# **Healthcare Data Engineering Project Plan – Initial Draft**

## **1. Data Sources**

As the first step, we will identify reliable healthcare-related data sources. These may be APIs or public websites (e.g. Wikipedia, which can be scraped if needed).

**Proposed sources:**

1. **disease.sh API** (<https://disease.sh/docs/#/>)  
   * Provides up-to-date data with many endpoints.
   * Global coverage (not country-specific).
   * Can support both batch and streaming ingestion.
2. **WHO Athena API** (<https://www.who.int/data/gho/info/athena-api-examples>)  
   * Provided by WHO.
   * Multiple endpoints available.
   * Can combine with disease.sh data (In theory)

## **2. Architecture**

We will start with open-source tools with the option to expand into cloud services later if required.

### **Extract**

* Python scripts to fetch data from each source.
* Each extractor will live in its own directory for separation and maintainability.

### 

### **Transform**

* Minimal preprocessing only:  
  + Add timestamps.
  + Standardize naming conventions.
* The raw data itself will not be modified.

### **Load**

* Load raw files into Postgres.
* Store data as single-column JSON tables (no upfront schema mapping).
* Use dbt to parse and transform raw JSON into staging/aggregate tables.
* Add dbt unit tests for validation.

### **Orchestration & Scheduling**

* Use Airflow for job orchestration, scheduling, and alerting.

### **Infrastructure Setup**

* All services run inside Docker containers.
* Each team member will install Docker and run the environment locally.

**Example team workflow:**

1. Install Docker and set up infra locally.
2. Write extractor that fetches data and writes into Postgres.
3. Write dbt model that parses raw JSON into structured tables.
4. Add Airflow DAG to orchestrate extraction → load → transform.

## 

## **3. GitHub Repository**

* Create a GitHub repository.
* Set branch protection rules and agree on PR review process.
* Assign branches to each developer for their tasks.
* Document the architecture and workflow in the README file.

## **4. Implementation Plan**

* Start development after architecture is finalized.
* Each member develops their assigned extractor/dbt model.
* Regular PR reviews and merges into the main branch.

## **5. Future Enhancements**

We can expand with:

* **Visualization:** dashboard
* **CI/CD:** GitHub Actions/similar to automate code integration and deployments.
* **Data Lake:** Add MinIO for file storage.
* **Streaming ingestion:** Explore Kafka or similar.

## **Notes**

* This is the **initial plan** — We can consider this as MVP. So it is flexible and open to improvements.
* The goal is to start with this and build out as we learn.

## **Immediate Tasks (After the team meeting)**

1. Create GitHub repo & set branch rules.
2. Setup Docker (Python + Postgres + Airflow + dbt + optionally Minio).
3. Each member runs Docker locally and commits a small README confirming success.
4. We could then start with one of API’s and implement a working example following the architecture we agreed on.