Online Learning Platform using AWS Serverless Architecture

Presented by: Subhashini & Aparna  
Login 360, Chennai

# Abstract

The Online Learning Platform is a serverless web application developed using Amazon Web Services (AWS). It enables students to enroll, select courses, and access video content securely hosted in AWS S3. The backend is powered by AWS Lambda and API Gateway, while DynamoDB stores student and course data. CloudFront ensures fast and reliable content delivery. This project demonstrates how AWS serverless architecture can create scalable and cost-efficient e-learning solutions.

# Introduction

The rapid adoption of e-learning has led to the need for scalable, low-maintenance solutions. Traditional systems often rely on dedicated servers and databases, increasing cost and maintenance overhead. This project uses AWS serverless architecture to develop a lightweight, cost-efficient, and highly available online learning platform without the need to manage servers.

# Objectives

• Automate student enrollment and course selection.

• Enable secure video playback through AWS S3 and CloudFront.

• Store and manage student progress using DynamoDB.

• Leverage AWS IAM for secure access control.

• Demonstrate a fully serverless application model using AWS Lambda and API Gateway.

# System Architecture

The platform follows a serverless architecture utilizing AWS components such as Lambda, API Gateway, S3, DynamoDB, and CloudFront. IAM manages permissions between services. The following diagram illustrates the architecture flow:

A diagram of a software development process

AI-generated content may be incorrect.

**Architecture Flow:**  
1. User interacts with frontend hosted on S3.  
2. API requests are handled by API Gateway.  
3. AWS Lambda executes backend logic.  
4. Data is stored in DynamoDB.  
5. Videos are delivered via S3 and CloudFront.  
6. IAM manages secure communication between services.

# Modules Description

* 1. Student Enrollment

A screenshot of a computer

AI-generated content may be incorrect.

Handles new student registration through Lambda and stores information in DynamoDB.

* 2. Course and Video Selection

A screenshot of a computer

AI-generated content may be incorrect.

Displays available courses and fetches video URLs from S3 using API Gateway.

* 3. Progress Tracking

A screenshot of a computer

AI-generated content may be incorrect.

Updates student completion status in DynamoDB via a Lambda function.

# Technology Stack

• Frontend: HTML, CSS, JavaScript (TailwindCSS).

• Backend: Python 3.11 (AWS Lambda).

• Database: Amazon DynamoDB.

• Storage: Amazon S3 (Video Files).

• Content Delivery: AWS CloudFront.

• API Management: AWS API Gateway.

• Security: AWS IAM.

# Implementation

Below are short code excerpts from the implemented Lambda and frontend logic:

* **Example: Lambda Function for Enrollment**

import json  
import boto3  
import uuid  
  
dynamodb = boto3.resource('dynamodb')  
table = dynamodb.Table('StudentCourses')  
  
def lambda\_handler(event, context):  
 body = json.loads(event.get('body', '{}'))  
 studentId = str(uuid.uuid4())  
 item = {  
 'studentId': studentId,  
 'name': body.get('name', ''),  
 'email': body.get('email', ''),  
 'courseId': 'default'  
 }  
 table.put\_item(Item=item)  
 return {'statusCode': 200, 'body': json.dumps({'studentId': studentId})}

* **Example: Frontend API Call (index.html)**

const API\_BASE = 'https://example.execute-api.ap-south-1.amazonaws.com/Staging';  
  
async function enrollStudent() {  
 const body = {  
 name: document.getElementById('name').value,  
 email: document.getElementById('email').value  
 };  
  
 const res = await fetch(`${API\_BASE}/enroll`, {  
 method: 'POST',  
 headers: {'Content-Type':'application/json'},  
 body: JSON.stringify(body)  
 });  
  
 const data = await res.json();  
 localStorage.setItem('studentId', data.studentId);  
}

# Database Design (DynamoDB)

**Table Name:** StudentCourses  
**Primary Key:** studentId (String)  
**Attributes:** name, email, courseId, progressStatus  
**Purpose:** To maintain student enrollment and progress details.

# System Workflow

1. Student opens the web application hosted on S3.

2. Enters details and submits the enrollment form.

3. API Gateway triggers the Enrollment Lambda.

4. Student details are stored in DynamoDB.

5. Videos are loaded from S3 and streamed via CloudFront.

6. Upon completion, Lambda updates progress status in DynamoDB.

# Testing & Results

All Lambda functions executed successfully and stored data in DynamoDB. Videos streamed correctly from S3 via CloudFront distribution. End-to-end testing confirmed that the architecture functions as expected.

# Challenges & Solutions

* **CORS Configuration:** Resolved by enabling CORS headers in API Gateway and Lambda responses.
* **DynamoDB Schema Design:** Optimized to store both enrollment and progress data efficiently.
* **CloudFront Caching:** Configured cache invalidation for timely updates.
* **IAM Permissions:** Implemented least-privilege policies for Lambda and S3 access.

# Conclusion

This project demonstrates how AWS Serverless Architecture can be used to build a scalable, secure, and cost-efficient online learning platform. **Future improvements** include integrating user authentication via AWS Cognito and adding analytics for learning behavior tracking.

# References

**1. AWS General Documentation**  
🔗 https://docs.aws.amazon.com/

**2. AWS Lambda Developer Guide**  
🔗 https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

**3. Amazon API Gateway Developer Guide**  
🔗 https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

**4. Amazon DynamoDB Developer Guide**  
🔗 https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

**5. Amazon S3 Developer Guide**  
🔗 https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html

**6. AWS CloudFront Documentation**  
🔗 https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html

**7. AWS Identity and Access Management (IAM)**  
🔗 https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html