# ENERGY LOAD PLATFORM

Technical Specification & API Reference

MVP v1.0 — For Client Review

# 1. Project Overview

This document outlines the technical architecture, API design, and feature scope for the Energy Load Platform MVP. It is intended for client review to confirm alignment before development proceeds.

The platform ingests raw energy consumption data in any format, normalizes it to a standard time grid, enriches it with weather and calendar context, validates it through a statistical quality pipeline, and produces an 8,760-hour annual load forecast vector with confidence intervals — the industry-standard output for energy pricing and procurement planning.

## 1.1 Core Value Proposition

- Upload any energy file format → receive a clean, forecast-ready annual load profile
- No manual data wrangling — AI handles format detection, unit correction, and time zone alignment
- Statistical quality assurance with automatic gap filling based on recognized usage patterns
- Output is the 8,760-hour vector (annual hours) with confidence intervals, ready for price calculation

## 1.2 MVP Scope Summary

| Area | In MVP | Post-MVP |
|---|---|---|
| File Formats | CSV, Excel, MSCONS | EDI, XML, custom formats |
| Time Normalization | 15-min UTC grid, DST handling | Configurable grid intervals |
| Quality Checks | Core stages (gaps, peaks, DST) | All 9 statistical stages |
| Weather Enrichment | API integration (read) | Historical backfill, multi-region |
| Calendar Enrichment | Public holidays | Custom vacation/shutdown calendars |
| Forecast Output | 8,760 vector + confidence intervals | Multi-scenario forecasts |
| Frontend | Drag & Drop upload + results view | Dashboard, multi-user, roles |
| Auth | Single user / API key | SSO, team management |

# 2. System Architecture

The platform is built as a containerized Python service with a lightweight frontend. All components communicate internally within a single deployable unit for the MVP, with clear module boundaries to allow future extraction into microservices.

## 2.1 Technology Stack

| Layer | Technology | Purpose |
|---|---|---|
| Runtime | Python 3.11+ (Docker container) | Core processing engine |
| Forecasting | Python Prophet (Meta) | Time series forecasting, seasonality |
| Numerical | NumPy / Pandas | Vector calculation, data processing |
| Time Series DB | PostgreSQL + TimescaleDB | Storage and querying of time series |
| Object Storage | Google Cloud Storage (GCS) | Raw file storage, output artifacts |
| API Framework | FastAPI | REST API layer, async, auto-docs |
| Frontend | React (minimal) | Drag & drop upload, results display |
| Containerization | Docker / Docker Compose | Deployment, environment consistency |
| External APIs | Open-Meteo (weather), Nager.Date (holidays) | Context enrichment |

## 2.2 Data Flow

The end-to-end data flow through the system follows these stages in sequence:

- User uploads file via drag-and-drop frontend
- File stored in GCS raw bucket, upload job created in DB
- Parser service detects format, extracts time series data
- Normalizer aligns data to 15-minute UTC grid, corrects units
- Enrichment service fetches weather and calendar data for the time range
- Quality check pipeline runs statistical tests and imputes missing values
- Forecast engine (Prophet) generates 8,760-hour annual vector
- Results stored in TimescaleDB and GCS output bucket
- Frontend polls job status and displays downloadable results

# 3. API Endpoints

All endpoints are prefixed with /api/v1. The API is documented automatically via FastAPI's built-in OpenAPI interface, accessible at /docs when the server is running. Authentication uses API key headers for the MVP.

## 3.1 Authentication

| Header | Value | Required |
|---|---|---|
| X-API-Key | Your API key string | Yes — all endpoints |
| Content-Type | multipart/form-data (uploads) or application/json | Yes |
| Accept | application/json | Recommended |

## 3.2 File Ingestion

### POST /api/v1/upload

Accepts a raw energy data file. Detects format automatically. Returns a job ID for async tracking.

| Field | Type | Description |
|---|---|---|
| file | multipart/form-data | The energy file (CSV, Excel, MSCONS) |
| timezone | string (optional) | Source timezone hint e.g. Europe/Berlin. Auto-detected if omitted. |
| label | string (optional) | Human-readable name for this dataset |

**Response (202 Accepted):**

```
{ "job_id": "uuid-v4", "status": "queued", "created_at": "ISO8601" }
```

### GET /api/v1/upload/{job_id}/status

Polls the processing status of an upload job.

| Status Value | Meaning |
|---|---|
| queued | Job received, waiting to process |
| parsing | File format detection and extraction in progress |
| normalizing | Time alignment and unit correction running |
| enriching | Weather and calendar data being fetched |
| quality_check | Statistical validation and imputation running |

| Status Value | Meaning |
|---|---|
| forecasting | Prophet model generating 8,760 vector |
| complete | All stages done, results available |
| failed | Processing error — see error field in response |

**Response (200 OK):**

```
{ "job_id": "...", "status": "forecasting", "progress_pct": 82, "stage":
"forecasting", "error": null }
```

## 3.3 Parsing & Normalization

### GET /api/v1/jobs/{job_id}/parsed

Returns the raw parsed time series after format detection and extraction, before normalization. Useful for verifying the parser understood the source file correctly.

| Response Field | Type | Description |
|---|---|---|
| detected_format | string | e.g. CSV, MSCONS, XLSX |
| detected_timezone | string | Source timezone detected or provided |
| detected_unit | string | kW or kWh as found in source |
| row_count | integer | Total data rows extracted |
| time_range | object | start and end timestamps in source timezone |
| sample | array | First 10 rows of extracted data for spot-check |

### GET /api/v1/jobs/{job_id}/normalized

Returns the normalized time series on the 15-minute UTC grid with units corrected to kW.

| Response Field | Type | Description |
|---|---|---|
| interval_minutes | integer | Always 15 for MVP |
| unit | string | Always kW after normalization |
| timezone | string | Always UTC after normalization |
| total_intervals | integer | Number of 15-min slots in the series |
| gap_count | integer | Gaps detected before imputation |
| dst_events | integer | DST transitions handled |
| data | array | Full normalized time series [{ts, value}] |

## 3.4 Quality Check

### GET /api/v1/jobs/{job_id}/quality-report

Returns the full statistical quality report. Each stage reports pass/fail status, findings, and actions taken.

| QC Stage | What It Checks | Action if Failed |
|---|---|---|
| Gap Detection | Missing intervals in the time series | Imputation based on user profile |
| Peak/Outlier Detection | Statistically anomalous load values | Flag and optionally replace |
| DST Conformity | Correct handling of clock changes | Re-align affected intervals |
| Unit Consistency | Consistent kW vs kWh throughout file | Correct and log change |
| Temporal Completeness | Full year coverage check | Warn, use available data |
| Negative Values | Values below zero (metering errors) | Replace with imputed value |
| Flatline Detection | Extended identical values (sensor fault) | Flag for review |
| Resolution Consistency | Mixed interval resolutions | Resample to 15-min grid |
| Profile Plausibility | Load shape matches recognized profile | Warn if no match found |

Response includes a stages array with per-stage results, an overall_pass boolean, and an imputation_count showing how many intervals were auto-filled.

## 3.5 Forecast Output

### GET /api/v1/jobs/{job_id}/forecast

Returns the 8,760-hour annual forecast vector. This is the primary output of the platform.

| Response Field | Type | Description |
|---|---|---|
| vector_length | integer | Always 8,760 (hours in a year) |
| base_year | integer | The year the forecast represents |
| confidence_level | float | e.g. 0.95 for 95% confidence interval |
| peak_load_kw | float | Maximum predicted load value |
| base_load_kw | float | Minimum predicted load value |
| annual_consumption_kwh | float | Total integrated consumption |
| data | array | 8,760 entries: [{hour, yhat, yhat_lower, yhat_upper}] |

### GET /api/v1/jobs/{job_id}/forecast/download

Returns the forecast as a downloadable file.

| Query Param | Options | Default |
|---|---|---|
| format | csv, xlsx, json | csv |
| include_confidence | true, false | true |
| timezone | Any IANA timezone string | UTC |

## 3.6 Enrichment Data

### GET /api/v1/jobs/{job_id}/enrichment

Returns the weather and calendar data that was attached to this dataset for context.

| Field | Source | Description |
|---|---|---|
| weather.provider | Open-Meteo API | Name of weather data provider |
| weather.variables | Open-Meteo API | e.g. temperature_2m, solar_radiation |
| weather.coverage_pct | Open-Meteo API | Percentage of time range covered |
| calendar.country_code | Nager.Date API | e.g. DE, GB, FR |
| calendar.holidays | Nager.Date API | List of public holidays in the time range |
| calendar.holiday_count | Nager.Date API | Number of holidays detected |

## 3.7 Health & System

| Endpoint | Method | Description |
|---|---|---|
| GET /health | GET | Liveness check. Returns 200 OK if service is up. |
| GET /api/v1/status | GET | Returns service version, DB connection status, storage status. |
| GET /docs | GET | Auto-generated OpenAPI interactive docs (FastAPI built-in). |
| GET /redoc | GET | Alternative API docs in ReDoc format. |

# 4. Key Data Models

The following describes the core data structures used throughout the platform.

## 4.1 Job

A Job represents a single file upload and its entire processing lifecycle.

| Field | Type | Description |
|---|---|---|
| id | UUID | Unique job identifier |
| status | enum | queued \| parsing \| normalizing \| enriching \| quality_check \| forecasting \| complete \| failed |
| file_name | string | Original uploaded filename |
| file_size_bytes | integer | File size |
| gcs_raw_path | string | Path to raw file in GCS |
| gcs_output_path | string | Path to forecast output in GCS |
| created_at | timestamp | Job creation time (UTC) |
| completed_at | timestamp | Completion time, null until done |
| error_message | string | Error detail if status is failed |

## 4.2 TimeSeries Record (TimescaleDB)

Each normalized data point is stored as a hypertable row, partitioned by time for efficient querying.

| Column | Type | Description |
|---|---|---|
| ts | timestamptz | Timestamp (UTC, PRIMARY KEY with job_id) |
| job_id | UUID | Foreign key to job |
| value_kw | float8 | Load value in kW |
| is_imputed | boolean | True if value was auto-filled |
| imputation_method | string | e.g. profile_mean, linear_interpolation |

## 4.3 Forecast Record (TimescaleDB)

| Column | Type | Description |
|---|---|---|
| hour_ts | timestamptz | Hourly timestamp for the forecast (UTC) |
| job_id | UUID | Foreign key to job |
| yhat | float8 | Predicted load in kW |
| yhat_lower | float8 | Lower confidence bound |

| Column | Type | Description |
|---|---|---|
| yhat_upper | float8 | Upper confidence bound |

# 5. External API Dependencies

## 5.1 Weather Data — Open-Meteo

| Attribute | Detail |
|---|---|
| Provider | Open-Meteo (open-meteo.com) |
| Cost | Free tier available; no API key required for basic use |
| Key Variables | temperature_2m, solar_radiation, wind_speed_10m, precipitation |
| Resolution | Hourly historical data |
| Coverage | Global, 1940 to present |
| Usage | Fetched per job using location coordinates from the dataset's region |

## 5.2 Holiday/Calendar — Nager.Date

| Attribute | Detail |
|---|---|
| Provider | Nager.Date (date.nager.at) |
| Cost | Free, open source |
| Coverage | 110+ countries, public holidays |
| Usage | Fetched once per country/year combination, cached in DB |
| Extension | Custom shutdown/vacation dates can be added post-MVP |

# 6. Frontend MVP

The MVP frontend is intentionally minimal. Its only job is to let a user upload a file and view/download the results. No complex UI, no dashboards, no multi-user management.

## 6.1 User Flow

- Land on page → see drag-and-drop upload zone
- Drop or select file (CSV, Excel, MSCONS)
- Optional: provide timezone hint and dataset label
- Submit → see processing status with stage indicator
- On completion → see summary card (peak load, annual consumption, quality score)
- Download forecast as CSV or Excel

## 6.2 Frontend Pages / Views

| View | Description |
|---|---|
| Upload | Drag & drop zone, optional fields, submit button |
| Processing | Job status with live polling, stage progress indicator |
| Results | Summary metrics, quality report overview, download buttons |
| Error | Clear error message with option to re-upload |

# 7. Open Questions for Client Review

The following items require client input before or during development. Please review and confirm or provide guidance on each.

| # | Question | Why It Matters |
|---|---|---|
| 1 | Which primary markets/countries should be supported at launch? (e.g. DE, GB, FR) | Determines default timezone, holiday calendar, and MSCONS variant |
| 2 | Is a user account/login required in the MVP, or is API-key-only access sufficient? | Affects auth implementation scope significantly |
| 3 | Should the 8,760 forecast vector be based on a specific future year, or always the next calendar year? | Affects Prophet model configuration |
| 4 | Is the weather enrichment mandatory for processing, or optional enhancement? | Affects what happens if weather API is unavailable |
| 5 | What is the expected maximum file size for uploads? (impacts GCS config and timeout settings) | Infrastructure sizing |
| 6 | Should imputed values be visible/flagged in the download output? | Affects output schema |
| 7 | Is there an existing product name or brand for the repo and API naming? | Affects naming of repo, endpoints, docs |
| 8 | Are there specific confidence interval widths required? (default: 95%) | Prophet CI configuration |

---

*End of Document — Prepared for Client Review*

*Please confirm, amend, or raise questions against any section above.*