Root Cause Analysis (RCA) and Alarm Clustering – High Level Architecture

1. Overview

This system simulates telecom alarm monitoring and analysis.
It generates synthetic logs, clusters alarms using **unsupervised learning (DBSCAN)**, and performs **Root Cause Analysis (RCA)** by identifying the primary source of issues in each cluster.

The architecture supports:

- **Synthetic log generation** for testing RCA logic.
- **Clustering alarms** based on temporal and categorical similarity.
- **Root Cause Analysis** based on frequency and severity of alarms.

2. System Workflow

Step-by-step pipeline:

1. **Log Generation** – Create synthetic alarm logs with fields: `timestamp, component, type, severity`.
2. **Log Storage** – Save logs to a CSV file for persistence.
3. **Log Loading** – Reload and convert logs to structured Python dictionaries.
4. **Preprocessing** – Transform logs into a numerical format (timestamp normalization, encoding of categorical features).
5. **Clustering** – Apply **DBSCAN** to group similar alarms into clusters.

6. **Root Cause Analysis (RCA)** – Analyze clusters to determine:
    - Dominant component
    - Dominant alarm type
    - Average severity

---

3. Core Components

3.1 Log Generation

- **Function:** `generate_synthetic_logs(num_logs)`
- **Purpose:** Create realistic but random telecom alarms.
- **Fields:**
    - `timestamp` → when the alarm occurred
    - `component` → network element (Router, RAN, Switch, etc.)
    - `type` → nature of issue (Link Failure, Packet Loss, etc.)
    - `severity` → integer 1–5 (low → critical)

3.2 Preprocessing

- **Function:** `preprocess_logs(logs)`
- **Purpose:** Convert logs into **numerical feature vectors** for clustering.
- **Steps:**
    1. **Timestamp normalization:** Convert datetime into seconds relative to earliest log.

    2. **Component encoding:** Map each unique component to an integer.

    3. **Type encoding:** Map each unique alarm type to an integer.

    4. **Severity:** Keep as numeric (1–5).

- **Final Feature Vector per log:**

    [normalized_timestamp, component_id, type_id, severity]

- **Output:** `numpy.ndarray` (matrix of logs × features)

3.3 Clustering

- **Function:** `cluster_alarms(data)`

- **Algorithm: DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**

- **Parameters:**

    - `eps = 1000` → max distance between two points to be in same neighborhood

    - `min_samples = 3` → minimum number of alarms to form a valid cluster

- **Why DBSCAN?**

    - Works with arbitrary shaped clusters.

    - Identifies outliers (noise points).

    - Suitable for time-series + categorical embeddings.

- **Output:** `list` of cluster IDs (same length as logs).

3.4 Root Cause Analysis (RCA)

- **Function:** `root_cause_analysis(logs, clusters)`

- **Purpose:** Identify the **primary issue** in each cluster.

- **Process:**
    1. Group logs by cluster ID.
    2. Count frequency of:
        - **Components** (e.g., Router most frequent in this cluster).
        - **Alarm Types** (e.g., Link Failure dominates).
    3. Compute average severity across alarms.
    4. Ignore noise points (`cluster_id = -1`).

RCA Output Example:

```
{
  0: {
    "Primary Component": "Router",
    "Primary Alarm Type": "Link Failure",
    "Average Severity": 4.2
  },
  1: {
    "Primary Component": "Switch",
    "Primary Alarm Type": "Packet Loss",
    "Average Severity": 2.8
  }
```

- }

4. Data Flow & Dependencies

```
+---------------------------+
|       Load Logs           |
| (CSV → list of dicts)  |
+---------------------------+
            |
            v
+------------------------------+
|   Preprocess Logs (opt)   |
| (np.array transformation) |
+---------------------------------+
          |
          v
+-----------------------------+
|     Cluster Alarms        |
| (DBSCAN: eps, min_samples)|
| → cluster labels          |
+-----------------------------+
          |
          v
+-------------------------------+
|  Root Cause Analysis (RCA)|
|  - Group by cluster_id    |
|  - Count components       |
|  - Count alarm types      |
|  - Avg severity           |
+--------------------------------+
          |
          v
+-------------------------------+
|    RCA Results (dict)     |
| {cluster_id: summary}   |
+-------------------------------+
```

5. Design Considerations

- **Clustering Basis:**
    - Time proximity (alarms close in time are likely related).
    - Component similarity (same network element).
    - Alarm type similarity.
    - Severity included as a feature to bias clustering.
- **RCA Basis:**
    - Frequency analysis → dominant component/type assumed as root cause.

    - Severity averaging → gives cluster risk level.

- **Scalability:**
    - DBSCAN may be replaced by hierarchical clustering for large datasets.
    - Logs can be streamed from real systems instead of synthetic generation.

---

6. Future Enhancements

- Weight features differently (e.g., severity > timestamp).
- Introduce **temporal correlation** beyond simple normalization.
- Integrate with **telecom Network Management Systems (NMS/OSS)**.
- Visualize clusters with dashboards.