Ministry of Higher Education and Scientific Research
University of Abdelhafid Boussouf - Mila
Institute of Mathematics and Computer Science
Department of Computer Science
Master 2 I2A – Big Data
2025/2026

Mini-Project: Building the "DzStore" Data Platform

Theme: End-to-End E-Commerce Analytics & Recommendation System **Value:** 12 Points (Practical Works Grade)
Deadline: 04/01/2026

1. Context & Objectives

DzStore is a rapidly growing Algerian electronics retailer. The CEO wants to modernize the platform to become "Data-Driven." The current system is slow and offers no personalization.

Your Mission: As the Lead Data Architect, you must build a new backend infrastructure that supports:

1. **Scalable Storage:** Storing products, users, and orders in a NoSQL database (MongoDB).
2. **High-Velocity Ingestion:** Handling massive logs of user activity (simulated CSVs).
3. **Intelligent Processing:** Using Apache Spark to calculate "Trending Products" and train a Recommendation Model.
4. **Data Serving:** A web dashboard to display the results to the marketing team and users.

The Architecture

1. **Source:** Python Data Generator (Simulates User Traffic).
2. **Storage Layer:** MongoDB (Operational Data) + Local/HDFS (Raw Logs).
3. **Processing Layer:** Apache Spark (ETL + MLlib).
4. **Serving Layer:** Streamlit Web App (Reads from MongoDB).

2. Project Phases

Phase 1: Data Generation (The "Simulator")

- **Goal:** Create realistic data to stress-test your system.
- **Tools:** Python (`faker`, `random`, `pandas`).
- **Requirements:** Write a script `data_generator.py` that generates:
  1. **Products:** 50+ items (Laptops, Phones) with embedded specs.
  2. **Users:** 1,000+ profiles.
  3. **Orders:** 2,000+ transactions. Use the **Snapshot Pattern** (embed product details in the order).
  4. **Traffic Logs:** A massive CSV file (`logs.csv`) with 10,000+ rows representing user clicks:
     `timestamp, user_id, product_id, event_type (view/cart/buy)`

Tip: Introduce "Bias" in your generation. Make "iPhone 15" have 500 views and "Unknown Phone" have 0 views so your Trends algorithm has something to find!

Phase 2: The Trend Engine (Spark ETL)

- **Goal:** Calculate "What is hot right now?"

- **Tools:** PySpark Core / SQL.
- **Task:** Write a Spark job `spark_trends.py` that:
    1. Reads the `logs.csv` file.
    2. Filters for `view` events.
    3. Aggregates counts by `product_id`.
    4. Joins with the `products` collection (from MongoDB) to get the product Name.
    5. **Writes** the Top 10 results to a new MongoDB collection: `analytics_trends`.

```
# Expected Output in MongoDB (analytics_trends):
{ "rank": 1, "product_name": "PlayStation 5", "views": 850 }
{ "rank": 2, "product_name": "iPhone 15",     "views": 620 }
```

Phase 3: The Recommendation AI (Spark MLlib)

- **Goal:** "Users who bought X also bought Y."
- **Tools:** PySpark MLlib (ALS - Alternating Least Squares).
- **Task:** Write a Spark job `spark_recs.py` that:
    1. Reads the `orders` collection from MongoDB.
    2. Creates a rating matrix (Implicit Feedback):
        - If User A bought Product B $\rightarrow$ Rating = 1.
    3. Trains an ALS Model.
    4. Generates top 3 product recommendations for every user.
    5. **Updates** the `users` collection in MongoDB by adding a new field `recommendations`.

```
# Updated User Document:
{
  "_id": "user_101",
  "name": "Amine",
  "recommendations": [
    { "product": "AirPods", "confidence": 0.89 },
    { "product": "Case",    "confidence": 0.75 }
  ]
}
```

Phase 4: The Frontend Dashboard (Visualization)

- **Goal:** Prove the data is accessible.
- **Tools:** Streamlit (Python).
- **Task:** Build a simple app `app.py` with two pages:
    1. **Dashboard:** Display a Bar Chart of the "Trending Products" (read from `analytics_trends`).
    2. **User Profile:** A dropdown to select a User ID. When selected, show:
        - Their name.
        - **"Recommended for you":** A list of products from their `recommendations` field.

3. Evaluation Criteria (Total: 12 Pts)

Component	Points	Criteria
Data Architecture	**3 pts**	Correct MongoDB Schema (Polymorphism, Embedding). Realistic data generation.
Spark ETL	**3 pts**	Correct Aggregation pipeline. Successful Write-Back to MongoDB.
Machine Learning	**3 pts**	Successful Training of ALS model. Recommendations are saved to DB.
Full Stack Integ.	**3 pts**	The Web App works. It reads the *processed* data (not raw logs).

4. Resources & Hints

1. MongoDB Connector for Spark: You must configure the JAR packages correctly.

```
spark = SparkSession.builder \
    .config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:10.2.1") \
    ...
```

2. Streamlit Boilerplate:

```
import streamlit as st
from pymongo import MongoClient

st.title("DzStore Analytics Dashboard")
client = MongoClient("YOUR_URI")
db = client['dzstore_db']

# Widget: Trending
st.header(" Hot Products")
trends = list(db.analytics_trends.find().sort("views", -1).limit(5))
for item in trends:
    st.write(f"{item['rank']}. {item['product_name']} ({item['views']} views)")
```

GOOD LUCK! BUILD SOMETHING IMPRESSIVE!