Software Engineer | Junior | Python

Homework task

You are tasked with developing an internal tool to identify and categorize expiring contracts for renewal. The goal is to process contract data, apply a set of prioritized notification rules, and generate a structured report of contracts that require attention, while preventing duplicate notifications.

Task

Your task is to write a script that processes contract data from a JSON file. The script must identify which contracts are due for renewal based on a configurable set of rules, categorize them by a single, highest-priority reason, and produce a JSON output of the notifications.

The script must manage state to ensure that a contract is not re-notified for the same or a lower-priority reason.

While we would prefer to see **Python** or **JavaScript/TypeScript**, you may write your solution in any language you feel comfortable with.

Inputs

1. `contracts.json`: A file containing a list of all contracts. Each contract is a JSON object with the following structure:

```JSON
{
  "id": 1,
  "software_name": "Atlassian",
  "owner": "John Smith",
  "organization": "Nord Security",
  "annual_cost_eur": 15000.00,
  "renewal_date": "2025-08-01"
}
```

2. `notification_log.json`: A file that records which contracts have already been notified. This file may be empty or non-existent on the first run. Its structure will be a dictionary mapping contract IDs to the last notification details:

```JSON
{
  "1": { "notified_on": "2025-07-25", "reason": "High-Cost" }
}
```

3. `config.json`: A configuration file that defines the notification rules.

```JSON
{
  "rules": [
    {"reason": "Urgent", "days_to_expiry": 3},
    {"reason": "High-Cost", "days_to_expiry": 30, "min_annual_cost": 10000},
    {"reason": "Upcoming", "days_to_expiry": 14}
  ],
  "priority": ["Urgent", "High-Cost", "Upcoming"]
}
```

Core Logic & Requirements

1. **Rule Prioritization**: A contract might meet multiple notification criteria. It must be assigned only to the **single, highest-priority category**. The order of priority is defined in the `config.json` file.
    - *Example*: A €12,000 contract expiring in 2 days is both "Urgent" and "High-Cost". Because "Urgent" is the highest priority, it should be categorized as "Urgent" only.
2. **Stateful Notification Filtering**: Once a notification is generated for a contract, it should not be notified again unless it escalates to a *higher-priority* category.
    - If a contract was previously notified as "Upcoming" and now qualifies as "Urgent," a new notification **should** be generated.
    - If a contract was previously notified as "Urgent," it **should not** be notified again.
3. **Date Calculations**: All date calculations should be based on a "current date," which for testing purposes should be an adjustable variable within your script.

Outputs

The script should print a single JSON array to standard output containing the notification objects for the current run. Each object must include the contract details along with the reason for the notification. After printing the output, the script must update the `notification_log.json` file with the new notifications.

Example output

```JSON
[
  {
    "software_name": "Atlassian",
    "owner": "John Smith",
    "organization": "Nord Security",
    "annual_cost_eur": 15000.00,
    "renewal_date": "2025-08-01",
    "reason": "Urgent"
  }
]
```

What we are looking for

- **Correct functionality:** The logic for prioritization and stateful filtering must be correctly implemented.
- **Code quality:** Clean, readable, and maintainable code.
- **Project structure:** The project should be logically organized.
- **Unit testing:** Comprehensive tests that cover the core logic.
- **Documentation:** Clear `README.md` explaining how to set up and run the script and tests.

Once you have finished, please upload the project to a GitHub repository and share the link with us.

Good luck!