# Assignment: Intelligent Power Demand Forecasting

**To the Candidate:**

Welcome to the hiring process at Exascale Deeptech & AI Pvt. Ltd.!

This assignment is a high-intensity, two-week sprint designed for the **Data Developer Intern** role. It will test your combined skills in ***machine learning*** and ***web development*** by asking you to build a complete, end-to-end prototype that addresses specific, real-world technical challenges.

**The Deadline is Final:** All submissions are due by **Friday, July 18, 2025, at 5:00 PM IST**. Late submissions will not be considered.

Please read the entire brief, paying close attention to the **Evaluation Criteria** and the **Technical & Data Considerations** section, before you begin.

## **Project Brief: Proof-of-Concept for Apex Power & Utilities (APU)**

The core objective is to predict electricity demand for every 10-minute block of the day (144 blocks total) for APU, a major power provider. The solution must be a scalable, container-deployable web application with a robust forecasting model at its heart.

## **Technical & Data Considerations**

To ensure your solution is not generic, you must incorporate the following real-world complexities into your model. We will provide you with mock datasets for some of these challenges.

1. **Historical Load Data**: You will be provided with a ***Utility\_consumption.csv*** file containing timestamped load data for several feeders. This data contains gaps, errors, and outliers that you must handle.
2. **Weather Data**: You must source weather data (temperature, humidity, cloud cover and wind speed) from a public API for the location **Dhanbad, Jharkhand, India**. Your model's performance will depend on how effectively you integrate this external data.
3. **Localized Holidays**: Power demand is heavily influenced by local events. You are responsible for sourcing a list of specific festive and industrial holidays relevant to the **Dhanbad, Jharkhand** region. A generic national holiday calendar will be insufficient, and failure to account for local holidays will negatively impact model accuracy.

## **Evaluation Criteria & Scoring (Total: 100 Points)**

Your submission will be graded on your ability to handle the specific challenges outlined above.

**Goal**: Build and deploy a functional, end-to-end forecasting prototype that addresses specific data complexities.

### **Milestone 1: Exploratory Data Analysis (EDA) & Data Cleaning (25 Points)**

* Conducts a thorough statistical and visual exploration of all datasets (load, frequency, weather, holidays) to uncover insights. **(15 points)**
* Effectively cleans the load data and handles outliers/gaps, justifying the methods based on your EDA findings. **(10 points)**

### **Milestone 2: Feature Engineering & Model Architecture Justification (35 Points)**

* Successfully sources and integrates weather data and the self-sourced localized holiday data. **(10 points)**
* Demonstrates thoughtful feature engineering, creating valuable predictors from all available data sources. **(10 points)**
* Provides a clear, data-driven justification in your notebook for the chosen modeling approach and architecture, linking it directly to your EDA. **(15 points)**

### **Milestone 3: Model Implementation & Backend API (20 Points)**

* Implements and validates the proposed model using the engineered features. **(10 points)**
* Builds working API endpoints that:
  + Load the trained model and serve a fresh 24-hour forecast (96 blocks). **(5 points)**
  + Provide weather data (temperature, humidity, cloud cover) and localized holiday data for the forecast period to support frontend visualizations. **(5 points)**

### **Milestone 4: Frontend Visualization & Deployment (20 Points)**

* Implements a simple dashboard with a working chart to display the API's forecast data, alongside appropriate visualizations or UI elements for weather data (temperature, humidity, cloud cover) and localized holiday data (e.g., markers, annotations, or tables). **(10 points)**
* Provides a working Dockerfile and a well-structured README.md explaining how to build and run the entire project. **(10 points)**

## **Your Assignment: Data Developer Intern**

Your task is to build a full, albeit simplified, version of the demand forecasting system, incorporating the specific data considerations.

### **Analysis & Modeling**

1. Build and document the core forecasting model.
2. Perform a thorough EDA on all datasets (*Utility\_consumption*.csv, weather data, and holiday data).
3. Clean the provided *Utility\_consumption*.csv file, addressing gaps, errors, and outliers.
4. Source and integrate weather data from a public API and your self-sourced holiday list for **Dhanbad, Jharkhand**.
5. Engineer features, select a model architecture, train the model, and save the trained artifact (e.g., a .pkl or .h5 file).
6. Document this entire process, especially your analysis and justifications, in a **Jupyter Notebook**.

### **Backend**

1. Create an API (e.g., using FastAPI, Flask, or Node.js/Express) that loads your saved model artifact.
2. The API must include endpoints that:
   * Generate and return a forecast for the next 24 hours (96 blocks).
   * Provide weather data (temperature, humidity, cloud cover) and localized holiday data for the forecast period to support frontend visualizations.

### **Frontend**

1. Build a minimal, **single-page web application** that calls your API and visualizes the returned forecast data on an interactive chart (e.g., using Chart.js, D3.js, or similar). The dashboard must also include appropriate visualizations or UI elements (e.g., charts, tables, or annotations) to display weather data (temperature, humidity, cloud cover) and localized holiday data for **Dhanbad, Jharkhand**.

### **Package**

1. Containerize your application using Docker.
2. Provide clear instructions in a README.md on how to build and run the entire project.

## **Submission**

Submit a link to a single Git repository containing:

* Your **Jupyter Notebook** documenting the EDA, data cleaning, feature engineering, and model justification.
* **Backend code** for the API.
* **Frontend code** for the visualization dashboard.
* **Dockerfile** for containerization.
* A comprehensive **README.md** explaining how to build and run the project.
* The provided mock data files (*Utility\_consumption*.csv ) for reproducibility.

**Good luck!** We are excited to see your end-to-end solution.