentation (Chart.js Community):
<https://www.chartjs.org/docs/>
- 6)General Web Development and Data Analytics
W3Schools Tutorials: Node.js, MySQL, Chart.js:
<https://www.w3schools.com/nodejs/>
https://www.w3schools.com/nodejs/nodejs_mysql.asp
https://www.w3schools.com/ai/ai_chartjs.asp
- 7)Project Report On Data Analytics (Scribd):
<https://www.scribd.com/document/816365441/Project-Report>
- 8)How to Write a Data Science Project Report? (ProjectPro):
<https://www.projectpro.io/article/how-to-write-a-data-science-project-report/437>
- 9)Exploratory Data Analysis Web Application (IJRASET):
<https://www.ijraset.com/research-paper/exploratory-data-analysis-web-application>