Universitat Politecnica de Catalunya

Big Data Management

Implementation of a (Big) Data Management Backbone[1]

Part 1 - The Landing Zone

Team BDMA11-C	Under the guidance of
HanLing Hu **Lucia Victoria Fernandez Sanchez** **Filipe Albuquerque Ito Russo**	Prof. BESIM BILALLI

2025

[1] GitHub Repository

Contents

Chapter 1

Introduction

University dropout rates remain a persistent global issue. The prolonged time to graduation and early withdrawals impose financial and emotional burdens on students while simultaneously straining institutional resources. With the growing availability of data from digital learning platforms, health surveys, and student services, we can now model, predict, and intervene earlier. However, educational organizations face fragmented data storage, lack of real-time processing capabilities and dedicated personnel, which limits the utility of this data.

Our project aims to tackle the problem by designing a robust data engineering pipeline that ingests and processes diverse data types to support the higher education community, by providing risk ranking of students more likely to drop out and clustering to identify different dropout typologies. These insights are crucial for universities and policymakers to deploy timely and personalized interventions.

1.1 Pipeline Architecture

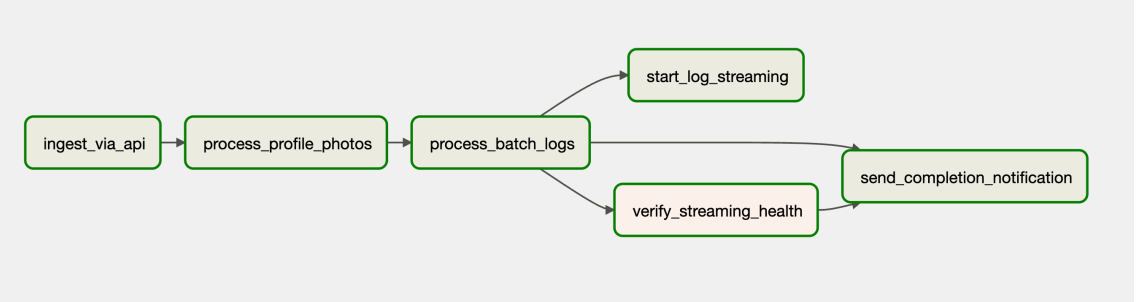

Figure 1.1: Pipeline Architecture

Chapter 2

Data Sources

Our pipeline integrates five core data domains to build a comprehensive view of student engagement, health, and risk factors:

Student and Course Information. This data was sourced from the Open University Learning Analytics dataset [1]. We anonymized the student records by generating synthetic names using the `Faker` library in Python. The cleaned dataset was split into two separate tables: `students.csv` (containing synthetic names and demographic placeholders) and `studentInfo.csv` (retaining only the `id_student` as a foreign key for downstream joins). This separation ensures GDPR compliance while maintaining relational integrity across datasets.

Mental Health Surveys. Survey data obtained from Kaggle's "Students Mental Health Assessments" dataset [2] underwent preprocessing to remove fields such as `Age` and `Gender`. To align the survey responses with our core dataset, we assigned existing `id_student` values from our synthetic student table. Redundant attributes like `Course` and `CGPA` were dropped, as they are already represented in the primary student records.

Physical Health Records. The physical health data was also sourced from a Kaggle dataset [3]. We removed repeated and unnecessary information including `Name`, `Age`, `Gender`, and `Hospital`. Billing values were normalized by capping or scaling outliers to reflect reasonable amounts. Each healthcare record was then mapped to a student by randomly sampling from the existing `students.csv` file,

maintaining a consistent `id_student` key across domains.

Interaction and Performance Logs. This dataset, obtained from GitHub [4], includes detailed logs of student behavior in a virtual learning environment. It reflects actions such as login frequency, resource views, and submission timing. The unique user identifier (`UID`) was transformed to align with the `id_student` format from other datasets, allowing us to unify schema and cross-reference behavioral features with academic performance and health data.

Unstructured Profile Photos. Profile photo data was sourced via the Kaggle API from a dataset of human profile images [5]. We organized these into folders by student ID, naming them systematically (e.g., `student_001/photo1.jpg`). Using our ingestion pipeline, these images were converted into binary format and stored in Delta Lake alongside metadata such as `photo_id`, `student_id`, and `ingestion_time`. This allows unstructured visual data to be integrated with structured features for future machine learning applications.

2.1 Summary of data pipeline for P1

Task Name	Description	Storage Location
`ingest_via_api`	Triggers ingestion of structured CSVs via REST API	`/data/delta/structured`
`process_profile_photos`	Converts unstructured images with metadata into Delta tables	`/data/delta/photos`
`process_batch_logs`	Transforms and joins VLE logs using Spark	`/data/delta/logs`
`start_log_streaming`	Launches Spark Structured Streaming for real-time data	`/data/delta/streaming`
`verify_streaming_health`	Verifies active Spark streaming jobs	N/A
`send_completion_notification`	Sends a status email upon DAG success	N/A

Table 2.1: Summary of Ingestion and Storage Tasks

Chapter 3

Data Ingestion and Storage Strategy

3.1 Overview

Our architecture leverages a modular data pipeline constructed using **Apache Airflow**, **Docker**, and **Apache Spark**, designed to ingest, transform, and persist both structured and unstructured data in a **Delta Lake** format. This design supports both batch and streaming data ingestion, ensuring high reliability through fault tolerance mechanisms, schema enforcement, and interoperability across microservices.

The selection of Delta Lake as the storage layer is deliberate—it provides ACID transaction support, scalable storage, and schema evolution, making it well-suited for analytics workloads. Airflow is employed for orchestration due to its declarative DAG structure, ease of scheduling. Apache Spark is selected for its scalability and support for both batch and streaming processing, enabling us to handle diverse data ingestion requirements effectively.

3.2 Ingestion Process

Data ingestion is orchestrated by Apache Airflow, with tasks encapsulated in Docker containers for environment consistency and reproducibility. API-based ingestion tasks are responsible for collecting structured datasets, including `students`,

`courses`, `assessments`, and `health records`. These APIs, hosted on port `8000`, parse incoming data, enforce schema validation, and persist validated records to Delta Lake at `/data/delta`.

For unstructured data such as student profile photos, a dedicated Spark job processes image binaries. Metadata such as photo ID and ingestion timestamps are extracted and associated with the files, which are then stored in binary format under the appropriate Delta Lake partition.

Batch jobs handle transformation of large VLE interaction datasets (e.g., logs, clicks), running within Dockerized Spark environments. These jobs focus on cleaning, normalization, and enrichment, ensuring consistency before data is persisted.

To support near-real-time analytics, a streaming Spark job is launched by Airflow to monitor incoming student behavior logs (e.g., clickstream data). This job utilizes Spark Structured Streaming to process events in real-time and writes output to Delta Lake. For prototyping purposes, the streaming job is time-boxed to terminate after a few seconds. However, Streaming with Spark can lead to latency in high-load scenarios without proper tuning.

3.3 Storage Strategy

All curated data is written to Delta Lake storage mounted at `/data/delta`. Delta Lake enables ACID transactions and scalable storage optimized for analytics. The data lake is organized into subfolders corresponding to each data domain (e.g., `structured/`, `photos/`, `logs/`), enabling fine-grained access and tracking. Currently, the Delta Lake is deployed locally for simplicity, cost-efficiency, and faster development iterations. This avoids the overhead of managing cloud resources during early prototyping and testing. However, a cloud-based data lake would be the natural evolution for production-scale deployment.

Bibliography

[1] O. University, "Open dataset for student and course information," https://analyse.kmi.open.ac.uk/open-dataset, accessed: 2025-04-04.

[2] S. Sonia, "Students mental health assessments," https://www.kaggle.com/datasets/sonia22222/students-mental-health-assessments, 2021, accessed: 2025-04-04.

[3] N. Chahar, "Student health dataset," https://www.kaggle.com/datasets/nikhilchahar/student-health-dataset/data, 2021, accessed: 2025-04-04.

[4] M. Riestra, "Moodle early performance prediction dataset," https://github.com/moisesriestra/moodle-early-performance-prediction/tree/master/data, 2020, accessed: 2025-04-04.

[5] O. K. Kurnaz, "Human profile photos dataset," https://www.kaggle.com/datasets/osmankagankurnaz/human-profile-photos-dataset, 2021, accessed: 2025-04-04.