Lab Assignment: Object Storage on Amazon S3

Problem Statement

A university is moving its learning management system (LMS) to the cloud. Instead of storing files on local servers, they want to use Amazon S3 to store and manage academic resources.

The university requires:

- Faculty to upload lecture slides (PDF), recorded lectures (MP4), and datasets (CSV).
- Students to download these resources and upload their assignment submissions.
- Teaching Assistants (TAs) to replace or delete incorrect files.
- Admins to manage all stored content.

You have been hired to design an S3 storage layout and implement CRUD operations using Python and boto3.

Requirements and Rules

1. **AWS Setup (First Step for All Students)**

Before starting coding, every student must:

- Sign in to AWS Console using their AWS account.
- Go to IAM (Identity and Access Management) and create a new user with Programmatic access.
- Attach policy: AmazonS3FullAccess (for this lab only).
- Download the Access Key ID and Secret Access Key.
- Save them in a .aws/credentials file (or use aws configure command).

2. **Bucket Name**
   Each student must create a unique bucket.
   Rule:

   - Bucket names must be globally unique.
   - Only lowercase letters, numbers, and hyphens allowed.

3. **S3 Structure**

   Each bucket must follow a proper structure

   Rules:

   File types must be correct:

- Slides → .pdf
- Lecture → .mp4
- Datasets → .csv
- Submissions → .pdf or .zip
- Announcements → .txt

4. **Enable Versioning**

   Students must enable versioning on their bucket.
   This ensures old files are not lost when new versions are uploaded.

   Rule:

   - Uploading a file with the same name creates a new version.

5. **Enable Versioning**

   Students must enable versioning on their bucket.

   This ensures old files are not lost when new versions are uploaded.

   Rule:

   - Uploading a file with the same name creates a new version.

6. **Use Presigned URLs**

   Students must generate presigned URLs for secure file sharing.

   Rules:

   - Generate presigned URL for:
   - One lecture slide (slides.pdf)
   - One student assignment submission (assignment.pdf)
   - URL must expire in 1 hour.

Data Creation Guidelines

For this lab, each student will simulate at least 2 university courses (e.g., CS101 and CS202) inside their S3 bucket. Each course will have its own resources and submissions.

1. **Lecture Slides (PDF)**

- Format: .pdf
- Content: A 2–3 page dummy file with random text.
- Quantity: At least 2 slides per course (Week 1 & Week 2).

- Example:
    - courses/CS101/weeks/week01/slides.pdf
    - courses/CS101/weeks/week02/slides.pdf
    - courses/CS202/weeks/week01/slides.pdf

2. **Recorded Lectures (MP4)**

- Format: .mp4
- Content: Small sample video (10–30 seconds).
- Quantity: At least 1 lecture per course.
- Example:
    - courses/CS101/weeks/week01/lecture.mp4
    - courses/CS202/weeks/week01/lecture.mp4

3. **Datasets (CSV)**

- Format: .csv
- Content: Small dataset (marks, attendance).
- Quantity: At least 1 dataset per course.
- Example:
    - datasets/CS101/marks.csv
    - datasets/CS202/attendance.csv

4. **Student Submissions (PDF/ZIP)**

- Format: .pdf or .zip
- Content: Dummy assignments from different students.
- Quantity: At least 2 submissions per course.
- Example:
    - submissions/CS101/assignment1/2023001/assignment.pdf
    - submissions/CS101/assignment1/2023002/assignment.pdf
    - submissions/CS202/assignment1/2023010/assignment.pdf

5. **Announcements (TXT)**

- Format: .txt
- Content: Short announcement text.
- Quantity: At least 2 announcements total (can be for any course).
- Example:
    - announcements/2025-09-25.txt
    - announcements/2025-10-05.txt

Minimum Data Requirement (Per Student)

Each student must create data for at least 2 courses.

- Per course:
    - 2 lecture slides (PDFs)
    - 1 recorded lecture (MP4)
    - 1 dataset (CSV)
    - 2 student submissions (PDF/ZIP)
- Shared across courses:
    - 2 announcements (TXT)

In total: ~12–14 files per student.

Scenarios for CRUD Operations on University LMS (S3)

1. Create (Upload)

Scenario 1:
A professor of CS101 prepares lecture materials for Week 1. He has a slides.pdf and a lecture.mp4 file. Since the university now uses S3, he uploads both into the correct folder:

courses/CS101/weeks/week01/

This ensures that all students can later access the lecture resources.

Your Task: Upload a slides.pdf and lecture.mp4 file into the correct folder in your bucket.

Scenario 2:
A student named Ali completes his Assignment 1 for CS101. He needs to submit it online, so he uploads his assignment.pdf into:

submissions/CS101/assignment1/2023001/

This makes his assignment available for grading.

Your Task: Upload one sample student submission (assignment.pdf) under your submissions folder.

2. Read (Retrieve/Download)

Scenario 1:
A student is preparing for an exam and wants to review lecture notes. Instead of downloading directly from the S3 console, the system generates a presigned URL that allows him to download slides.pdf securely for 1 hour.
This ensures only authorized students access the file.

Your Task: Generate a presigned URL for slides.pdf and test it to confirm it works.

Scenario 2:
A TA wants to check all resources uploaded for CS202 Week 2. By listing objects inside:

courses/MATH202/weeks/week02/

The TA can quickly verify whether both the slides and lecture video are available.

Your Task: List all objects inside the folder for Week 2 of your second course.

3. Update (Modify)

Scenario 1:
The professor realizes that the uploaded slides.pdf for CS101 Week 1 had a mistake. The TA re-uploads the corrected file with the same name. Since versioning is enabled, S3 automatically stores the new version while keeping the old one.
Now students can see the updated slides without losing the older version.

Your Task: Re-upload a corrected version of slides.pdf to demonstrate versioning.

Scenario 2:
The administration office wants to update an announcement.
File: announcements/2025-09-25.txt originally said:

Assignment 1 deadline: Sept 30

But now the deadline is extended to Oct 5. The announcement file is updated with the new text. Students accessing the file will always see the latest announcement.

Your Task: Update the contents of one announcement file with new text.

4. Delete (Remove)

Scenario 1:
The professor uploaded an incorrect dataset file:

datasets/CS101/temp.csv

Since it's not needed, he deletes it from the bucket.
This keeps the storage clean and prevents confusion.

Your Task: Delete one unwanted dataset file from your bucket.

Scenario 2:
A student accidentally uploaded the wrong assignment file under his ID. The TA deletes that submission so the student can re-upload the correct one.
This prevents multiple incorrect versions from piling up.

Your Task: Delete one sample student submission file from your bucket.

Scenario 3:
To verify deletion, the TA runs a head_object query. Since the file no longer exists, S3 returns a 404 error.
This confirms the file was successfully deleted.

Your Task: Verify the deletion of a file by running head_object and confirm that a 404 error is returned.

Lab Report Requirements (Final Deliverables)

At the end of the lab, each student must submit a report (PDF/Word) including:

1. Python Scripts –Brief working details with code used for performing CRUD operations (Create, Read, Update, Delete) on S3 using boto3.
2. Amazon S3 Snapshots – Screenshots of bucket creation, object uploads, versioning, presigned URLs, and deletions.
3. Evidence of CRUD Operations – Show how data in the bucket changes after each operation (upload → new version → presigned URL → delete).