**Test Case #1:**

**Feature: Conditional Field Validation for Programme Type**

Scenario: Validate conditional presence of “editorial\_subtitle” based on “programme\_type”

Given the valid GET request is sent to the endpoint URL

When the API returns a response containing event data

Then the editorial\_subtitle field should:

Be present if programme\_type is "sequential"

Be absent if programme\_type is "self-contained"

**Expected Output:**

* The “programme\_type” field should always be present in the dataset.
* The “programme\_type” field should only contain pre-defined valid values such as:
* "sequential"
* "self-contained"
* If the “programme\_type” field contains an invalid value, the response should include an error message indicating the invalid type.

**When “programme\_type” is "sequential":**

* The “editorial\_subtitle” field must be present.
* The “editorial\_subtitle” value should be a valid, non-empty string (e.g., "All episodes available now").
* If “editorial\_subtitle” is missing or contains an empty value, the response should indicate this as an error.

**When programme\_type is "self-contained":**

* The “editorial\_subtitle” field must not be present.
* If “editorial\_subtitle” is present, it should trigger a validation error.

**Test Case #2:**

**Feature: API Validation for Broadcast Events**

Scenario: Validate Event Data Structure in the Response

Given the valid GET request is sent to the endpoint URL

When the API returns a response containing event data

Then the response should contain a list of "events"

And each "event" in the list should have a valid "name" field (e.g., "started", "ended")

And each "event" should contain a valid "offset" field (numeric, e.g., 30)

And each "event" should contain a valid "system" field (e.g., "uas", "optimizely", "dax")

**Expected Output:**

* The response should contain a list of events under the "events" key.
* The list should contain at least one event.
* If the list is empty, the response should return a valid error message indicating no events available.

**Name field validation:**

* Each "event" in the list should have a valid "name" field.
* The "name" field should be a non-empty string.
* The "name" field should contain only pre-defined valid values such as:

"started", "ended", "iplxp-ep-started"and "iplxp-ep-watched"

* If the "name" field contains any invalid value, the response should return an error message stating the invalid name.

**Offset field validation:**

* Each "event" should contain a valid "offset" field.
* The "offset" field should be a numeric value representing the event's time offset (e.g., 30, 1566).
* The "offset" value must be a positive integer.
* If the "offset" field contains a non-numeric value or a negative integer, the response should return an error message stating the issue.

**System field validation:**

* Each "event" should contain a valid "system" field.
* The "system" field should be a non-empty string.
* The "system" field should only contain valid pre-defined values, such as:

"uas", "optimizely" and "dax"

* If the "system" field contains any invalid value, the response should return an error message showing the invalid system.

**Test Case #3:**

**Feature: API Validation for Episode Availability**

Scenario: Validate Availability Data Structure in the Response

Given the valid GET request is sent to the endpoint URL for a specific episode

When the API returns a response containing the availability data

Then the response should contain an "availability" object

And the "availability" object should contain the following fields: "start", "end" and "remaining"

And the "start" field should contain a valid date in ISO 8601 format

And the "end" field should contain a valid date in ISO 8601 format

And the "remaining" field should contain a readable duration string such as "Available for X days" or "Available until {end\_date}"

**Expected Output:**

* The response should include an "availability" node.
* The "availability" object should contain the following required fields: "start", "end" and "remaining"

**Start Date field validation:**

* The "start" field should contain a valid date in ISO 8601 format (e.g., "2023-09-11T10:15:00Z").
* The "start" field should represent a valid timestamp in UTC.
* The "start" field should be non-empty and not null.
* The date in the "start" field must be either in the past or current for availability that has started, or a future date for upcoming availability.

**End Date field validation:**

* The "end" field should contain a valid date in ISO 8601 format (e.g., "2023-10-11T12:00:00Z").
* The "end" date must always be later than the "start" date.
* The "end" field must be non-empty and not null.
* The date in the "end" field must represent a future timestamp if the content is still available, or a past date if the content has expired.

**Remaining (Time) field validation:**

* The "remaining" field should contain a readable duration string (e.g., "Available for 29 days" or "Available until 2023-10-11T12:00:00Z").
* The "remaining" field should be non-empty and not null.
* The value should be dynamically updated based on the current date.
* If the "end" date is in the future, the "remaining" field should show the number of days remaining.
* If the "end" date has passed, the "remaining" field should reflect expired status or a similar message indicating unavailability (e.g., "Expired").

**Date range consistency:**

* Ensure that the "start" date is before the "end" date. If not, then API should return an error message stating the issue.