### **Project Title: End-to-End Data Engineering Project – Wistia Video Analytics**

### **🧠 Business Objective**

The marketing team uses **Wistia** to track video engagement across **Facebook** and **YouTube**. We aim to:

* Collect **media-level** and **visitor-level** analytics from Wistia’s **Stats API**
* Build an automated data pipeline to ingest and analyze performance
* Use these insights to improve marketing strategies

This project simulates a **real-world data engineering assignment** with full responsibility placed on the student team to **design, implement, and operate** the system.

### **🔍 Project Scope**

**You will be responsible for:**

* Designing the entire system architecture: ingestion, storage, processing, and reporting
* Authenticating and ingesting data from the Wistia Stats API (both media and visitor level)
* Handling **pagination** and **incremental data pulls**
* Running the pipeline in **production mode for 7 days**
* Implementing **CI/CD** using GitHub.
* Documenting decisions, assumptions, and tradeoffs

### **🛠️ Technical Constraints & Freedom**

* You **must design the architecture yourself** and present it to **SME for approval**
* ***DO NOT USE DBT FOR TRANSFORMATION.***
* ***DO NOT USE ANYTHING APART FROM AWS/Azure.***
* You **must use GitHub** for version control and CI/CD
* You **must use Python** for API ingestion and **PySpark** for data transformation.
* The pipeline should run for **7 days**.

### 

### **🔑 Credentials**

Wistia API Documentation: <https://docs.wistia.com/reference/getmedias-media-id-json>

**Wistia API token**.(DO NOT SHARE)  
0323ade64e13f79821bdc0f2a9410d9ec3873aa9df01f8a4a54d4e0f3dd2e6b4

***Media Ids to be used: gskhw4w4lm & v08dlrgr7v***

### **🐍 Example Python Snippet to Connect to Wistia**

import requests

# Wistia API Configuration

API\_TOKEN = "0323ade64e13f79821bdc0f2a9410d9ec3873aa9df01f8a4a54d4e0f3dd2e6b4"

MEDIA\_ID = "v08dlrgr7v" # The given media ID

# Wistia Stats API Endpoint

url = f"https://api.wistia.com/v1/stats/medias/{MEDIA\_ID}.json"

# API Headers

headers = {

"Authorization": f"Bearer {API\_TOKEN}"

}

# Make API Request

response = requests.get(url, headers=headers)

# Handle Response

if response.status\_code == 200:

stats = response.json()

print("✅ Video Stats Retrieved Successfully:\n")

print(stats)

elif response.status\_code == 401:

print("❌ Unauthorized: Check your API token permissions.")

elif response.status\_code == 404:

print("❌ Error: Media not found. Check if the media ID is correct.")

else:

print(f"⚠️ Error: Received status code {response.status\_code} - {response.text}")

### **✅ Requirements Summary**

|  |  |
| --- | --- |
| **ID** | **Requirement** |
| FR1 | **Design** your own architecture for ingestion, processing, and storage |
| FR2 | Authenticate to **Wistia Stats API** using token-based Basic Auth |
| FR3 | Extract **media metadata** (title, ID, hashed\_id, created\_at, etc.) |
| FR4 | Extract **engagement metrics** (plays, play rate, watch time, etc.) |
| FR5 | Extract **visitor-level data** (IP, engagement events) |
| FR6 | Implement **pagination** to fetch all pages of results |
| FR7 | Implement **incremental ingestion** based on created\_at/updated\_at |
| FR8 | Run this pipeline in **"production mode" for 7 consecutive days** |
| FR9 | Implement a **CI/CD pipeline** using GitHub Actions or equivalent |
| FR10 | Store results in a **structured data model** (DWH or cloud database) |
| FR11 | Create **final reports or dashboards** for insights (optional) |
| FR12 | Submit a **GitHub repo** with documentation, pipeline code, CI/CD setup, and instructions |

### 📊 Data Model (Simplified)

#### dim\_media

### media\_id

### title

### url

### channel (Facebook, YouTube)

### created\_at

#### dim\_visitor

### visitor\_id

### ip\_address

### country

#### fact\_media\_engagement

### media\_id

### visitor\_id

### date

### play\_count

### play\_rate

### total\_watch\_time

### watched\_percent

### 📅 Suggested Milestones

|  |  |
| --- | --- |
| Week | Milestone |
| 1 | API Exploration + Authentication |
| 2 | Extract & stage raw data |
| 3 | Design & build DWH schema |
| 4 | Load data into dimensional model |
| 5 | Build dashboard + write final report |
| 6 | Submit GitHub repo + recorded walkthrough |

### 

### **🧠 Evaluation Criteria**

|  |  |
| --- | --- |
| **Area** | **Criteria** |
| Architecture | Clear, scalable, modular design |
| Data Quality | Correct use of pagination, incremental logic, and schema |
| Engineering | Effective error handling, retries, logging |
| CI/CD | Working CI/CD for deployment or validation |
| Documentation | README + architecture diagram + setup instructions |