ASSIGNMENT DOCUMENT — BIG DATA INGESTION + CREDIT-BASED API SYSTEM

Project Title:

Unified Big Data Ingestion, Storage, and Credit-Regulated API Delivery System

---

1. PROJECT OVERVIEW

You are assigned to design and implement a complete solution capable of:

1. **Ingesting extremely large datasets** (datasets provided in mixed formats, schemas, and structures; estimated scale: **70–200 million+ records**, potentially expanding to **700M+ records**).
2. **Normalizing and storing** this data optimally using **technologies of your choice**, with justification.
3. **Exposing this data securely via an API**, with:
    - **Credit-based access control**
    - **Query-based filters**
    - **Usage logs**
    - **Response pagination**
4. **Processing large volumes** efficiently while maintaining:
    - High read/write performance
    - Horizontal scalability
    - Fault tolerance
    - Low API latency

You may choose **any modern big-data stack**, as long as you justify the architecture and demonstrate scalability.

---

2. DATA PROVIDED

You will receive a large volume of files containing structured and semi-structured data in **multiple schemas** and **multiple formats**, including (but not limited to):

Data Types

- CSV (multiple schemas)
- JSON / JSONL
- XML
- Parquet
- SQL dumps
- Excel sheets
- Nested, denormalized, and inconsistent structures
- Files may contain missing fields, malformed records, corrupted lines, or encoding issues

Data Size

- Initial size: **70M–200M+ records**
- Phase II scale: **300M–700M+ records or more**

Download Access

A sample big dataset (for testing and initial ingestion) can be downloaded here:

👉 **Download Big Data:**
Contact us on whatsapp +919004925988 to get the sample data

---

3. YOUR OBJECTIVES

A. Handle Big Data at Scale

You must design an ingestion pipeline that can read and unify data from all provided formats and schemas.

Your system must:

- Support multi-format ingestion
- Automatically detect schema variations
- Normalize & unify fields into a master schema

- Remove duplicates using deterministic logic
- Support incremental ingestion
- Perform data validation & cleansing
- Generate logs and quality reports

You may use any big data or ETL-oriented stack.

B. Choose the Right Technology Stack

You are free to choose **any technologies** but must **justify them** based on:

- Scalability
- Cost efficiency
- Performance
- Maintainability
- Compatibility with API-delivery

Suggested Options (You may choose any):

- **Databases:**
    - BigQuery
    - Snowflake
    - ClickHouse
    - Apache Cassandra
    - PostgreSQL + Citus
    - MongoDB / DocumentDB (if justified)
- **Data Lake / Warehousing:**
    - AWS S3 + Athena
    - Delta Lake / Databricks
    - Apache Hadoop / HDFS
- **Processing Engines:**
    - Apache Spark
    - Flink
    - Dask
    - Airflow / Prefect (orchestration)
- **API Layer:**
    - Node.js / Python FastAPI / Go
    - GraphQL optional

- gRPC optional

C. Build a Credit-Based API System

You must implement a secure API layer that fetches data from your unified dataset.

API Requirements

1. **Authentication**
    - API Key → tied to user account
    - Rate limits per user
    - Credit consumption per request
2. **Credit System**
    - Admin should assign credits to each user
    - Every API request deducts credits
    - If credits exhausted → API returns: `"Insufficient Credits"`
3. **Querying / Filtering** The API must allow:
    - Filter by attributes
    - Range queries
    - Wildcard / fuzzy search (if supported by DB)
    - Pagination of results
    - Metadata return (`total_records`, `credits_used`, `response_time`)
4. **Response Structure**
    - JSON + optional CSV export
    - Maximum items per request should be configurable
5. **Usage Logs**
    - Store each API call with timestamp
    - Record credits deducted
    - Store query parameters
    - Store execution duration

---

4. SYSTEM ARCHITECTURE REQUIREMENTS

You must deliver:

A. Full Architecture Diagram

Include:

- Ingestion layer
- Processing layer
- Storage layer
- API layer
- Authentication layer
- Credit system
- Monitoring components

B. Data Model Documentation

You must provide:

- Unified master schema
- Field definitions
- Data dictionary
- Example queries

C. Infrastructure Choice

Indicate whether you choose:

- On-prem
- AWS
- GCP
- Azure
- Hybrid

Provide justification for:

- Compute layer
- Storage type
- Caching layer (Redis, Memcached, etc.)

- Load balancing strategy

---

5. DELIVERABLES

The assignee must submit all the following:

Deliverable 1 — Technical Proposal Document

A detailed document explaining:

- Technologies chosen
- Why they were chosen
- Architecture explanation
- Scalability considerations
- Fault tolerance design
- Cost estimate (rough)
- Security considerations

Deliverable 2 — Data Ingestion Pipeline

- Scripts / code to import & normalize data
- Handling of multiple schemas
- Validation + cleansing logic
- Duplicate removal logic
- Benchmark results (ingest speed on sample dataset)

Deliverable 3 — Master Dataset Storage

You must deliver:

- Final stored dataset in database/storage solution
- Indexing and partition strategy
- Performance optimization report

Deliverable 4 — API Implementation

API documentation must include:

- Endpoints
- Parameters
- Sample requests
- Sample responses
- Error codes
- Authentication method
- Credit usage rules

Deliverable 5 — Credit Management System

- Admin panel or script to assign credits
- Credit deduction logic
- Database schema for users & credit logs

Deliverable 6 — Monitoring & Logging

Provide:

- API logs
- Ingestion logs
- Error logs
- Performance dashboard (optional)

Deliverable 7 — Deployment Strategy

Explain how your system will be deployed:

- Docker
- Kubernetes
- Serverless
- Bare metal

Include CI/CD recommendations.

---

6. ACCEPTANCE CRITERIA

Your submission will be evaluated on:

✔ Technical depth

✔ Scalability of chosen architecture

✔ API correctness & reliability

✔ Data quality of ingested dataset

✔ Documentation clarity

✔ Security and credit-control reliability

✔ Performance benchmarks

---

7. BONUS POINTS (Optional)

You may implement any of the following for higher scoring:

- Caching layer (Redis) to optimize API speed
- Async processing for faster ingestion
- API analytics dashboard
- Rate-limiting per user tier
- Bulk download endpoints
- Integration tests
- Dockerized demo environment

---

8. SUBMISSION FORMAT

Submit the following as a packaged delivery:

```
/project-root
  /docs
    architecture.pdf
    schema-definition.pdf
    api-spec.md
    credit-system.md
  /src
    ingestion/
    api/
    utils/
  /deployment
    docker-compose.yml
    kubernetes/
  /sample-output
    api-log.json
    unified-dataset-sample.csv
```

A **GitHub repository** is preferred.

---

9. SUPPORTING MATERIALS

Sample Big Dataset (Download)

Contact us on whatsapp +919004925988 to get the sample data

Optional Reference APIs

- OpenAI Credits API
- Stripe Rate-Limited API
- Algolia Search API

---

10. END OF ASSIGNMENT

You must propose, design, implement, document, and deliver a fully functional **Big Data + Credit-Regulated API System** that can handle datasets with up to **700M+ records** and scale reliably as the dataset grows.