I'll analyze this FastAPI-based ISA-18.2 Alarm Rationalization system and explain how it works step by step with examples.

## \*\*System Overview\*\*

This is an \*\*alarm management and analysis system\*\* that follows \*\*ISA-18.2 standards\*\* (International Standard for alarm management in process industries). It helps analyze alarm data from industrial systems to improve alarm performance.

## \*\*How It Works - Step by Step\*\*

### \*\*Step 1: Data Input\*\*

The system accepts alarm data in two ways:

- \*\*Upload files\*\*: CSV/Excel files containing alarm data

- \*\*Auto-discovery\*\*: Automatically finds alarm files in project folders

\*\*Example alarm data structure:\*\*

```csv

Event Time, Location Tag, Priority, Description

2023-01-01 10:30:00, TI-101, P1, High Temperature Alarm

2023-01-01 10:31:00, PI-201, P2, High Pressure Alarm

```

### \*\*Step 2: Data Processing\*\*

```python

def \_normalize\_columns\_to\_canonical(df):

# Converts various column names to standard format

# "Event Time", "eventtime", "timestamp" → "Event Time"

# "Location Tag", "tag", "tagname" → "Location Tag"

```

### \*\*Step 3: Column Detection\*\*

The system automatically detects important columns:

```python

def detect\_timestamp\_column(df): # Finds time/date columns

def detect\_tag\_column(df): # Finds alarm tag/location columns

def detect\_priority\_column(df): # Finds priority/severity columns

```

### \*\*Step 4: ISA-18.2 Metrics Calculation\*\*

```python

def calculate\_isa\_metrics(df, ts\_col, tag\_col, priority\_col):

# Calculates key performance indicators:

# - Alarms per hour (target: ≤6, max: ≤12)

# - Flood periods (target: <1% of time)

# - Top 10 contributors (target: <5% of total alarms)

```

## \*\*API Endpoints and Their Uses\*\*

### \*\*🏠 Basic Operations\*\*

| Endpoint | Method | Purpose | Example Use |

|----------|--------|---------|-------------|

| `/` | GET | Health check | `GET /` → Returns system info |

| `/health` | GET | System status | `GET /health` → Returns benchmarks |

### \*\*📁 File Management\*\*

| Endpoint | Method | Purpose | Example Use |

|----------|--------|---------|-------------|

| `/upload` | POST | Upload alarm file | Upload CSV/Excel file |

| `/files` | GET | List uploaded files | See all uploaded files |

| `/auto-files` | GET | Auto-discovered files | Files found in project folders |

| `/auto-files/refresh` | POST | Refresh file scan | Rescan project directories |

| `/all-files` | GET | All available files | Combined view of all files |

\*\*Example Upload:\*\*

```bash

curl -X POST "http://localhost:8000/upload" \

-F "file=@alarm\_data.csv"

```

### \*\*🔍 Analysis Operations\*\*

| Endpoint | Method | Purpose | Example Use |

|----------|--------|---------|-------------|

| `/analyze/{file\_id}` | POST | Analyze single file | Get ISA-18.2 metrics |

| `/analyze/batch` | POST | Analyze all files | Bulk analysis |

| `/file/{file\_id}/preview` | GET | Preview file data | See first 10 rows |

| `/file/{file\_id}/statistics` | GET | Detailed statistics | Data quality info |

\*\*Example Analysis Response:\*\*

```json

{

"total\_alarms": 1500,

"time\_span\_hours": 168,

"avg\_alarms\_per\_hour": 8.9,

"isa\_performance": "Acceptable",

"flood\_periods": 12,

"flood\_percentage": 2.1,

"flood\_performance": "Poor"

}

```

### \*\*📊 Visualization Charts\*\*

| Endpoint | Method | Purpose | Example Use |

|----------|--------|---------|-------------|

| `/charts/alarm-rate/{file\_id}` | GET | Hourly alarm rate chart | Track alarm trends |

| `/charts/top-contributors/{file\_id}` | GET | Top alarm sources | Identify problematic tags |

| `/charts/priority-distribution/{file\_id}` | GET | Priority breakdown | P1, P2, P3 distribution |

| `/charts/pareto-analysis/{file\_id}` | GET | Pareto chart | 80/20 analysis |

| `/charts/flood-analysis/{file\_id}` | GET | Flood periods chart | When alarm bursts occur |

\*\*Example Chart Request:\*\*

```bash

GET /charts/alarm-rate/file\_123

```

Returns Plotly.js chart showing alarms per hour over time.

### \*\*📈 Advanced Analytics\*\*

| Endpoint | Method | Purpose | Example Use |

|----------|--------|---------|-------------|

| `/advanced/chattering/{file\_id}` | GET | Detect repeating alarms | Find oscillating sensors |

| `/advanced/standing-alarms/{file\_id}` | GET | Long-duration alarms | Find stuck alarms |

| `/insights/{file\_id}` | GET | AI-generated insights | Actionable recommendations |

\*\*Example Chattering Analysis:\*\*

```bash

GET /advanced/chattering/file\_123?window\_minutes=10&threshold=5

```

Finds alarms that repeat 5+ times within 10-minute windows.

### \*\*🎯 Dashboard & Reports\*\*

| Endpoint | Method | Purpose | Example Use |

|----------|--------|---------|-------------|

| `/dashboard/summary` | GET | Overall KPIs | Management dashboard |

| `/export/metrics/{file\_id}` | GET | Export analysis | JSON or CSV download |

| `/benchmarks` | GET | ISA-18.2 targets | Standard benchmark values |

## \*\*Real-World Usage Example\*\*

### \*\*Scenario: Analyzing Refinery Alarm Data\*\*

1. \*\*Upload Data:\*\*

```python

# Upload alarm export from DCS system

POST /upload (file: refinery\_alarms\_jan2024.csv)

# Returns: {"file\_id": "refinery\_123", "rows": 15000, "columns": 8}

```

2. \*\*Analyze Performance:\*\*

```python

POST /analyze/refinery\_123

# Returns ISA-18.2 metrics showing 15 alarms/hour (POOR performance)

```

3. \*\*Investigate Root Causes:\*\*

```python

GET /charts/top-contributors/refinery\_123

# Shows TI-401 (temperature) generates 12% of all alarms

GET /advanced/chattering/refinery\_123

# Reveals TI-401 chatters 8 times per hour

```

4. \*\*Get Recommendations:\*\*

```python

GET /insights/refinery\_123

# Returns: "❌ Alarm rate (15/hr) exceeds ISA maximum - focus on TI-401 deadband tuning"

```

## \*\*Key ISA-18.2 Benchmarks\*\*

```python

ISA\_BENCHMARKS = {

'target\_alarms\_per\_hour': 6, # Good performance

'max\_alarms\_per\_hour': 12, # Maximum acceptable

'flood\_threshold': 10, # >10 alarms in 10 minutes = flood

'flood\_target\_percent': 1.0, # <1% of time should have floods

'top\_10\_contribution\_target': 5.0, # Top 10 tags should be <5% of total

}

```

## \*\*System Benefits\*\*

1. \*\*Automatic Analysis\*\* - No manual data processing needed

2. \*\*ISA-18.2 Compliance\*\* - Industry standard benchmarks

3. \*\*Visual Insights\*\* - Interactive charts and dashboards

4. \*\*Actionable Reports\*\* - Specific improvement recommendations

5. \*\*Batch Processing\*\* - Analyze multiple alarm sources simultaneously

This system helps industrial facilities improve their alarm management by identifying problematic alarms, reducing operator workload, and ensuring compliance with international standards.