

# **LearnMate: An AI-Enhanced Learning Management System for Online Video Lectures**

Chua Qi Wang

## Table of Contents

Software Requirements and Specifications .....	3
Functional Requirements .....	3
Non-Functional Requirements .....	5
Software Design.....	6
System Design and Architecture .....	6
Microservices .....	8
System Workflow .....	9
Database Design.....	18
Implementation .....	21
AI Integration.....	22
Software Testing .....	24
Unit Testing and Integration Testing.....	24
End to End Testing.....	27
User Interface Walkthrough .....	28
Conclusion .....	32

## **LearnMate: An AI-Enhanced Learning Management System for Online Video Lectures**

This project aims to develop a learning management system with an AI teaching assistant for online video lectures. The core functionality of this application revolves around streaming lectures, complemented by a chatbot that efficiently addresses students' questions and clarifications in real-time.

The AI teaching assistant harnesses the power of Large Language Models (LLM) to enhance the learning process. When a student poses a question or seeks clarification during a lecture, the system seamlessly combines video context with the student's query. This contextual information is then fed into a LLM to generate more accurate and insightful responses.

This project aims to address common challenges faced by students, such as the need for immediate clarification during lectures and the availability of additional context to enhance comprehension. By integrating AI-driven capabilities into the educational experience, the AI Teaching Assistant not only provides valuable assistance but also empowers students to engage more actively with the material.

### **Software Requirements and Specifications**

#### **Functional Requirements**

##### **1. User Authentication and Authorization**

- Students and teachers must be able to register an account by providing necessary information (e.g. email, password).
- Verification of email addresses must be required to complete registration.
- Registered users (students and teachers) must be able to log in using their email and password.

- The system must differentiate between students and teachers, providing different access levels and functionalities based on roles.

## 2. Course Management (Teachers)

- Teachers must be able to create new courses by providing course details (e.g., course name, description).
- Teachers must be able to update course details.
- Teachers must be able to delete courses.
- Teachers must be able to manage student enrollments in their courses.

## 3. Lesson Management (Teachers)

- Teachers must be able to upload video lectures to their courses.
- Teachers must be able to update existing video lectures.
- Teachers must be able to delete video lectures.

## 4. Video Lecture Streaming (Students)

- Students must be able to watch video lectures for the courses they are enrolled in.
- Support for streaming video content.

## 5. AI Teaching Assistant

- Students must have access to a chatbot interface to ask questions in real-time during lectures.
- The AI teaching assistant must be able to answer student questions using a Large Language Model (LLM).
- The AI must consider the current video context to provide accurate and insightful responses.

## 6. Security

- All user data must be securely stored and transmitted.
- Proper access controls must be implemented to prevent unauthorized access to sensitive data.

## **Non-Functional Requirements**

### **1. Performance**

- The system must ensure that all requests are processed and responded to within 3 seconds under normal load conditions.
- The system must handle up to 30,000 concurrent users without performance degradation.

### **2. Reliability**

- The system must have an uptime of 99.9% or higher.
- The system must automatically recover from failures using AWS services such as AWS SQS with dead-letter queues and AWS Lambda retries.

### **3. Security**

- All user data must be encrypted in transit using HTTPS
- Implement robust RBAC to ensure that only authorized users can access specific functionalities and data.

### **4. Maintainability**

- The code must adhere to industry best practices and be well-documented.
- The system must be built using a microservices architecture to ensure that individual services can be developed, deployed, and maintained independently.
- Implement automated testing (unit, integration, and end-to-end tests) to ensure the reliability of code changes.

## 5. Usability

- The user interface must be intuitive and easy to navigate for both students and teachers.

## 6. Scalability

- The system must be able to scale horizontally by adding more instances of AWS Lambda functions and other services as needed.

## 7. Cost Efficiency

- The system must utilize AWS Cost Explorer and AWS Budgets to monitor and control cloud expenditure.
- Implement resource optimization strategies, such as using AWS Lambda for on-demand compute power and AWS S3 for cost-effective storage.

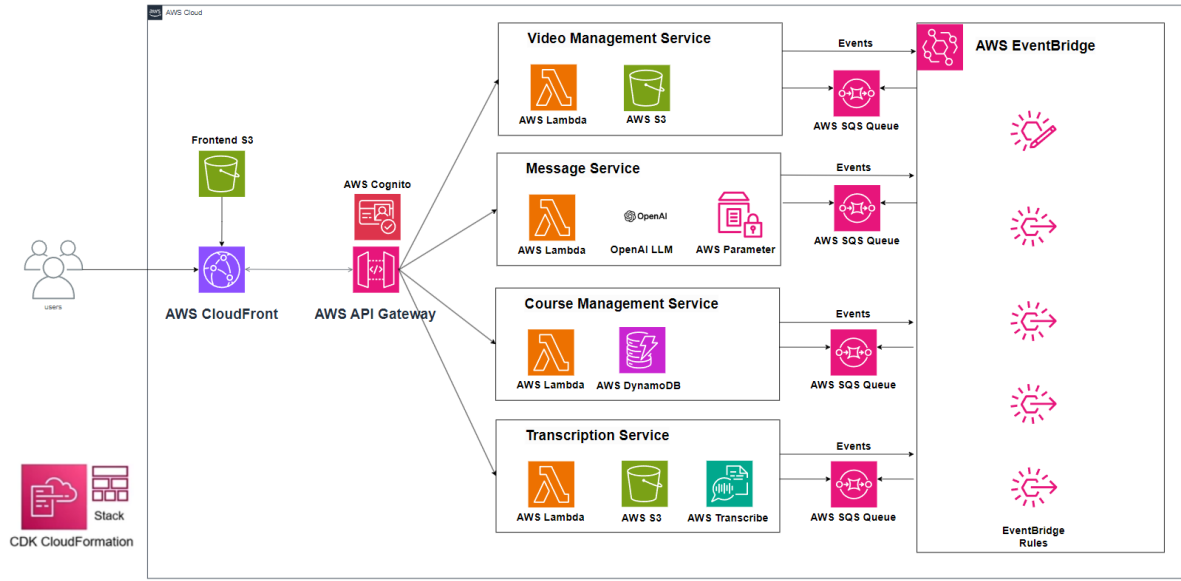
# Software Design

## System Design and Architecture

The project will embrace serverless microservices and event-driven architecture. The goal is to leverage the power of AWS to create a robust and scalable platform that adapts to the dynamic demands of modern web applications. The microservices approach ensures modular and independent components while gaining agility in development, deployment, and maintenance.

## Figure 1

*Overall architecture*



## Serverless patterns employed

- RESTful Microservices
- Fanout pattern
- Pub/Sub
- Topic-Queue Chaining with SQS
- Circuit Breaker

## AWS services used

- AWS CloudFront for distribution of static content i.e. frontend and video/images
- AWS Cognito for user authentication, authorization, and user management
- AWS API gateway for creating and managing APIs
- All microservices will be powered by lambda functions with its main choice of database as DynamoDB

- AWS S3 for storing static content like videos, frontend objects (HTML, Javascript, images)
- AWS EventBridge to decouple the microservices in the deployment
- AWS SQS with EventBridge to ensure overall system reliability and fault tolerance
- AWS CDK for managing infrastructure as code provisioning, enabling the definition and management of cloud infrastructure resources using TypeScript. It provides a higher level of abstraction to create reusable constructs and patterns thus improving productivity and maintainability

## **Microservices**

The system architecture follows a microservices-based approach, comprising four distinct and independently deployable services, each responsible for specific functionalities and communicating through well-defined interfaces.

### **Video Management Service**

The Video Management Service is responsible for handling all video-related functionalities within the learning management system. This service manages the uploading, updating and deletion of video lectures.

### **Message Service**

The Message Service is responsible for facilitating communication between students and the AI teaching assistant. This service handles the real-time interaction of students with the chatbot and processes their queries to the LLM.

### **Course Management Service**



The Course Management Service is responsible for managing course-related functionalities. This service enables teachers to create, update, and delete courses / lessons, as well as manage student enrollments.

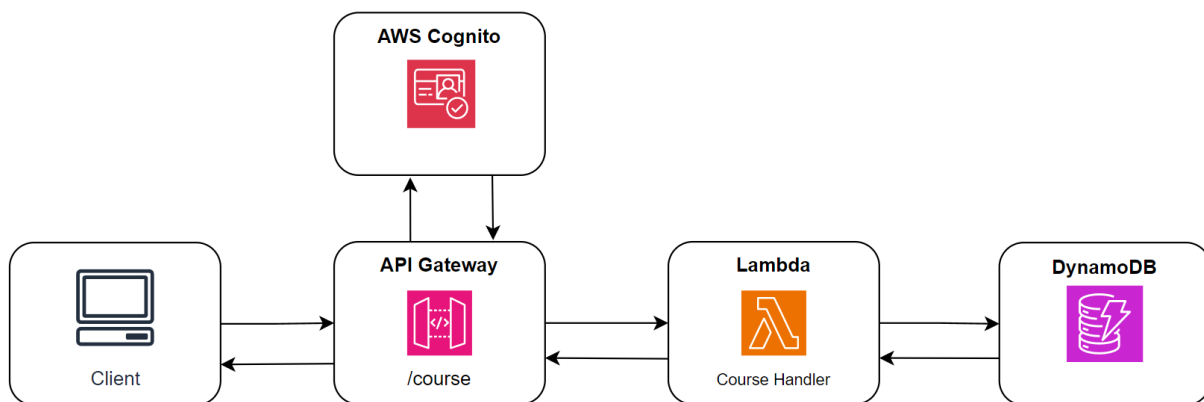
## Transcription Service

The Transcription Service is responsible for generating and managing transcriptions of video lectures. This service processes video files to create accurate text transcriptions, which will be used for providing better context to the student's queries.

## System Workflow

This section describes the detailed workflows and interactions between various components of the learning management system.

### Course / Lesson (Create, Read, Update)



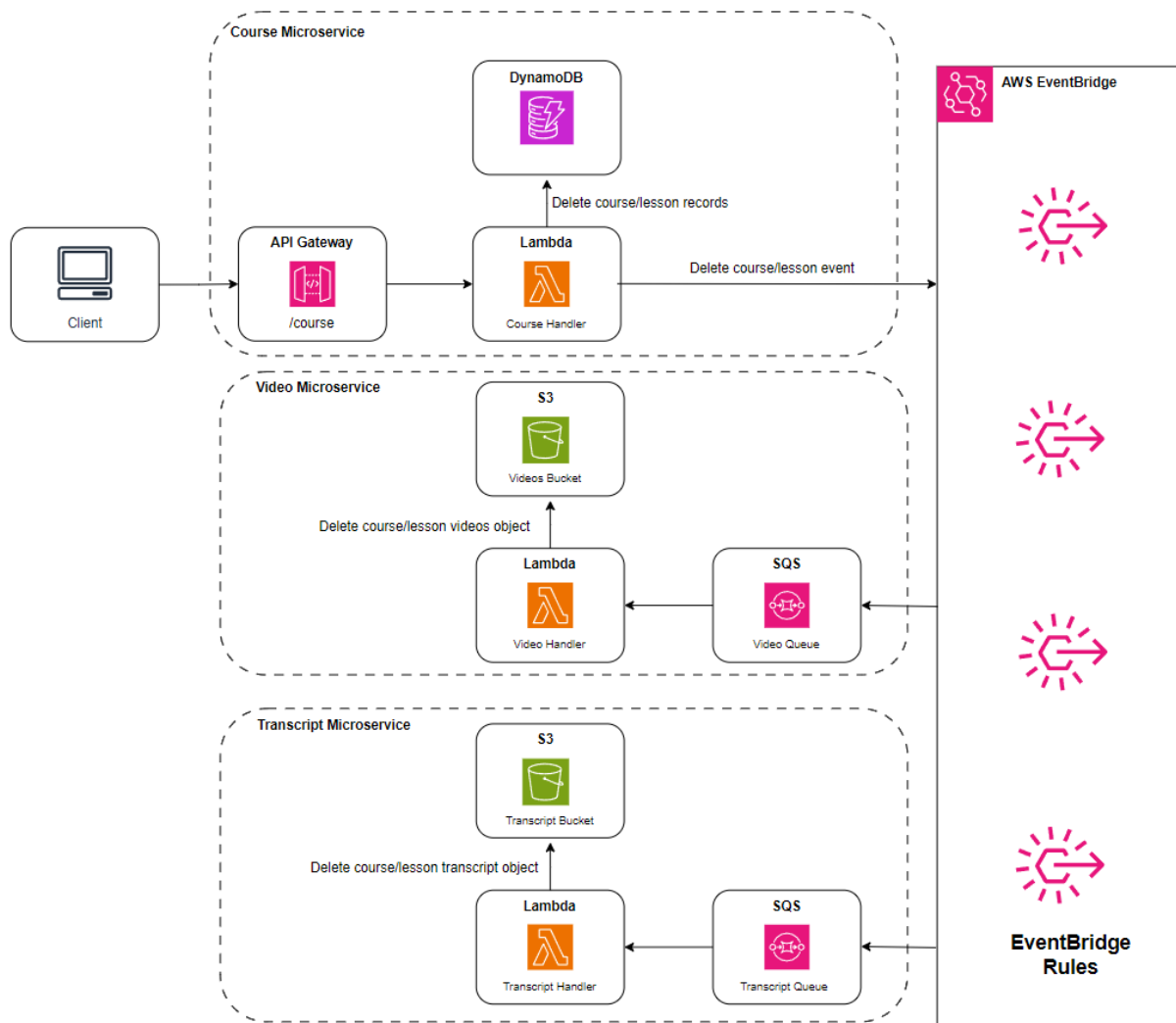
1. The client initiates a request to the /course API endpoint hosted on AWS API Gateway. The request includes an authentication token obtained from AWS Cognito.
2. API Gateway receives the incoming request and first validates the authentication token with Cognito. This step ensures that only authenticated and authorized users can access

the API. If the token is invalid or expired, API Gateway will reject the request immediately.

3. Assuming the authentication is successful, API Gateway then routes the request to the Course Handler AWS Lambda function based on the API path and HTTP method. The Lambda function serves as the backend logic for the API.
4. The Lambda function is invoked with the details of the request, including any path parameters, query string parameters, and the request body. It processes the request according to the business logic.
5. If the Lambda function needs to interact with the database, it makes calls to DynamoDB using AWS SDK. This could involve reading data, writing new records, or updating existing ones.
6. After processing the request and interacting with DynamoDB as needed, the Lambda function prepares a response. This might include fetched data, confirmation of an action, or any other relevant information.
7. The Lambda function returns this response back to API Gateway. API Gateway then sends it back to the client.

Throughout this process, AWS services handle scaling automatically. API Gateway can handle large numbers of concurrent API calls, Lambda functions scale instantly to meet demand, and DynamoDB automatically adjusts its capacity units based on traffic.

### **Course / Lesson Deletion**



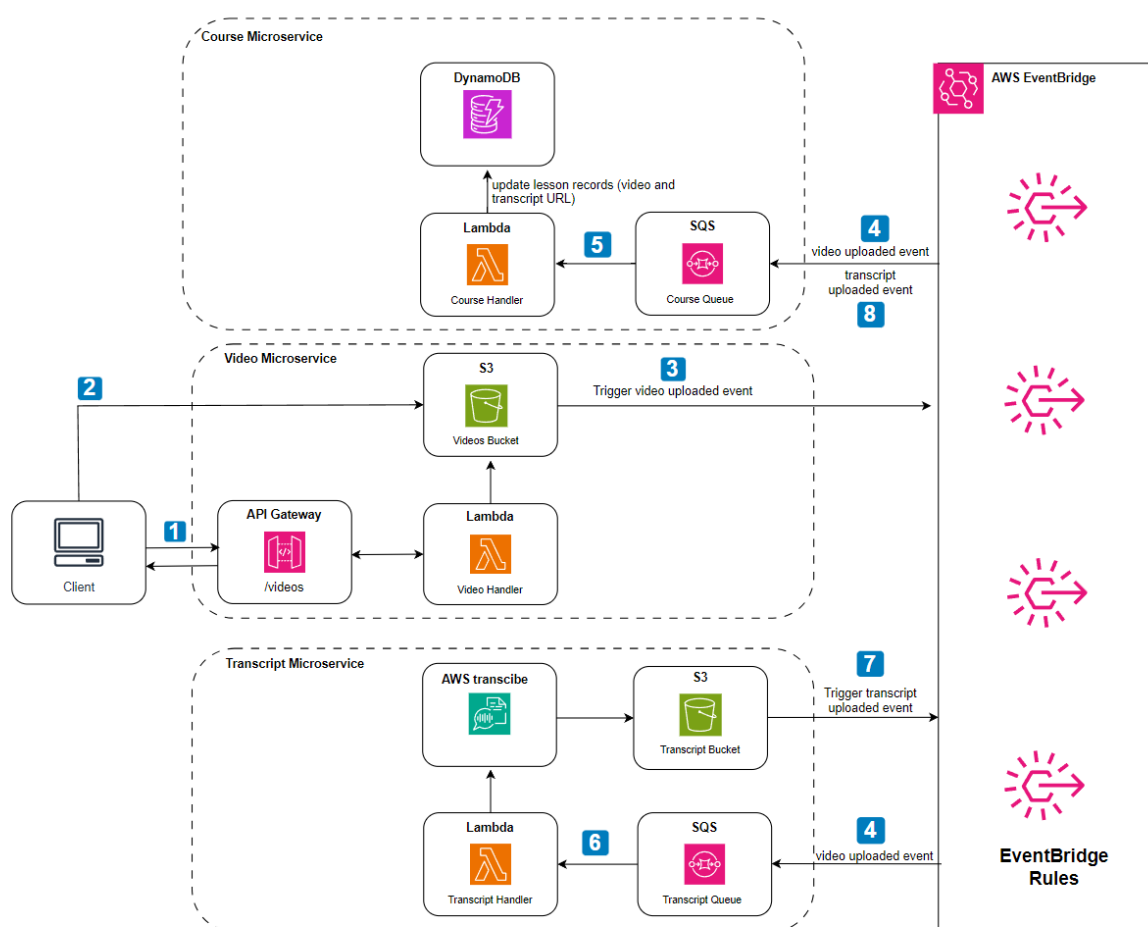
1. The process begins when a client makes a DELETE request to the /course endpoint on API Gateway. This request is intended to delete either a course or a lesson, depending on the specific parameters provided.
2. API Gateway receives this request and routes it to the course handler Lambda function. This Lambda function is responsible for handling the deletion logic for courses or lessons.

3. The Lambda function first interacts with DynamoDB to delete the corresponding records for the course or lesson. This ensures that the data is removed from the database, maintaining consistency in the application's data layer.
4. After successfully deleting the records from DynamoDB, the Lambda function emits a "DeleteCourse" or "DeleteLesson" event to AWS EventBridge. This event contains relevant information about the deleted item, such as its courseId and lessonId.
5. EventBridge receives this event and based on a predefined rule (deleteCourseRule / deleteLessonRule), routes it to two separate SQS queues: the Video Queue and the Transcript Queue. This fan-out pattern allows for parallel processing of related tasks.
6. The Video Queue receives a message about the deletion event. This queue is configured to trigger a Video Handler Lambda function. When the message arrives in the queue, it automatically invokes this Lambda function.
7. The Video Handler Lambda function processes the message from the Video Queue. Its task is to delete the corresponding video file from the Videos S3 bucket. It uses the information provided in the event message to locate and remove the correct video object.
8. Simultaneously, the Transcript Queue also receives a message about the deletion event. Similar to the Video Queue, this queue is set up to trigger a Transcript Handler Lambda function when a message arrives.
9. The Transcript Handler Lambda function is invoked by the message in the Transcript Queue. This function is responsible for deleting the associated transcript file from the Transcripts S3 bucket. It uses the event information to identify and remove the correct transcript object.

Both the Video Handler and Transcript Handler Lambda functions operate independently and in parallel, optimizing the deletion process. They each focus on cleaning up their respective S3 buckets, ensuring that all associated files are removed along with the course or lesson data.

This architecture demonstrates an event-driven, serverless approach to managing complex deletion processes. It leverages various AWS services to create a scalable, efficient system that maintains data consistency across multiple storage locations. The use of SQS queues also provides a buffer that can handle high volumes of deletion requests without overwhelming the system.

## Video Upload

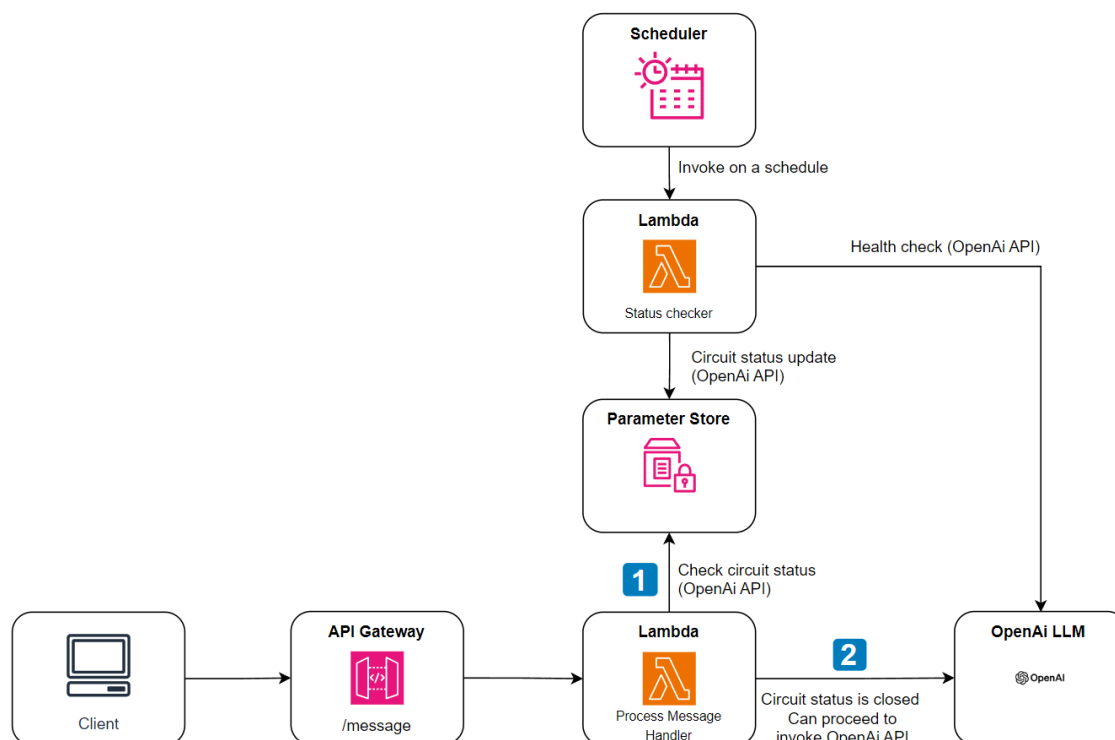


1. The process begins when a client makes a GET request to the /videos/upload-url endpoint on API Gateway. This request is made to obtain a pre-signed URL for uploading a video. API Gateway routes this request to the Video Handler Lambda function. This Lambda function is responsible for generating a pre-signed URL for the Videos S3 bucket. The Video Handler Lambda generates a pre-signed URL for the Videos S3 bucket. This URL grants temporary permission to upload a file to a specific location in the bucket, ensuring secure and direct upload from the client. The Lambda function then sends the pre-signed URL back to API Gateway, which in turn sends it back to the client. This completes the first part of the process, providing the client with a secure means to upload the video.
2. Using the pre-signed URL, the client uploads the lesson video directly to the Videos S3 bucket. This direct upload to S3 is efficient as it bypasses any server, reducing load and allowing for larger file uploads.
3. Upon successful upload of the video to the S3 bucket, an event is automatically generated. This event is picked up by EventBridge based on the rule defined, which looks for "Object Created" events in the specified Videos S3 bucket.
4. EventBridge processes this event and based on the rule, routes it to two SQS queues: the Course Queue and the Transcript Queue. This fan-out pattern allows for parallel processing of related tasks.
5. For the Course Queue, the message triggers the Course Handler Lambda function. This function updates the corresponding lesson record in DynamoDB with the new video URL, ensuring that the database reflects the newly uploaded video.

6. Simultaneously, the Transcript Queue triggers the Transcript Handler Lambda function. This function initiates a transcription job using AWS Transcribe, starting the process of generating a transcript for the uploaded video.
7. When AWS Transcribe completes the transcription job, it stores the resulting transcript in the Transcripts S3 bucket. This storage event triggers another EventBridge rule that is defined, which looks for JSON files created in the Transcripts S3 bucket.
8. This new EventBridge rule routes the transcript creation event to the Course Queue. The Course Handler Lambda function is triggered again, this time updating the lesson record in DynamoDB with the URL of the newly created transcript.

This architecture demonstrates a sophisticated, event-driven approach to handling video uploads and related processes. It leverages various AWS services to create a scalable, efficient system that maintains consistency across multiple data stores and automatically generates transcripts. The use of pre-signed URLs for uploads and EventBridge for event routing allows for a decoupled, highly responsive system that can handle complex workflows triggered by file uploads.

## **Message**



The system begins with a scheduled task set up in AWS EventBridge Scheduler. This task is configured to run every 15 minutes, invoking a Status checker Lambda function. This regular health check forms the basis of the circuit breaker pattern.

When invoked, the Status Checker Lambda function performs a health check on the OpenAI API. This involves making a simple request to the API ([status.openai.com](https://status.openai.com)) and evaluating the response. The function then updates a parameter in AWS Systems Manager Parameter Store with the status of the OpenAI API, which could be "operational" or "outage".

On the client side, when a user wants to generate a message, they send a request to the /message endpoint on AWS API Gateway. This request includes the content for which a message needs to be generated.



API Gateway receives this request and routes it to the Message Handler Lambda function. This function is responsible for processing the message generation request and implementing the circuit breaker logic.

Upon invocation, the Message Handler Lambda first retrieves the status of the OpenAI API from the Parameter Store. This status, updated regularly by the Status Checker Lambda, determines whether the circuit is closed (operational) or open (outage).

If the status is "operational", indicating that the circuit is closed, the Message Handler Lambda proceeds to invoke the OpenAI API. It sends the user's content to the API's Large Language Model (LLM) to generate the requested message.

Once the OpenAI API returns the generated message, the Message Handler Lambda receives it and prepares the response. This response is then sent back through API Gateway to the client, completing the successful message generation process.

However, if the status retrieved from the Parameter Store is not "operational", the circuit is considered open. In this case, the Message Handler Lambda does not attempt to call the OpenAI API. Instead, it returns an error message to the client, explaining that the service is temporarily unavailable.

This implementation of the circuit breaker pattern helps prevent cascading failures that could occur if the system continually tried to call an unavailable or failing API. It also allows for quick recovery once the OpenAI API is operational again, as the Status Checker will update the status accordingly.

By using AWS services like EventBridge Scheduler for regular health checks, Parameter Store for shared state, and Lambda for serverless compute, this architecture creates a robust,

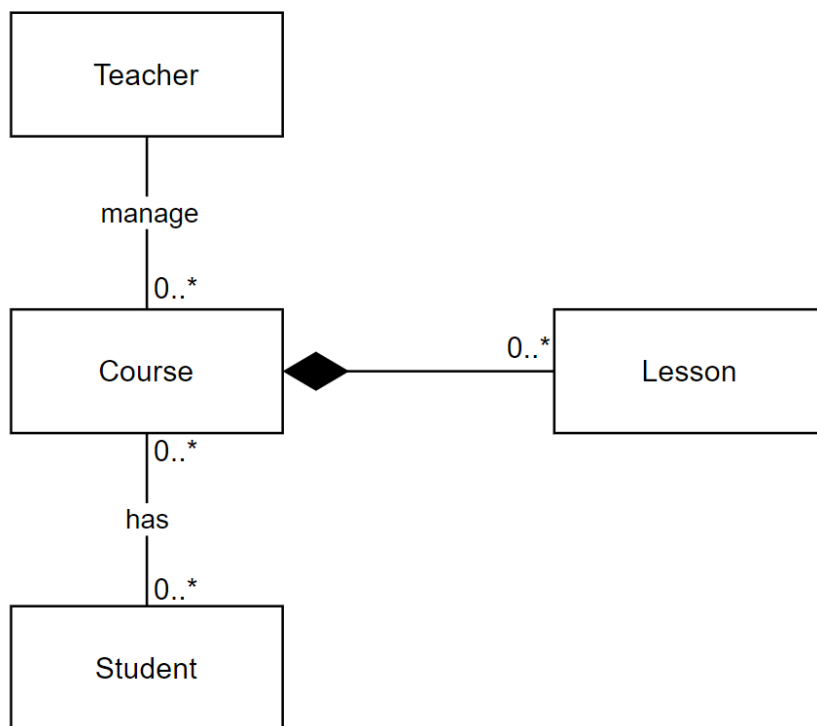
scalable system. It efficiently manages dependencies on external services while providing a responsive experience to the end users.

## Database Design

The database design for this learning management system is centered around Amazon DynamoDB, a NoSQL database service that provides fast and predictable performance with seamless scalability. This section outlines the key aspects of the database design, including the use of a single table design and schema considerations.

**Figure 2**

*UML diagram for course management*



In the context of database design, especially when working with a NoSQL database like Amazon DynamoDB, it is crucial to model the relationships and interactions between various

entities before implementing a single table design. One effective way to achieve this is by using Unified Modeling Language (UML) diagrams.

### 1. Database Selection

DynamoDB was chosen for its high availability, automatic scaling, and low-latency performance, which are essential for the real-time and high-traffic requirements of the learning management system.

### 2. Single Table Design

The single table design approach in DynamoDB involves using a single table to store multiple types of entities. This approach leverages DynamoDB's efficient querying capabilities and simplifies the management of relationships between different entities.

#### Advantages

- **Simplified Schema Management:** Reduces the complexity of managing multiple tables.
- **Efficient Queries:** Optimizes read and write operations by minimizing the number of queries needed.
- **Scalability:** Enhances performance and scalability by reducing the number of database partitions and maximizing throughput.

### 3. Schema Design

#### a. Primary Key Structure

- i. **Partition Key (PK):** A composite key combining PK (Partition Key) and SK (Sort Key).
- ii. **Sort Key (SK):** Allows for efficient querying of related items.

#### b. Entity Types

- Courses
  - `c#<course_id>`
- Lessons
  - `l#<lesson_id>`
- Students
  - `s#<student_id>`
- Teacher
  - `t#<teacher_id>`

#### c. Query Patterns

- Get course information
  - Query: PK = `'c#<course_id>'` and SK = `'c#<course_id>'`
- Get lesson information
  - Query: PK = `'c#<course_id>'` and SK = `'l#<lesson_id>'`
- Get students enrolled in a course
  - Query: PK = `'c#<course_id>'` and SK BEGINS WITH `'s#'`
- Get courses managed by a teacher
  - Query: PK = `'t#<teacher_id>'` and SK BEGINS WITH `'c#'`
- Get lessons from a course

- Query: PK = 'c#<course\_id>' and SK BEGINS WITH 'l#'
- Get a student's courses
  - Query: PK = 's#<student\_id>' and SK BEGINS WITH 'c#'

## Implementation

The implementation phase of the LearnMate project involved several key steps, including setting up the development environment, building the front-end and back-end components and deploying the application on AWS. This section provides a detailed account of the technologies used.

### 1. Development Environment

- Git: For version control.
- Docker: For containerizing services during development.

### 2. Front-End Development

- React.js: The core framework for building the user interface, providing a component-based architecture that ensures reusability and maintainability.
- TanStack Query: Used for data fetching, caching and synchronization, thus ensuring efficient and seamless data handling.
- TailwindCSS: A utility-first CSS framework for rapidly building custom designs directly within the HTML structure, ensuring a clean and responsive design.
- shadcn: A library of pre-designed, customizable UI components that integrate well with TailwindCSS, enhancing the UI with a cohesive and modern look.

- AWS Amplify: Simplifies the integration of AWS services with the front end, handling authentication.
- Vite: A build tool that provides fast development server start times and optimized production builds, improving the overall development experience.
- TypeScript: A superset of JavaScript that adds static types, ensuring type safety and reducing the potential for runtime errors.

### 3. Back-End Development

- AWS SDK: Used for interacting with various AWS services programmatically, enabling seamless integration with the back-end components.
- Node.js: The runtime environment for executing JavaScript code on the server side, providing event-driven architecture and non-blocking I/O, which is ideal for building scalable applications.
- JavaScript: The primary programming language used for writing the back-end logic, ensuring compatibility and ease of development.

### 4. Deployment

- AWS CDK (Cloud Development Kit): Employed to define and provision AWS resources using a high-level abstraction, making the infrastructure more manageable and reusable.
- Typescript: Utilized for defining and managing infrastructure as code, providing type safety and better tooling support.

## **AI Integration**

When a user makes a prompt in LearnMate, the system injects context into the prompt to enhance the accuracy and relevance of the AI's response. This context includes the course title,

lesson title, lesson description, and the transcript of what was said in the lecture around the time the user paused the video. This is achieved using a sliding window approach: if a user pauses at the 02:34 mark, the system captures and includes what was said one minute before and one minute after the pause (01:34 to 03:34), providing a comprehensive snapshot of the immediate context.

By default, LearnMate utilizes GPT-3.5 for generating responses. This choice is driven by the fast response times. GPT-3.5 is optimized for speed, ensuring that students receive prompt responses to their queries, which is crucial for maintaining engagement during a lesson.

In addition to this, LearnMate offers an advanced feature where users can choose to send the current video frame along with their prompt. This feature leverages GPT-4o with vision capabilities, which provides a more nuanced understanding by analyzing visual content from the lecture. While GPT-3.5 excels in textual analysis, GPT-4V can interpret visual data, enabling it to answer questions that involve diagrams, charts, or other visual aids present in the lecture frame.

This is the system prompt used to guide the AI in providing high-quality responses:

You are a professor that will be teaching a university student.

The course title is \${courseTitle}.

The lesson title is \${lessonTitle}.

The lesson description is \${lessonDescription}.

The student is currently watching the lesson video online and will ask questions regarding the lesson.

Answer the following question in a smart, highly accurate way, and think about your answers carefully before responding.

Break down the question into smaller, easier-to-understand parts. Use analogies and real-life examples to simplify the concept and make it more relatable.

You will be given

1) a transcript of a portion of the lesson for context

Use the information provided to gain a better context of the question and answer the student's question.

This prompt ensures that the AI acts as a knowledgeable professor, capable of providing detailed and sound explanations. By incorporating both textual and, when applicable, visual context, LearnMate's AI teaching assistant can enhance the learning experience, making it more interactive and effective for students.

## **Software Testing**

Testing was a crucial part of ensuring the reliability of the application. This section details the approach taken for unit testing, integration testing, and end-to-end testing using Jest and Playwright.

### **Unit Testing and Integration Testing**

Jest was utilized for backend testing, focusing on verifying the correctness of configurations and ensuring that integrations were set up correctly within the AWS CDK infrastructure.

### **Configurations**

Jest was used to verify the correctness of configurations, such as DynamoDB schema definitions and Lambda runtime environments. This ensured that database structures were properly defined and that Lambda functions executed within expected runtime environments, optimizing performance and data consistency.

```
test("Lambda properties", () => {
  backendStackTemplate.hasResourceProperties("AWS::Lambda::Function", {
    Handler: "index.handler",
    Runtime: "nodejs16.x",
```



```
});
});
```

```
test('DynamoDB Table "course" has PK and SK attributes', () => {
  backendStackTemplate.hasResourceProperties("AWS::DynamoDB::Table", {
    TableName: "course",
    AttributeDefinitions: [
      { AttributeName: "PK", AttributeType: "S" },
      { AttributeName: "SK", AttributeType: "S" },
    ],
  });
});
```

## IAM Permissions and Roles

Testing IAM permissions and roles was crucial to validate secure access controls within the application. Jest tests were designed to verify that appropriate permissions were assigned to AWS services, ensuring that only authorized actions could be performed, thus enhancing data security and compliance. In the code snippet below, it's a test written to check if the transcribe lambda function has the policy to call aws transcribe to start a transcription job.

```
test("MicroservicestranscribeLambdaFunction policy allows start
transcription job", () => {
  backendStackTemplate.hasResourceProperties("AWS::IAM::Policy", {
    PolicyDocument: Match.objectLike({
      Statement: Match.arrayWith([
        Match.objectLike({
          Action: "transcribe:StartTranscriptionJob",
          Effect: "Allow",
          Resource: "*",
        }),
      ]),
    }),
    Roles: Match.arrayWith([
      {
        Ref: Match.stringLikeRegexp(
          "MicroservicestranscribeLambdaFunctionServiceRole.*"
        ),
      },
    ]),
  });
});
```

```
});
});
```

## Integration Points

Jest tests were instrumental in validating seamless integration between different AWS services utilized within the application. This included testing interactions between Lambda functions, EventBridge, and other services provisioned via AWS CDK. These tests ensured that service dependencies and data flows were correctly configured, guaranteeing reliable communication and functionality across the application ecosystem. Below is a code snippet testing if the EventBridge Rule targets the correct services.

```
test("EventBridge rule for delete course has correct properties", () => {
  backendStackTemplate.hasResourceProperties("AWS::Events::Rule", {
    Description: "When course microservice deletes a course",
    EventBusName: {
      Ref: Match.stringLikeRegexp("EventBusLmEventBus.*"),
    },
    EventPattern: {
      source: ["com.lm.course.deletecourse"],
      "detail-type": ["DeleteCourse"],
    },
    Name: "DeleteCourseRule",
    State: "ENABLED",
    Targets: Match.arrayWith([
      Match.objectLike({
        Arn: Match.objectLike({
          "Fn::GetAtt": Match.arrayWith([
            Match.stringLikeRegexp("QueueVideosQueue.*"),
            "Arn",
          ]),
        }),
        Id: "Target0",
      }),
      Match.objectLike({
        Arn: Match.objectLike({
          "Fn::GetAtt": Match.arrayWith([
            Match.stringLikeRegexp("QueueTranscribeQueue.*"),
            "Arn",
          ]),
        }),
      }),
    ]),
  }),
});
```

```
        Id: "Target1",  
      })),  
    ]),  
  }));  
});
```

## End to End Testing

End-to-end (E2E) testing was performed using Playwright, a modern testing framework for web applications that allows automation of browser interactions. This type of testing simulated real user scenarios to verify critical paths and user interactions within the application.

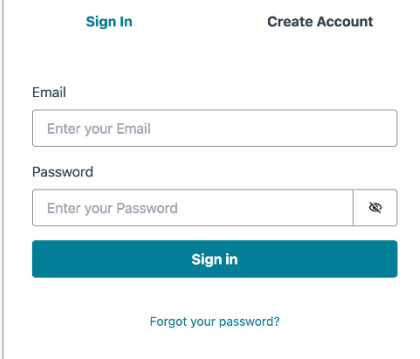
### Critical Paths Tested:

- Sending a Message to the Chatbot: Validated real-time interaction with the AI teaching assistant, confirming accurate responses to student queries.
- Creating a Course: Tested the creation of new courses by teachers, ensuring proper setup and availability for student enrollment.
- Creating a Lesson: Verified the addition of new lessons within courses
- Adding/Removing a Student to/from a Course: Ensured smooth management of student enrollments, validating access permissions and updates.
- Deleting a Lesson: Tested the removal of lessons from courses, ensuring data integrity and proper cleanup.
- Deleting a Course: Validated the deletion of courses, including associated lessons and student enrollments, with appropriate permissions and data handling.
- Signing In: Verified user authentication and authorization processes.
- Signing Out: Tested logout functionality, confirming session management and user state transitions.

## User Interface Walkthrough

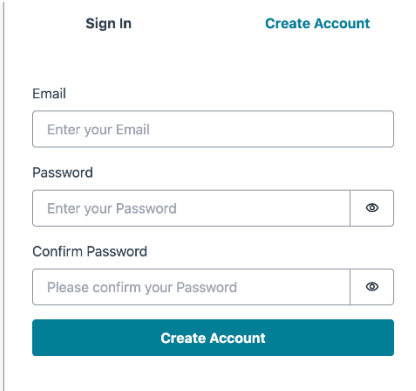
This section provides a visual tour of LearnMate's key features and functionalities, demonstrating the user experience for both students and teachers.

### Signing In




The Sign In form is located at the top of the page. It features two tabs: "Sign In" (active) and "Create Account". Below the tabs are two input fields: "Email" with the placeholder "Enter your Email" and "Password" with the placeholder "Enter your Password". The Password field has a toggle icon on the right. Below the input fields is a large teal "Sign in" button. At the bottom, there is a link "Forgot your password?".

### Creating an account



The Create Account form is located at the top of the page. It features two tabs: "Sign In" and "Create Account" (active). Below the tabs are three input fields: "Email" with the placeholder "Enter your Email", "Password" with the placeholder "Enter your Password", and "Confirm Password" with the placeholder "Please confirm your Password". Each input field has a toggle icon on the right. Below the input fields is a large teal "Create Account" button.

## Courses Overview




III Courses ▾


➔ Sign Out

Manage Courses


Courses



TIC4002 Intro to Machine Learning




TIC4004 Software Quality Control



TIC3920 Data Structures and Algorithms

## Lessons Overview




III Courses ▾

➔ Sign Out

➔ Exit

TIC4004 Software Quality Control



Lecture 1

- about 1 month ago

## Lesson Video With Chatbot Page

**Types of white-box testing**

```

graph TD
    A[White box Testing] --> B[Control flow testing]
    A --> C[Data flow testing]
  
```

14:10 / 18:09

what is the difference?

11:52

In the context of software testing, the user is likely asking about the difference between white box testing, control flow testing, and data flow testing. Let's break it down:

**White Box Testing vs. Control Flow Testing vs. Data Flow Testing:**

**White Box Testing:** White box testing is a comprehensive testing approach where the internal structure, design, and implementation of the software are examined. It involves looking at the code paths, branches, and various other aspects to ensure thorough testing.

**Control Flow Testing:** Control flow testing is a type of white box testing that specifically focuses on identifying the execution path through a module or program code. It helps in creating and executing test cases based on the

**Lecture 1**

In this lesson, you will delve into three crucial concepts in software testing and quality assurance: Control Flow Graphs (CFG), White Box Testing, and Control Flow Testing. These techniques are essential for understanding the internal workings of a software program and ensuring it functions correctly under various conditions.

Enter your question...

☐ Use current video frame for question (Response will take longer)

## Courses Overview (Teacher View)

LearnMate

Courses

Sign Out

Exit

New Course

TBA2102 Introduction to Business Analytics [2320]

TIC4002 Intro to Machine Learning

TIC4004 Software Quality Control

TIC3920 Data Structures and Algorithms

TIC4003 Software Project Management [2320]

## Creating a Course (Teacher View)

[→ Exit

Complete all fields (2/2)



## Lessons

 Edit title

TBA2102 Introduction to Business Analytics [2320]

 [Edit description](#)

update

## Student Management

## Students

 Add a Student

No students

 Add a Lesson

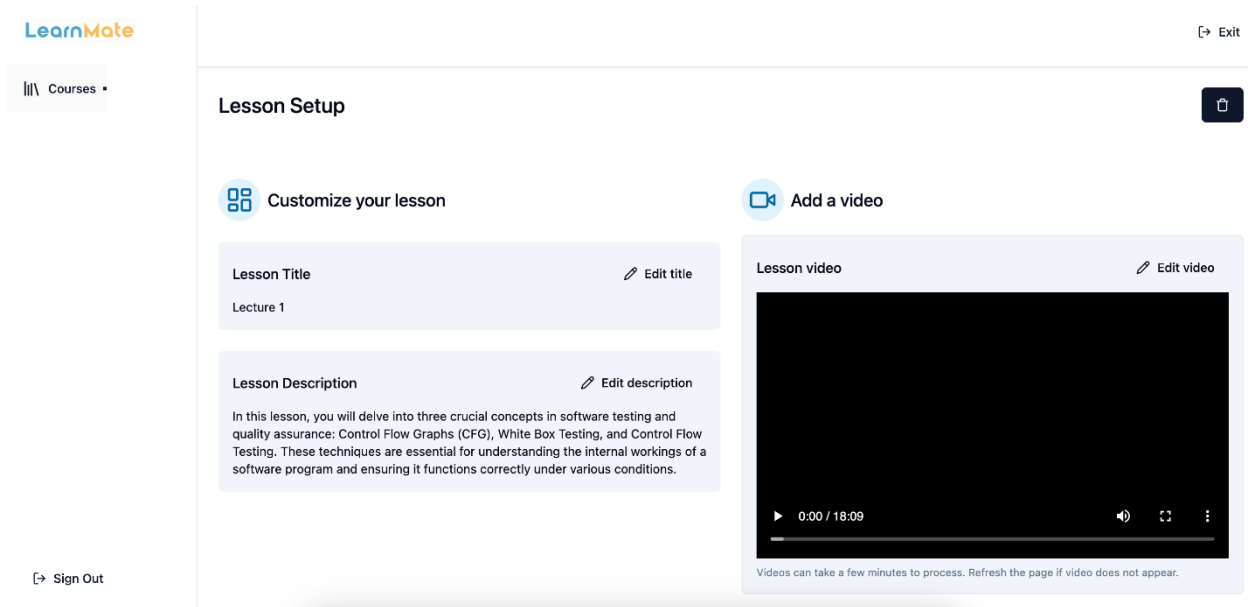
Lesson 1

Lesson 2

Lesson 3

lesson 33

12222



## Conclusion

The LearnMate project has successfully developed an AI-enhanced Learning Management System for online video lectures, addressing common challenges faced by students in online learning environments. By leveraging cutting-edge technologies and a well-designed architecture, LearnMate offers a robust, scalable, and user-friendly platform that enhances the educational experience.

While the current implementation of LearnMate meets its initial objectives, there are opportunities for future enhancements. These could include expanding AI capabilities, better context retrieval, implementing more advanced analytics for learning progress tracking, and integrating with other educational tools and platforms.

In conclusion, LearnMate demonstrates the potential of AI-driven technologies in enhancing education by making online learning a more engaging and effective experience.