Problem Statement

Build a backend service to capture website analytics events. This service must be able to handle a high volume of incoming "ingestion" requests very quickly, and provide a separate API for retrieving summarized data.

The primary challenge is that the ingestion endpoint must be extremely fast. A good solution will not make the client wait for a database write to complete.

Time Limit

4 hours

Core Requirements

You must build two distinct services or modules that work together.

Service 1: The "Ingestion" API (Must be FAST)

This service receives events. Its only job is to acknowledge the event as fast as possible and pass it on for processing.

- **Endpoint:** POST /event
- **Request Body (JSON):**
  JSON
  ```
  {
    "site_id": "site-abc-123",
    "event_type": "page_view",
    "path": "/pricing",
    "user_id": "user-xyz-789",
    "timestamp": "2025-11-12T19:30:01Z"
  }
  ```

- **Success Action:**
  1. Validate the JSON body (e.g., site_id and event_type are required).
  2. Place the event into an **asynchronous processing queue**.

3. Immediately return Success.

Service 2: The "Processor"

This is a background worker (it is not a public API). Its job is to:

1. Pull events from the queue (created by Service 1).
2. Process these events.
3. Write the events into a db

Service 3: The "Reporting" API

This service reads from the database to provide insights.

- **Endpoint:** GET /stats
- **Query Parameters:** ?
- **Success Response (JSON):**
  - It must **aggregate** the data for the given site_id and optional date.
  - It should return a summary, not a raw list of events.

```
JSON
{
  "site_id": "site-abc-123",
  "date": "2025-11-12",
  "total_views": 1450,
  "unique_users": 212,
  "top_paths": [
    { "path": "/pricing", "views": 700 },
    { "path": "/blog/post-1", "views": 500 },
    { "path": "/", "views": 250 }
  ]
}
```

Deliverables

1. **Source Code:** A link to a Git repository (or a ZIP file).
2. **A README.md file** that explains:
   - **Architecture Decision:** How you implemented the asynchronous processing (the queue) and why.
   - **Database Schema:** A simple description or diagram of your database tables.
   - **Setup Instructions:** Clear, step-by-step instructions to build and run the entire system (the API, the processor, and the database).
   - **API Usage:** Example curl commands to test the POST /event and GET /stats.