**Online Academy Design Document**

**Team Members:**

Srilekha Soma 700762977

Adudodla Sindhu 700756302

Bindu Sree Attipatla 700760695

Manas Abhinay Gajula 700757201

K. Shiva Sai Prasad 700763464

Table of Contents

[Introduction 5](#_Toc164775901)

[Purpose 5](#_Toc164775902)

[Scope 5](#_Toc164775903)

[Audience 5](#_Toc164775904)

[Document Overview 6](#_Toc164775905)

[System Overview 6](#_Toc164775906)

[Functional Description 6](#_Toc164775907)

[System Interactions 7](#_Toc164775908)

[Infrastructure 8](#_Toc164775909)

[Design Considerations 8](#_Toc164775910)

[Assumptions and Dependencies: 8](#_Toc164775911)

[General Constraints: 9](#_Toc164775912)

[Goals and Guidelines: 9](#_Toc164775913)

[Development Methods: 9](#_Toc164775914)

[Architectural Strategies 10](#_Toc164775915)

[Strategy 1: Microservices Architecture 10](#_Toc164775916)

[Strategy 2: Managed Database Services 11](#_Toc164775917)

[Strategy 3: Security and Compliance 11](#_Toc164775918)

[Strategy 4: Scalable Storage Solutions 11](#_Toc164775919)

[System Architecture 12](#_Toc164775920)

[Core Components 12](#_Toc164775921)

[System Interactions 13](#_Toc164775922)

[Detailed System Design 14](#_Toc164775923)

[Component 2: Course Progress 15](#_Toc164775924)

[ER Diagram 16](#_Toc164775925)

[Use Cases Diagrams 17](#_Toc164775926)

[Glossary 19](#_Toc164775927)

[Appendix 22](#_Toc164775928)

**List of Figures**

[Figure 1: Architecture Diagram 13](#_Toc164777304)

[Figure 2: ER Diagram 16](#_Toc164777305)

[Figure 3: User Authentication 16](#_Toc164777306)

[Figure 4: Course Creation 17](#_Toc164777307)

[Figure 5: Course Enrollment 17](#_Toc164777308)

[Figure 6: Dashboard 21](#_Toc164777309)

[Figure 7: Browse page 21](#_Toc164777310)

[Figure 8: Course page 22](#_Toc164777311)

[Figure 9: Teacher mode Courses page 22](#_Toc164777312)

[Figure 10: New Course Creation Page 23](#_Toc164777313)

[Figure 11: New Course Setup page 23](#_Toc164777314)

[Figure 12: Chapter Creation Page 24](#_Toc164777315)

[Figure 13: Analytics Page 24](#_Toc164777316)

[Figure 14: Stripe payments page 25](#_Toc164777317)

# 

# Introduction

This design document provides a detailed overview of the structure and features of the "Online Academy: A Comprehensive e-learning platform". This platform is designed to be a strong, adaptable, and easy-to-use online space that enables a wide variety of learning opportunities. The objective is to establish a connection between students and teachers in an interactive online environment, where knowledge can be shared and acquired using advanced technological tools.

## Purpose

This document serves as a detailed plan for the creation, implementation, and upkeep of the Online Academy platform. It functions as a reference for developers, engineers, and stakeholders to comprehend the system's features, architectural structure, and the technologies incorporated into the platform. This document guarantees that all parties involved possess a distinct vision and uniform comprehension of the project objectives and specifications.

## Scope

The Online Academy platform will provide a wide range of features specifically designed to improve the online learning experience for both educators and learners. The platform will offer teachers the ability to create, manage, and monetize courses. Users will have the capability to incorporate multimedia content, oversee chapters, and monitor their earnings and student involvement using extensive analytics.

The platform will provide students with a comprehensive learning environment, allowing them to enroll in courses, monitor their progress, and access a wide range of educational materials customized to their specific needs. The system will provide support for a wide range of course categories, encompassing both free and paid options. Additionally, it will incorporate a secure payment integration feature to facilitate course enrollment.

## Audience

This document is designed for:

* **Project Managers:** In order to furnish them with a comprehensive summary and regular assessment points, guaranteeing that the project adheres to the predetermined schedules and objectives.
* **Development Team:** In order to provide a comprehensive comprehension of the technical prerequisites, system structure, and features that necessitate development.
* **Design Team**: The objective is to synchronize the user interface design with the functional requirements and user experience objectives.
* **Quality Assurance Team:** In order to comprehend the system's functionality and meet the required performance metrics.
* **Stakeholders:** The purpose is to offer a comprehensive understanding of the project's extent, technological framework, and commercial goals, guaranteeing that their expectations are fulfilled.

## Document Overview

The remaining portion of this document is structured in the following manner:

* **System Overview:** Provides a comprehensive description of the overall system functionality and the ways in which different system components interact with each other.
* **Design Considerations:** Explores the limitations, presumptions, objectives, and principles that impact the design choices.
* **Architectural Strategies:** Describes the overarching strategies that direct the technical implementation of the platform.
* **System Architecture:** Offers an intricate explanation of the structure of the system, encompassing the movement of data, interactions between components, and the establishment of infrastructure.
* **Detailed System Design:** Analyzes the constituent elements and modules, providing a thorough explanation of the precise implementations and interactions.
* **Glossary:** Provides precise definitions of specialized terminology and concepts utilized within the document.
* **Appendix:** includes additional materials, such as diagrams and charts, that visually represent the architecture and design of the system.

This document is intended to be a dynamic document that will be revised as the project progresses and additional requirements are discovered.

# System Overview

The Online Academy platform is a comprehensive e-learning system that aims to offer a smooth and interactive educational experience for teachers and students. This platform combines diverse features focused on the creation, administration, and utilization of courses within a user-friendly digital setting. The system utilizes contemporary web technologies and cloud services to guarantee scalability, dependability, and accessibility.

## Functional Description

* **Teachers:** Educators possess the ability to generate and oversee their instructional programs. Users have the ability to designate courses as either free or paid, upload course images, generate chapters, attach supplementary materials, and incorporate multimedia content using video hosting platforms. Teachers are also able to utilize analytics in order to monitor revenue and engagement metrics. Course management is made easier by using a specialized dashboard (see Appendix, Figure 9: Teacher mode Courses page and Figure 10: New Course Creation page).
* **Students:** Students have the ability to peruse and register for courses, monitor their progress in learning, and gain access to a wide range of educational resources. The student interface encompasses functionalities for accessing and observing both available and enrolled courses, facilitating categorized browsing, and integrating payment systems for courses that require payment. The student experience is intentionally crafted to be user-friendly and captivating, providing effortless access to course materials and regular updates on progress (see Appendix, Figure 6: Dashboard and Figure 7: Browse page).
* **Authentication:** Authentication is managed through the use of "Clerk," which guarantees secure management of user data and proper control over access to different areas of the platform.
* **Video Hosting:** MUX is responsible for managing video hosting and offers powerful video streaming capabilities that are crucial for delivering educational content of superior quality.
* **File Uploads:** The "uploadthing" feature enables the efficient management of educational resources and attachments related to the courses through file uploads.

## System Interactions

* **User Authentication:** Users (teachers and students) access the platform by logging in through Clerk, which verifies their identities and redirects them to the relevant dashboard based on their role.
* **Course Management and Creation:** Educators utilize user-friendly and functional interfaces to create and modify courses. Users have the ability to upload videos using the MUX platform and handle course files through uploadthing. For more details, please refer to the Appendix, specifically Figure 11: New Course Setup page and Figure 12: Chapter Creation page.
* **Course Enrollment and Access:** Students peruse the catalog of courses, apply filters based on categories, and register for them. The secure handling of course payments is facilitated through the integration of Stripe (see Appendix, Figure 14: Stripe Payments page).
* **Content Delivery:** After enrolling, students can access course content, which is efficiently and securely streamed through MUX, while other course materials are provided through uploadthing.
* **Analytics and Reporting:** Educators utilize the analytics dashboard to access comprehensive reports on course performance, revenue, and metrics related to student engagement (see Appendix, Figure 13: Analytics page).

## Infrastructure

The platform is hosted on Amazon Web Services (AWS), leveraging a variety of services to guarantee optimal availability and performance.

* **Amazon EC2** instances are responsible for hosting the web servers and application logic.
* **Amazon RDS (PostgreSQL)** offers reliable and strong database services.
* **AWS Lambda** and **API Gateway** oversee serverless functions, specifically email notifications.
* **Amazon SES** manages email communications.
* **AWS IAM** guarantees the secure management of access to AWS resources.

This overview offers a comprehensive explanation of the functionality and infrastructure of the Online Academy platform, serving as a foundation for more in-depth discussions in later sections of this document.

# Design Considerations

The design of the Online Academy platform is shaped by various factors that determine the approach to both the technical development and user experience design. This section provides an overview of important factors to consider, such as assumptions, dependencies, constraints, goals, guidelines, and the selected development methodology.

## Assumptions and Dependencies:

* **Assumptions:**
  + Users will possess reliable internet connectivity, which is essential for streaming video content and accessing online resources.
  + Users possess a level of familiarity with fundamental navigation and interaction in digital learning environments, which has an impact on the design of interfaces and instructional layouts.
* **Dependencies:** 
  + The performance of the platform relies significantly on third-party services such as Clerk for authentication, MUX for video hosting, and Stripe for payment processing. Any interruption in these services may affect the operation of the platform.
  + The efficacy of the system depends on the stability and dependability of the AWS infrastructure, which encompasses services such as EC2, RDS, Lambda, and SES.

## General Constraints:

* **Technological limitations:**
  + The selection of technologies, specifically for the backend (e.g., utilizing AWS Lambda and API Gateway), imposes restrictions on the implementation and scalability of serverless functions.
  + Ensuring consistent user experience across different devices and browsers is crucial, necessitating design adaptations to address browser compatibility and mobile responsiveness.
* **Budgetary constraints:** 
  + Financial limitations may impact the extent of features that can be initially implemented, requiring a gradual approach to platform development.
* **Time Constraints:**
* The project's timeline is limited by the requirement to launch before the upcoming academic year, which affects the extent of the initial release.

## Goals and Guidelines:

* **Performance Goals:** 
  + The platform should be capable of accommodating simultaneous usage by thousands of users without experiencing substantial decline in performance, especially during periods of high usage.
  + The user did not provide any text. The objective is to ensure that video content loads with minimal buffering, in order to provide a smooth user experience, as described in the appendix (see Figure 8: Course page).
* **Security Objectives:**
* Safeguard user data through strong encryption and adhere to data protection regulations such as GDPR and CCPA.
* Employ robust payment protocols to ensure the protection of sensitive financial data throughout transactions (see Appendix, Figure 14: Stripe Payments page).
* **Usability Goals:** 
  + Develop a user interface that is easy to understand and use, reducing the time it takes for new users to become proficient and improving the overall satisfaction of users.
  + Ensure compliance with accessibility standards to cater to users with disabilities.

## Development Methods:

* **Agile Development:** 
  + The project will employ an agile development methodology to facilitate iterative testing, development, and feedback. This approach enables adaptability in accommodating new requirements or modifications to existing ones during the development process.
  + Regular sprints and scrum meetings will be utilized to guarantee ongoing advancement and synchronization with project objectives.
* **User-Centered Design (UCD):**
  + Prioritizes user feedback and usability testing during the design process to ensure that the platform effectively addresses the practical needs of teachers and students.
  + The user did not provide any text. The prototyping and user testing phases will be essential, as outlined in the appendix under Figures 10: New Course Creation page and Figure 12: Chapter Creation page.

This section of the design document establishes the groundwork for tackling technical and user experience obstacles, providing guidance to the development team in constructing a platform that is both resilient and intuitive for users. These factors influence the subsequent architectural and detailed design decisions discussed later in this document.

# Architectural Strategies

The architectural strategies implemented for the Online Academy platform are specifically designed to guarantee scalability, reliability, and security, all while ensuring a smooth and uninterrupted user experience. These strategies utilize AWS services extensively to construct a resilient and adaptable architecture that can accommodate evolving needs and scale accordingly. In this document, we present the primary architectural approaches utilized in the creation of the platform.

## Strategy 1: Microservices Architecture

* **AWS Implementation:** 
  + **AWS Lambda:** The platform will employ AWS Lambda to execute backend functions responsible for performing specific tasks, such as sending emails through SES or processing payments. This approach enables each function to independently adjust its capacity according to the level of demand.
  + **API Gateway:** The API Gateway is a tool used to effectively handle and direct API requests to Lambda functions and other services. This configuration facilitates the maintenance of orderly and protected access to the backend functionalities of the platform.
* **Benefits:**
* Enhanced scalability and simplified administration of individual components.
* Improved capability to update and implement services autonomously, resulting in decreased periods of inactivity and faster introduction of new features.

## Strategy 2: Managed Database Services

* **Utilizing AWS for implementation:**
  + Amazon RDS(PostgreSQL): Employing Amazon RDS (PostgreSQL) for the purpose of ensuring dependable and expandable database management. RDS simplifies the process of configuring, managing, and expanding relational databases in the cloud.
* **Benefits:** 
  + The use of automated backups, patching, and scaling reduces the amount of administrative work required and improves the durability and availability of data.
  + The user did not provide any text. By integrating with AWS IAM, database access is effectively and securely managed, following the principle of least privilege.

## Strategy 3: Security and Compliance

* **Utilization of AWS:**
* **AWS IAM:** Oversees user identities and access policies, guaranteeing that only authorized users can gain access to specific resources within the AWS environment.
* **Advantages:**
  + Strong security measures safeguard sensitive information and guarantee adherence to different data protection regulations.
  + Efficient access control mechanisms deter unauthorized access and mitigate data breaches.

## Strategy 4: Scalable Storage Solutions

* **Implementation with “uploadthing”:**
* The "uploadthing" service is used for the storage and retrieval of data of any size, at any given time. This is optimal for storing course materials, user-generated content, and backup files.
* **Benefits:**
* Highly durable and effortlessly scalable storage that automatically adapts to meet the system's requirements.
* A cost-effective storage solution that seamlessly integrates with other services utilized by the platform, thereby improving overall system efficiency and dependability.

The architectural strategies play a crucial role in determining the fundamental infrastructure of the Online Academy platform. By utilizing AWS services, the platform guarantees its ability to effortlessly manage large quantities of user interactions and data processing, all while upholding stringent security measures and ensuring user satisfaction. The System Architecture section of this document provides further details on the role and interaction of each component.

# System Architecture

The Online Academy platform is built with a system architecture that is specifically designed to provide a scalable, secure, and efficient online learning environment. It combines multiple AWS services and third-party tools to establish a unified framework that fulfills the functional needs of both students and teachers. This section presents a concise summary of the fundamental elements and their interconnections within the system.

## Core Components

* **Frontend of the web application**
* **Technologies utilized:** React.js, HTML5, Tailwind CSS, JavaScript
* **Functionality:** Offers the user interface for both students and teachers. The system comprises dynamic pages for browsing courses, managing dashboards, creating courses, and visualizing analytics.
* **AWS interaction:** Utilizes Amazon EC2 for deployement.
* **Backend API**
* **Technologies Utilized:** Node.js
* **Functionality:** Manages business logic, user administration, course administration, and other server-side operations.
* **AWS Interaction:** Deployed on AWS Elastic Beanstalk or EC2 instances, communicating with Amazon RDS for data storage and AWS Lambda for executing serverless functions.
* **Authentication**
* **Tool Utilized:** Attendant
* **Purpose:** Oversees user authentication and session management, ensuring secure access control throughout the platform.
* **AWS interaction:** Incorporated into the backend API, utilizing AWS IAM for secure access to the API.
* **Video Hosting and Streaming**
* **Utilized Tool:** Multiplexer
* **Functionality:** The platform hosts and streams video content for the courses, guaranteeing excellent playback quality on various devices.
* **AWS Interaction:** Utilizes Amazon EC2 to distribute video content in order to reduce latency.
* **Database**
* **Technology Utilized:** Amazon RDS (PostgreSQL)
* **Purpose:** The system is designed to store and manage all long-lasting data, such as user profiles, course materials, transaction records, and analytics data.
* **Integration with AWS:** AWS manages the interaction, guaranteeing high availability, security, and scalability.
* **File Storage**
* **Technological Utilized:** Uploadthing
* **Purpose:** The purpose of this system is to store and organize various types of files, such as images, documents, and other materials related to courses. These files can be uploaded by both teachers .
* **Payment Processing**
* **Utilized Tool:** Stripe
* **Purpose:** Manages the secure processing of payments for course enrollments.
* **Email Services**
* **Technologies Used:** Amazon Web Services Simple Email Service (AWS SES)
* **Purpose:** Sends transactional emails to users for the purpose of verifying accounts, providing notifications, and conducting marketing activities.
* **AWS Integration:** Utilizes Lambda to initiate email sending in response to user actions or scheduled events.

### System Interactions

* **User Actions (e.g., registering for a course):**
* The user engages with the graphical user interface (GUI) of the web application, which then transmits requests to the Backend Application Programming Interface (API).
* The Backend API handles these requests by communicating with the Database to retrieve or store data, as well as with third-party services such as Stripe for payment processing or MUX for video access.
* **Content Delivery:** Video and static content are distributed and delivered efficiently.(see Appendix, Figure 3: Course page).
* **Data Flow:** The flow of data occurs between the frontend, backend, and different AWS services. This flow is protected by AWS IAM roles and policies, which guarantee that each component can only interact with authorized services.

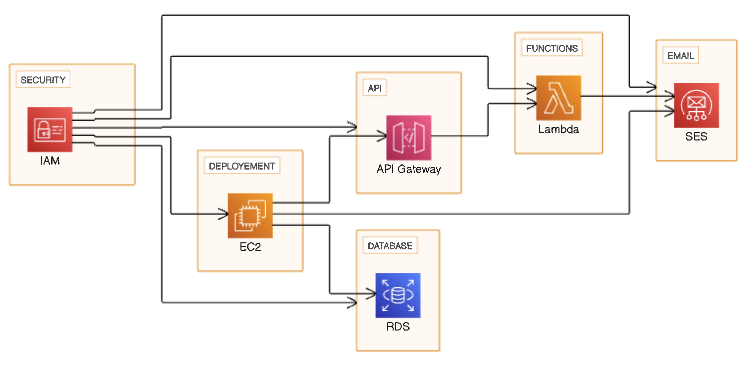


Figure 1: Architecture Diagram

This architecture offers a resilient framework for the Online Academy platform, enabling intricate functionalities while upholding exceptional performance and user contentment. The subsequent sections of the design document provide further elaboration on the intricate interactions and specific design details of each component.

# Detailed System Design

The section on detailed system design provides a comprehensive explanation of the specific elements, their execution, and the way they interact within the Online Academy platform. This segment provides detailed information about two crucial elements that are vital for improving user engagement and functionality: the Editor and the Course Progress components.   
  
Component 1: Editor

**Functionality:**

The Editor component plays a vital role in the process of creating and managing courses for teachers. It enables teachers to format and input course content, including text, images, and links, with the ability to edit the text in a visually appealing manner. This component is widely utilized in the creation of course descriptions, chapter summaries, and any other text that needs formatting.   
  
**Implementation Details:**

* **Technology Used:** The component is implemented using React.js and utilizes the `react-quill` library, which is a robust rich text editor packaged as a React component to ensure smooth integration.
* **Dynamic Loading:** Implements Next.js's `dynamic` function to achieve client-side rendering without server-side rendering (SSR), thereby improving performance and enhancing user experience.
* **Read-Only Mode:** The read-only mode is set up to show content in a format that cannot be edited. It uses the "bubble" theme from Quill, which is designed to display formatted text in a visually pleasing way. You can find an example of this on the New Course Creation page in the Appendix, Figure 10.

**Interactions:**

* Teachers engage with the Editor during the course creation phase, inputting and formatting their course material.
* The entered content is subsequently stored in the Amazon RDS database upon saving or updating the course or chapter.
* Course information is retrieved and displayed in a read-only format using the Editor component when it needs to be displayed.

Component 2: Course Progress

**Functionality:**

The Course Progress component is utilized to visually depict a student's advancement in a specific course. It offers instant feedback on the progress made in the course, thereby increasing the user's motivation and involvement.   
  
**Implementation Details:**

* **Technology Utilized:** This component is constructed utilizing React.js and Tailwind CSS for the purpose of styling.
* **Process Indicator:** The progress indicator uses a generic progress bar component that is styled dynamically based on the course completion status and design variants provided through props.
* **Responsive Design:** The responsive design of the platform adjusts to different sizes (default, sm) and color schemes (default, success) depending on the context in which it is used. This ensures a consistent appearance and user experience across the platform. For more details, please refer to Appendix, Figure 6: Dashboard.

**Interactions:**

* The component retrieves the completion percentage from the backend, which calculates the progress by comparing the number of chapters completed by the student with the total number of chapters in the course.
* The data is continuously updated in real-time on the user's dashboard and course overview pages, offering the user the most current progress information.

**Security and Data Handling:**

* The database interactions are protected using AWS IAM roles, guaranteeing that only authenticated and authorized users can retrieve and update data.
* User progress, along with other sensitive data, is handled in accordance with privacy standards and regulations to ensure the preservation of data integrity and confidentiality.

These components are essential to the Online Academy platform, with each serving distinct roles that improve the functionality and user experience of the system. Their implementation not only satisfies the platform's operational requirements but also guarantees the system's durability, scalability, and security. Additional information regarding the remaining elements and their interconnections can be located in the following sections of this document.

## ER Diagram

The Entity-Relationship (ER) Diagram visually displays the entity types, their attributes, and the relationships between them in the Online Academy platform. The diagram is crucial for comprehending the database's structure and the interconnections and interactions of data elements throughout the system.

A screenshot of a computer

Description automatically generated

Figure 2: ER Diagram

## Use Cases Diagrams

A diagram of a software company

Description automatically generated

Figure 3: User Authentication

A diagram of a course

Description automatically generated

Figure 4: Course Creation

A diagram of a course

Description automatically generated

Figure 5: Course Enrollment

# Glossary

The purpose of this glossary is to offer precise definitions for the terms and concepts that are utilized in the design document of the Online Academy platform. Gaining comprehension of these terms will enhance the clarity of their utilization within the framework of the system and its architecture.

* **A**
* **API (Application Programming Interface):** Software applications are constructed and operated according to a collection of regulations and procedures known as rules and protocols. An API facilitates communication between various software systems.
* **AWS (Amazon Web Services):** Amazon provides a comprehensive and dynamic cloud computing platform that offers a combination of infrastructure as a service (IaaS), platform as a service (PaaS), and packaged software as a service (SaaS) options.
* **AWS IAM (Identity and Access Management):** An AWS resource access management service that provides secure control over user access to AWS resources.
* **AWS Lambda:** A serverless compute service that allows you to execute code without the need to provision or manage servers. It operates by responding to events and automatically handles the allocation and management of compute resources.
* **B**
* **Backend:** The back-end of a software system refers to the portion that is not directly accessible to the user. It is primarily responsible for storing and manipulating data, as well as carrying out background operations.
* **C**
* **Clerk:** A third-party service employed for the purpose of overseeing user authentication and session management in applications.
* **D**
* **Database:** A systematically arranged compilation of organized information or data, usually stored digitally in a computer system. The backend utilizes this to store and retrieve data as needed by the software applications.
* **E**
* **EC2 (Elastic Compute Cloud):** Amazon's cloud computing platform, AWS, includes a service that enables users to lease virtual computers for running their own computer applications.
* M
* **MUX:** A video platform that simplifies the process of uploading and streaming video content.
* P
* **Progress Component:** A user interface element that graphically depicts the degree of completion of a process or activity.
* **PostgreSQL:** A DBMS that is open-source and focuses on being extensible and compliant with SQL.
* R
* **React.js:** A JavaScript library designed for constructing user interfaces, particularly for single-page applications that require synchronization between the state and the data.
* **RDS (Relational Database Service):** Amazon Web Services (AWS) offers a distributed relational database service that aims to streamline the process of setting up, managing, and expanding a relational database for application use.
* S
* **SES (Simple Email Service):** A service offered by AWS that enables the transmission of emails from any application.
* **SSL/TLS (Secure Sockets Layer / Transport Layer Security):** Internet encryption protocols are used to secure data transmitted between systems, such as web servers and browsers, in order to prevent unauthorized access and ensure that only the intended recipient can read the information.
* **Stripe:** An e-commerce payment gateway that securely manages online transactions for internet businesses.
* **U**
* **UI (User Interface):** The methods through which a user interacts with a computer system, especially when using software and input devices.
* **V**
* **Video Hosting:** An online platform that enables users to upload and stream video content.

The purpose of this glossary is to facilitate comprehension of the specialized terminology and acronyms utilized in the design document. It serves as a point of reference for project stakeholders and technical teams involved in the project.

# Appendix

A screenshot of a computer

Description automatically generated

Figure 6: Dashboard

A screenshot of a computer

Description automatically generated

Figure 7: Browse page

A screenshot of a computer

Description automatically generated

Figure 8: Course page

A screenshot of a computer

Description automatically generated

Figure 9: Teacher mode Courses page

A screenshot of a computer

Description automatically generated

Figure 10: New Course Creation Page

A screen shot of a computer

Description automatically generated

Figure 11: New Course Setup page

A screenshot of a computer

Description automatically generated

Figure 12: Chapter Creation Page

A screenshot of a computer

Description automatically generated

Figure 13: Analytics Page

A screenshot of a computer

Description automatically generated

Figure 14: Stripe payments page