

## Backend Intern Assignment – Test Case Documentation

**Project:** Machine Event Ingestion and Statistics System

**Tech Stack:** Java, Spring Boot

**Testing Tool:** Postman

### Test Case 1: Duplicate Event → Deduplication

**Objective:** Check that identical events are deduplicated.

**Step:** Send two events with same eventId and same payload in one batch.

**Output:**

Accepted = 1, Deduped = 1

The screenshot shows the Postman interface with the following details:

- Request:** POST <http://localhost:8080/events/batch>
- Body:** JSON (shown in the preview tab)
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {  
2   "accepted": 1,  
3   "deduped": 1,  
4   "rejected": 0,  
5   "rejections": [],  
6   "updated": 0  
7 }
```

## Test Case 2: Same Event ID with New Data → Update

**Objective:** Check event update when receivedTime is newer.

**Step:** Send same eventId again with different payload and newer receivedTime.

**Output:**

Step 1: First request (store original event)

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/events/batch`. The request body is a JSON array containing one event with an eventId of "E-200". The response status is 200 OK, and the response body shows a summary of the event processing: accepted: 1, deduped: 0, rejected: 0, rejections: [], and updated: 0.

```
[{"eventId": "E-200", "eventTime": "2026-01-15T11:00:00Z", "receivedTime": "2026-01-15T11:00:05Z", "machineId": "M-001", "durationMs": 1000, "defectCount": 0}], {"accepted": 1, "deduped": 0, "rejected": 0, "rejections": [], "updated": 0}
```

Step 2: Second request (same eventId, different payload, newer receivedTime)

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/events/batch`. The request body is a JSON array containing the same event with an eventId of "E-200", but with a newer receivedTime of "2026-01-15T11:02:00Z". The response status is 200 OK, and the response body shows a summary of the event processing: accepted: 0, deduped: 0, rejected: 0, rejections: [], and updated: 1.

```
[{"eventId": "E-200", "eventTime": "2026-01-15T11:00:00Z", "receivedTime": "2026-01-15T11:02:00Z", "machineId": "M-001", "durationMs": 1500, "defectCount": 2}], {"accepted": 0, "deduped": 0, "rejected": 0, "rejections": [], "updated": 1}
```

### Test Case 3: Older Received Time → Ignore

**Objective:** Check older event does not overwrite newer data.

**Step:** Send same eventId with older receivedTime.

#### Output:

Step 1: Insert the newer event first

Step 2: Send an older version of the same event

The screenshot shows the Postman application interface. A collection named "factoryevents" is selected. A new POST request is being prepared to the endpoint `http://localhost:8080/events/batch`. The request body is defined as follows:

```
2
{
  "eventId": "E-300",
  "eventTime": "2026-01-15T12:00:00Z",
  "receivedTime": "2026-01-15T12:01:00Z",
  "machineId": "M-002",
  "durationMs": 500,
  "defectCount": 0
}
```

The response from the server is a 200 OK status with a response time of 7 ms and a body size of 231 B. The response body is:

```
1
{
  "accepted": 0,
  "deduped": 1,
  "rejected": 0,
  "rejections": [],
  "updated": 0
}
```

The bottom navigation bar includes links for Postbot, Runner, Vault, and other tools.

## Test Case 4: Invalid Duration → Rejected

**Objective:** Check rejection for invalid duration.

**Step:** Send event with negative or very large durationMs.

**Output:**

Rejected with reason INVALID\_DURATION

The screenshot shows the Postman application interface. In the top navigation bar, the URL `http://localhost:8080/events/batch` is selected. The main workspace displays a POST request to `http://localhost:8080/events/batch`. The request body is set to `JSON` and contains the following JSON payload:

```
3   "eventId": "E-400",
4   "eventTime": "2026-01-15T13:00:00Z",
5   "receivedTime": "2026-01-15T13:00:05Z",
6   "machineId": "M-003",
7   "durationMs": -100,
8   "defectCount": 1
9 }
10 ]
11 ]
```

The response status is **400 Bad Request**, with a timestamp of `2026-01-31T11:03:14.521Z`, status `400`, error `Bad Request`, and path `/events/batch`.

## Test Case 5: Future Event Time → Rejected

**Objective:** Check rejection for future eventTime.

**Step:** Send event with eventTime more than 15 minutes in future.

**Output:**

Rejected with reason EVENT\_TIME\_IN\_FUTURE

The screenshot shows the Postman application interface. A POST request is being sent to `http://localhost:8080/events/batch`. The request body is a JSON object representing an event with the following fields:

```
2 | {
3 |   "eventId": "E-500",
4 |   "machineId": "M-004",
5 |   "eventTime": "2026-02-01T15:00:00Z",
6 |   "durationMs": 1000,
7 |   "defectCount": 0
8 | }
9 |
10 ]
```

The response status is `200 OK`, indicating the request was successful, but the event was rejected. The response body shows the event was rejected because its event time was in the future:

```
{}
{
  "accepted": 0,
  "deduped": 0,
  "rejected": 1,
  "rejections": [
    {
      "eventId": "E-500",
      "reason": "EVENT_TIME_IN_FUTURE"
    }
  ],
  "undelivered": 0
}
```

## Test Case 6: Valid Single Event → Accepted

**Objective:** Check normal valid event ingestion.

**Step:** Send a valid event using /events/batch.

**Output:**

Accepted = 1

The screenshot shows the Postman application interface. The URL in the header is `http://gold-robot-552570.postman.co`. The main request details show a POST method to `http://localhost:8080/events/batch`. The request body is a JSON array with one element:

```
1 [  
2   {  
3     "eventId": "E-701",  
4     "machineId": "M-001",  
5     "eventTime": "2026-01-15T11:00:00Z",  
6     "receivedTime": "2026-01-15T11:00:05Z",  
7     "durationMs": 1000,  
8     "defectCount": 1  
9   },  
-
```

The response section shows a green `200 OK` status with a response time of `147 ms` and a response size of `231 B`. The response body is also a JSON object:

```
{ } JSON < Preview | Visualize >  
1 {  
2   "accepted": 1,  
3   "deduped": 0,  
4   "rejected": 0,  
5   "rejections": [],  
6   "updated": 0  
7 }
```

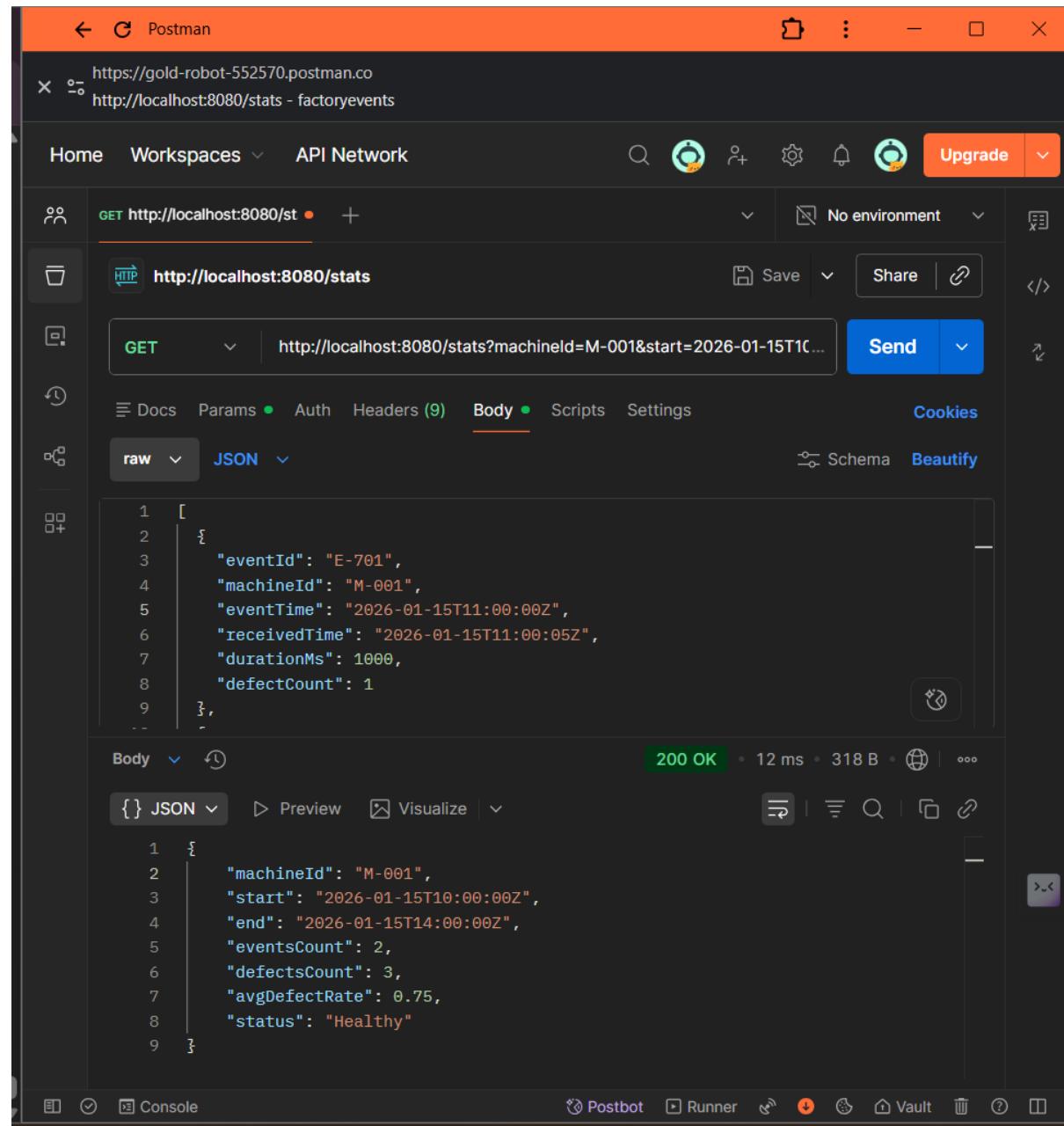
## Test Case 7: Stats API with No Events

**Objective:** Check stats response when no events exist in time range.

**Step:** Call /stats API for a machine with no events.

**Output:**

eventsCount = 0, status = Healthy



The screenshot shows the Postman application interface. A collection named "factoryevents" is selected. A new request is being prepared for the URL `http://localhost:8080/stats`. The request method is set to `GET`, and the URL is `http://localhost:8080/stats?machineId=M-001&start=2026-01-15T10:00:00Z&end=2026-01-15T14:00:00Z`. The "Body" tab is selected, showing the raw JSON response. The response is a 200 OK status with a duration of 12 ms and a size of 318 B. The response body is as follows:

```
1 [  
2 {  
3   "eventId": "E-701",  
4   "machineId": "M-001",  
5   "eventTime": "2026-01-15T11:00:00Z",  
6   "receivedTime": "2026-01-15T11:00:05Z",  
7   "durationMs": 1000,  
8   "defectCount": 1  
9 }]
```

Below the raw JSON, there is a preview of the JSON response, which includes the same data structure with additional summary statistics:

```
1 {  
2   "machineId": "M-001",  
3   "start": "2026-01-15T10:00:00Z",  
4   "end": "2026-01-15T14:00:00Z",  
5   "eventsCount": 2,  
6   "defectsCount": 3,  
7   "avgDefectRate": 0.75,  
8   "status": "Healthy"  
9 }
```

## Test Case 8: Concurrent Event Ingestion

**Objective:** Check system handles multiple requests at same time.

**Step:** Send multiple batch requests simultaneously with same eventId.

**Output:**

Only one event stored, others deduped

The screenshot shows the Postman interface with a test collection named "Concurrent Ingestion Test - factoryevents". The "Run results" section indicates a successful run completed at 08:17:16 PM. It shows 50 iterations, a duration of 4s 228ms, and 0 errors. The "All Tests" tab is selected, showing 0 passed, 0 failed, 0 skipped, and 0 errors. The results are grouped by iteration:

- Iteration 3:** POST http://localhost:8080/events/batch. Response: 200 • 6 ms • 358 B •. No tests found.
- Iteration 4:** POST http://localhost:8080/events/batch. Response: 200 • 4 ms • 358 B •. No tests found.
- Iteration 5:** POST http://localhost:8080/events/batch. Response: 200 • 4 ms • 358 B •. No tests found.

A modal window displays the message: "Run completed. All requests executed." with a progress bar at 100%.