# **Data Engineering Final Project**

## **Overview**

For your final project, you will build a complete end-to-end data engineering solution based on your mid-semester project **(after fixing the solution based on feedback)**. This will involve implementing a full data pipeline including data ingestion, processing, storage, and orchestration using modern big data technologies. The project will test your ability to apply concepts learned throughout the course in a practical, real-world scenario.

## **Technical Requirements**

### **Required Technologies**

1. **Data Storage & Table Format**
   * Apache Iceberg for table format
   * MinIO (S3-compatible storage) organized in bronze/silver/gold layers
2. **Processing**
   * Apache Spark for batch and streaming data processing
3. **Streaming**
   * Apache Kafka for real-time data source
   * Python producer for generating messages
4. **Orchestration**
   * Apache Airflow for scheduling ETL pipelines
5. **Data Modeling**
   * Build upon your mid-semester project data models
   * Use mermaid.js to define all table structures
6. **Data Quality**
   * Implement basic data quality checks
7. **Documentation**
   * Full documentation and setup instructions (I will run your project locally on my computer using docker, so the docker image your will provide should be very simple to use)

### **Bonus Components**

* Implement data quality checks with Great Expectations
* Add data lineage tracking with DataHub

**DON’T USE**: Hive, HDFS, any technologies we didn’t learn that chatGPT suggests you to use

## **Project Structure**

Your implementation must use a separated multi-service architecture with the following components:

/project-root

├── /orchestration # Airflow components

├── /streaming # Kafka and producers

├── /processing # Spark applications

**├── README.md # How to start the project on a local machine**

└── docs/ # Project documentation

Each component should have its own Docker Compose file for isolated development and testing, while still being able to communicate via shared networks.

## 

## **Implementation Requirements**

### **1. Data Modeling**

Use mermaid.js to define your data models (you don’t need to do it from scratch, apply improvements to the mid-semester project).

### **2. Data Sources**

Building on your mid-semester project, implement:

* **Batch data source**: At least one dataset loaded into Iceberg tables
* **Streaming data source**: Real-time data stream using Kafka
* **Late-arriving data**: Implement logic to handle data that arrives out of order up to 48 hours after event time

### **3. Processing Logic**

* **Batch Processing**: ETL jobs for transforming data between bronze, silver, and gold layers
* **Stream Processing**: Real-time processing of Kafka messages
* **Data Quality**: Validation checks at different stages of the pipeline

### **4. Data Modeling**

* Implement the data models you designed in your mid-semester project
* Ensure proper implementation of fact and dimension tables
* Implement at least one Type 2 SCD (Slowly Changing Dimension)

### **5. Orchestration**

* Define DAGs in Airflow to schedule both batch and streaming jobs
* Implement proper dependencies between tasks
* Add appropriate error handling and alerting

**Guide to get started with the docker environment:**

If you’re struggling with building the environment, you can use the following guide as reference.   
Please note it doesn’t include kafka

<https://medium.com/@ahmed.trifa1/spark-iceberg-and-airflow-quickstart-369956327fe0>

## **Submission Requirements**

### **1. GitHub Repository**

Submit a link to your GitHub repository containing:

* Complete source code
* 3 Docker Compose files (one for each component)
* Comprehensive README with setup instructions (I will need to run your project locally on my computer and check that your demo really works end-to-end)

### **2. Link to Presentation Video (10 minutes)**

Create a 10-minute presentation video covering (don’t speed up the video):

* Project overview and business context
* Architecture overview
* Data model of bronze, silver and gold
* Show your dags and explain what each step does
* Implementation challenges and solutions

### **3. Link to Demo Video**

Record a demonstration video showing:

* Setup and deployment of your solution (so I could reproduce your steps)
* Real-time data flowing through Kafka
* Airflow scheduling Spark jobs
* Spark jobs processing data
* Data quality checks (bonus)
* Data lineage (bonus)

### **4. Documentation (inside GitHub Repo)**

* README file with setup instructions
* Architecture diagrams
* Data model documentation
* Component descriptions
* Data quality checks (must)