The Data Archaeologist: Chronologicon Engine Development Node.js Backend Developer Take-Home Assignment

ArchaeoData Inc.

Unearthing History, Reconstructing Timelines

Introduction & Scenario

Welcome to ArchaeoData Inc., a pioneering organization dedicated to salvaging and analyzing fragmented historical data. Our mission is to reconstruct complete, coherent historical timelines from disparate and often corrupted data sources. We receive historical event data in various forms, primarily as large text files that simulate ancient database dumps or fragmented historical records. Each entry contains partial information about a "Historical Event," including a unique ID, dates, and potential references to other events. The core challenge lies in piecing together these fragments into meaningful, hierarchical narratives.

Your task is to develop the Chronologicon Engine, Temporal Gap Finder and Event Influence Spreader, a robust combination of Node.js backend services that can effectively process, store, and query historical event fragments to reconstruct and analyze complete timelines.

The Chronologicon Engine

Objective: Design and implement a Node.js backend service to ingest, manage, and query historical event data. This assignment is designed to assess your proficiency in modern Node.js development, database interactions, API design, and problem-solving.

1. Backend Implementation Requirements

Your solution should demonstrate mastery of the following concepts:

- **Database Design (PostgreSQL/MySQL Recommended):**
  a. Design a relational schema for HistoricalEvents. Each event must include: i. event_id (Primary Key, UUID type)
  ii. event_name (String)
  iii. description (Text, nullable)
  iv. start_date (Timestamp with Time Zone, indexed)

v. end_date (Timestamp with Time Zone, indexed)

vi. duration_minutes (Integer, a calculated field that should be derived from start_date and end_date within your application logic or a generated column if supported by your DB but stored for quick access).

vii. parent_event_id (UUID, Foreign Key referencing event_id itself, nullable – representing nested or sub-events).

viii. metadata (JSONB/JSON field for additional, unstructured data like original source file name, line number, parsing flags, etc.).

- **File Handling:**
  b. The service must be capable of ingesting large text files (e.g., historical_data_1.txt). Each line in these files represents a potential historical event fragment.
  c. **Input Format:** Each line adheres to the format: EVENT_ID|EVENT_NAME|START_DATE_ISO|END_DATE_ISO|PARENT_ID_OR_ NULL|DESCRIPTION.
    i. Example: a1b2c3d4-e5f6-7890-1234-567890abcdef|Founding of ArchaeoData|2023-01-01T10:00:00Z|2023-01-01T11:30:00Z|NULL|Initial establishment of the company.
  d. Implement robust error handling for malformed lines (e.g., missing fields, invalid date formats, non-UUID IDs). Malformed lines should be logged but not prevent the processing of valid lines.

2. API Endpoints Specification

Assume the base URL is http://localhost:3000.

A. POST /api/events/ingest

- **Description:** Initiates the ingestion of historical event data from a provided text file. This operation should be asynchronous, returning a job ID that can be used to monitor its status.
- **Method:** POST
- **Request:**
  - **Headers:** Content-Type: multipart/form-data (for file upload) OR Content Type: application/json (if providing a server file path for simplicity).

- **Body (JSON with Server File Path):**

```
{
 "filePath": "/path/to/your/server/data/sample_historical_data.txt"
}
```

• **Response (202 Accepted):**

```
{
 "status": "Ingestion initiated",
 "jobId": "ingest-job-12345-abcde",
 "message": "Check /api/events/ingestion-status/ingest-job-12345-abcde for updates."
}
```

B. GET /api/events/ingestion-status/:jobId

• **Description:** Retrieves the current status and progress of an ingestion job. • **Method:** GET

• **URL Example:** http://localhost:3000/api/events/ingestion-status/ingest-job-12345-abcde

• **Response (200 OK - Processing):**

```
{
 "jobId": "ingest-job-12345-abcde",
 "status": "PROCESSING",
 "processedLines": 10,
 "errorLines": 2,
 "totalLines": 15,
 "errors": [
 "Line 11: Malformed entry: 'malformed-id-1|Broken Event|2023-01-02T09:00:00Z|2023-01-02T10:00:00Z|NULL|Missing one field.'",
 "Line 12: Invalid date format for event 'another-bad-line': '2023/01/03 10:00'"
]
}
```

• **Response (200 OK - Completed):**

```
{
 "jobId": "ingest-job-12345-abcde",
 "status": "COMPLETED",
 "processedLines": 10,
 "errorLines": 2,
```

```
"totalLines": 12,
"errors": [
"Line 11: Malformed entry: 'malformed-id-1|Broken Event|2023-01-02T09:00:00Z|2023-01-02T10:00:00Z|NULL|Missing one field.'",
"Line 12: Invalid date format for event 'another-bad-line': '2023/01/03 10:00'"
],
"startTime": "2023-06-25T10:00:00Z",
"endTime": "2023-06-25T10:00:05Z"
}
```


C. GET /api/timeline/:rootEventId

- **Description:** This is a core endpoint. Given a rootEventId, it must return the entire hierarchical timeline, including all its direct and indirect child & parent events, presented as a nested JSON structure.
- **URL Example:** http://localhost:3000/api/timeline/a1b2c3d4-e5f6-7890-1234-567890abcdef
- **Response (200 OK):**


```
{
"event_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
"event_name": "Founding of ArchaeoData",
"description": "Initial establishment of the company, focusing on data salvage.", "start_date": "2023-01-01T10:00:00.000Z",
"end_date": "2023-01-01T11:30:00.000Z",
"duration_minutes": 90,
"parent_event_id": null,
"children": [
{
"event_id": "f7e6d5c4-b3a2-1098-7654-3210fedcba98",
"event_name": "Phase 1 Research",
"description": "Early research on data fragmentation techniques.",
"start_date": "2023-01-01T10:30:00.000Z",
"end_date": "2023-01-01T11:00:00.000Z",
"duration_minutes": 30,
"parent_event_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
"children": [
{
"event_id": "11223344-5566-7788-9900-aabbccddeeff",
"event_name": "Internal Review Meeting",
"description": "Reviewing initial research findings.",
"start_date": "2023-01-01T10:45:00.000Z",
```

```
"end_date": "2023-01-01T11:15:00.000Z",
"duration_minutes": 30,
"parent_event_id": "f7e6d5c4-b3a2-1098-7654-3210fedcba98",
"children": []
}
]
}
]
}
```


D. GET /api/events/search

- **Description:** Allows searching for events based on various criteria. Supports partial string matching for event_name (case-insensitive), date range filtering (start_date_after, end_date_before), pagination, and sorting.
- **Query Parameters:**
  - name: (Optional) Partial match for event_name.
  - start_date_after: (Optional) Events starting after this ISO 8601 date. ○ end_date_before: (Optional) Events ending before this ISO 8601 date. ○ sortBy: (Optional) Field to sort by (e.g., start_date, event_name).
  - sortOrder: (Optional) asc or desc. Defaults to asc.
  - page: (Optional) Page number (defaults to 1).
  - limit: (Optional) Number of results per page (defaults to 10).
- **URL Examples:**
  - http://localhost:3000/api/events/search?name=phase&sortBy=start_date&sortOrder=asc&page=1&limit=5
  - http://localhost:3000/api/events/search?start_date_after=2023-01-05T00:00:00Z&end_date_before=2023-01-10T23:59:59Z

- **Response (200 OK):**


```
{
"totalEvents": 2,
"page": 1,
"limit": 5,
"events": [
{
"event_id": "f7e6d5c4-b3a2-1098-7654-3210fedcba98",
"event_name": "Phase 1 Research"
```

```
},
{
"event_id": "5f6e7d8c-9a0b-1c2d-3e4f-5a6b7c8d9e0f",
"event_name": "Analysis Phase Alpha"
}
]
}
```


E. GET /api/insights/overlapping-events

- **Description:** Returns a list of all distinct event pairs that have overlapping timeframes.
- **Method:** GET
- **Input:** A startDate and endDate defining the overall period of interest. • **URL:** http://localhost:3000/api/insights/overlapping-events • **Response (200 OK):**


```
[
{
"overlappingEventPairs": [
{
"event_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
"event_name": "Founding of ArchaeoData",
"start_date": "2023-01-01T10:00:00.000Z",
"end_date": "2023-01-01T11:30:00.000Z"
},
{
"event_id": "f7e6d5c4-b3a2-1098-7654-3210fedcba98",
"event_name": "Phase 1 Research",
"start_date": "2023-01-01T10:30:00.000Z",
"end_date": "2023-01-01T11:00:00.000Z"
}
],
"overlap_duration_minutes": 30
}
]
```


The "Temporal Gap Finder"

Description: ArchaeoData Inc. sometimes receives historical data with unintentional "temporal gaps"—periods where no events are recorded, indicating

missing data.

Task: Design and implement a Node.js function (which can be exposed as an internal utility or a new API endpoint GET /api/insights/temporal-gaps) that, given a list of HistoricalEvent objects within a specified date range, identifies the largest continuous "gap" in recorded events within that timeframe.

• **Input:** A startDate and endDate defining the overall period of interest. • **Output (for GET /api/insights/temporal-gaps):** An object indicating the start_of_gap, end_of_gap, and durationMinutes for the largest temporal gap. If no significant gaps exist (e.g., only trivial milliseconds, or no events), return an appropriate message.

• **Example Request:** http://localhost:3000/api/insights/temporal gaps?startDate=2023-01-01T00:00:00Z&endDate=2023-01-20T00:00:00Z

• **Example Response (Gap Found):**

```
{
 "largestGap": {
 "startOfGap": "2023-01-10T16:00:00.000Z",
 "endOfGap": "2023-01-15T09:00:00.000Z",
 "durationMinutes": 6780, // Example calculation
 "precedingEvent": {
 "event_id": "9b8a7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d",
 "event_name": "Marketing Campaign Launch",
 "end_date": "2023-01-10T16:00:00.000Z"
 },
 "succeedingEvent": {
 "event_id": "0d9e8f7a-6b5c-4d3e-2f1a-0b9c8d7e6f5a",
 "event_name": "Customer Onboarding Phase",
 "start_date": "2023-01-15T09:00:00.000Z"
 }
 },
 "message": "Largest temporal gap identified."
}
```

• **Example Response (No significant gaps):**

```
{
 "largestGap": null,
 "message": "No significant temporal gaps found within the specified range, or too
        few events."
}
```

The "Event Influence Spreader"

Objective:

Given a source_event_id and a target_event_id, find the **minimum total duration of events** along any valid path from the source event to the target event (following parent-child relationships). If no path exists, indicate that. The "duration" refers to the end_date - start_date of each event along the path.

Description:

ArchaeoData Inc. wants to understand the quickest way "knowledge" or "influence" from one event propagates to another through their hierarchical timeline. They define this propagation as moving from a parent event to its child events. The "cost" of moving through an event is its own duration. Your task is to find the path with the *least cumulative duration* from a specified source event to a target event.

A. GET /api/insights/event-influence

- **Description:** Calculates the shortest temporal path (minimum total duration) between a source and a target event, following parent-child relationships. • **Method:** GET
- **Query Parameters:**
  - sourceEventId: (Required) UUID of the starting event.
  - targetEventId: (Required) UUID of the destination event.
- **URL Example:** http://localhost:3000/api/insights/event influence?sourceEventId=d1e2f3a4-b5c6-7d8e-9f0a 1b2c3d4e5f6a&targetEventId=c6d7e8f9-a0b1-c2d3-e4f5-a6b7c8d9e0f1

Scenario 1: Path Found

Input (Query Parameters):

```
sourceEventId: d1e2f3a4-b5c6-7d8e-9f0a-1b2c3d4e5f6a
targetEventId: c6d7e8f9-a0b1-c2d3-e4f5-a6b7c8d9e0f1
```

A possible path: Project Gaia Initiation -> Algorithm Development -> Model Training -> Deployment Planning

Expected Output (200 OK):


```
{
 "sourceEventId": "d1e2f3a4-b5c6-7d8e-9f0a-1b2c3d4e5f6a",
 "targetEventId": "c6d7e8f9-a0b1-c2d3-e4f5-a6b7c8d9e0f1",
 "shortestPath": [
 {
 "event_id": "d1e2f3a4-b5c6-7d8e-9f0a-1b2c3d4e5f6a",
 "event_name": "Project Gaia Initiation",
 "duration_minutes": 60
 },
 {
 "event_id": "a4b5c6d7-e8f9-a0b1-c2d3-e4f5a6b7c8d9",
 "event_name": "Algorithm Development",
 "duration_minutes": 480
 },
 {
 "event_id": "b5c6d7e8-f9a0-b1c2-d3e4-f5a6b7c8d9e0",
 "event_name": "Model Training",
 "duration_minutes": 960
 },
 {
 "event_id": "c6d7e8f9-a0b1-c2d3-e4f5-a6b7c8d9e0f1",
 "event_name": "Deployment Planning",
 "duration_minutes": 180
 }
 ],
 "totalDurationMinutes": 1680,
 "message": "Shortest temporal path found from source to target event."
}
```


Scenario 2: No Path Found

If you try to find a path from Marketing Campaign Launch to Pilot Project Alpha, which are in different, unrelated branches of the timeline:

Input (Query Parameters):

```
sourceEventId: 9b8a7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d
targetEventId: c8d7e6f5-a4b3-2109-8765-4321fedcba98
```

Expected Output (200 OK with specific message, or 404 Not Found if you prefer):

```
{
 "sourceEventId": "9b8a7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d",
 "targetEventId": "c8d7e6f5-a4b3-2109-8765-4321fedcba98",
 "shortestPath": [],
 "totalDurationMinutes": 0,
 "message": "No temporal path found from source to target event."
}
```


Deliverables

Please provide your solution as a well-organized project.

1. **Backend Code:**
    a. A complete Node.js project.
    b. Clear separation of concerns.
    c. **README.md** file at the root of the backend project with:
        i. Detailed setup instructions (dependencies, how to install, configure database, and run the application).
        ii. Comprehensive API documentation (all endpoints, request/response formats, example curl commands).
        iii. An explanation of your key design choices and how each required concept was addressed.
2. **Database Schema:**
    a. A .sql file containing the DDL (Data Definition Language) script for creating your database tables and indexes.
3. **Frontend Code (if applicable):**
    a. A separate, self-contained folder for the UI project with its own README.md and setup instructions.
4. **Sample Data File:**
    a. The provided sample_historical_data.txt file.

Evaluation Criteria

Your submission will be evaluated based on the following:

- **Correctness & Completeness.**
- **Code Quality**
- **Database Design**
- **API Design**
- **Problem-Solving**

- **Performance and Optimization**
- **Documentation**