

**THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY**

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE TECHNOLOGY

PROJECT TITLE:

**COURSE MANAGEMENT INTEGRATE A
RECOMMENDATION SYSTEM**

Instructor: Ph.D. PHAM CONG THANG

Student: TRUONG HUY HOANG

Student ID: 102200090

Class: 20TCLC_DT2

Da Nang, 06/2024

THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE TECHNOLOGY

PROJECT TITLE:
**COURSE MANAGEMENT INTEGRATE A
RECOMMENDATION SYSTEM**

Instructor: **Ph.D. PHAM CONG THANG**

Student: **TRUONG HUY HOANG**

Student ID: **102200090**

Class: **20TCLC_DT2**

Da Nang, 06/2024

SUMMARY

Topic title: Course management integrate a recommendation system

Student name: Truong Huy Hoang

Student ID: 102200090

Class: 20TCLC_DT2.

In the field of educational technology, integrating advanced systems to enhance learning experiences is crucial. This thesis presents the development and implementation of a sophisticated course management system, incorporating a recommendation engine to improve user engagement and streamline course-related processes.

This project's main aim is to design and deploy a comprehensive course management system with an embedded recommendation engine. The system is intended to facilitate course discovery, enrollment, and engagement, ultimately enhancing the overall learning experience for users. The objectives include developing an intuitive interface for easy course navigation and interaction, implementing a robust engine to suggest relevant courses based on user preferences and history, providing tools for course management and analytics to assist administrators, and enhancing the system's capacity to accommodate a growing number of users.

The project focuses on the development and implementation of the course management system. The target audience includes students, learners, instructors, and administrators who interact with the platform for managing courses, enrollments, and recommendations.

GRADUATION PROJECT REQUIREMENTS

Student Name: Truong Huy Hoang Student ID: 102200090

Class: 20TCLC_DT2 Faculty: Information Technology

Major: Software Engineering

1. *Topic title: Course management integrate a recommendation system*
2. *Project topic : ☐has signed intellectual property agreement for final result*
3. *Initial figure and data:*

Initial data sources include publicly available course data from educational platforms such as Udemy, Coursera, and academic journals focusing on educational technology. This includes information on course enrollment trends, user preferences, and feedback collected from users and instructors.

4. *Drawings, charts (specify the types and sizes of drawings):*
 - *Use Case Diagrams (Standard size)*
 - *Class Diagrams (Standard size)*
 - *Architecture Structure Diagrams (Standard size)*

5. <i>Name of instructor:</i>	<i>Content parts:</i>

6. *Date of assignment :* /...../202.....

7. *Date of completion :* /...../202.....

Da Nang, date month year

Head of Division.....

Instructor

PREFACE

In today's educational landscape, efficient course management is crucial for enhancing the learning experience. Both instructors and students encounter challenges in organizing courses, tracking progress, and discovering relevant educational content.

Learners often struggle to find courses that align with their interests and skill levels, while instructors require a robust platform to manage and deliver their courses effectively. Administrators need comprehensive tools to oversee the system, ensuring quality and consistency.

This project involves developing a course management website integrated with a recommendation system to address these challenges for end users, instructors, and administrators. End users can search for courses, register using various authentication methods, enroll in free or paid courses, receive personalized course recommendations, update their profiles, and engage with course content through comments and ratings. Instructors can manage their courses, track orders, and respond to student feedback. Administrators have tools to manage courses, orders, user accounts, and categories, along with a dashboard for system-wide monitoring.

The recommendation system uses deep learning algorithms trained on course data from platforms like Udemy, providing users with tailored course suggestions. This integration not only enhances the user experience but also aligns with the trend of applying digital technology in education.

This project leverages modern management and recommendation technologies to offer an effective solution for educational institutions, improving the quality and delivery of courses.

ASSURANCE

I understand the school's policy on academic integrity and hereby certify that:

- This project has been carried out under the guidance of Ph.D. Pham Cong Thang.
- All references used in this thesis are clearly and honestly cited with the author's name, project title, publication date, and location.
- The content of this project is my own and has not been copied from any other source.

Student Performed

TABLE OF CONTENT

Chapter 1: OVERVIEW OF TOPIC	4
1.1 Introduction to the Topic.....	4
1.2 Reasons for Choosing the Topic.....	4
1.3 Objectives of the Project.....	4
1.4 Practical Significance	4
1.5 Problem Statement	5
1.5.1 Functionalities	5
1.5.2 Requirements.....	5
1.6 Project Phases	5
1.7 Challenges and Solutions	6
Chapter 2: LITERATURE REVIEW	6
2.1 Recommendation Systems	7
2.1.1 Types of Recommendation Systems:	7
2.1.2 Content-Based Filtering Approach:.....	7
2.2 Technology Stack.....	8
2.2.1 Backend Technologies.....	8
2.2.2 Database Management.....	8
2.2.3 Frontend Technologies	9
2.2.4 Storage and Media Management	9
2.2.5 Deep Learning Framework	9
2.2.6 Authentication and Security	10
2.2.7 Payment Integration	10
2.3 Integration of Recommendation Systems in Course Management.....	10
2.3.1 Current State of Integration.....	10
2.3.2 Case Studies	10
2.4 Summary	11
Chapter 3: SYSTEM ANALYSIS AND DESIGN	12
3.1 Use case overview	12
3.2 Use Case Diagrams.....	13
3.3 Use Case Specifications	16
3.3.1 Registration use case	16

3.3.2 Login use case	18
3.3.3 Logout use case	19
3.3.4 Search course use case	21
3.3.5 Course registration use case	22
3.3.6 Update account information use case	24
3.3.7 Rating course use case	25
3.3.8 Comment on lesson use case	27
3.3.9 Create course use case	28
3.3.10 Update course use case	29
3.3.11 Delete course use case	31
3.3.12 View instructor dashboard use case	32
3.3.13 Respond to comments use case	33
3.3.14 Manage order use case	35
3.3.15 Manage users use case	36
3.3.16 Manage categories use case	37
3.3.17 Manage roles use case	39
3.3.18 Analytics and reporting use case	40
3.4 Sequence Diagrams	42
3.5 Database Design.....	46
3.3.1 Database Tables	46
3.3.2 Detail Table Descriptions	47
Chapter 4. RECOMMENDATION SYSTEM.....	50
4.1 Data Collection	50
4.1.1 Introduction	50
4.1.2 Methodology	51
4.1.3 Data Handling and Storage	51
4.2 Data Preprocessing.....	52
4.2.1 Introduction	52
4.2.2 EDA (Exploratory Data Analysis)	52
4.2.3 Data Cleaning	54
4.2.4 Feature Engineering	55
4.2.5 Integration and Transformation	57
4.3 Training and Evaluating the Model.....	57
4.3.1 Model Construction	57
4.3.2 Training the Model.....	58

4.3.3 Model Evaluation	59
4.3.4 Results	59
4.3.5 Conclusion	59
Chapter 5. Demo application	61
5.1 Authentication	61
5.2 Home screen.....	61
5.3 Search course	62
5.4 Profile	63
5.5 Admin dashboard.....	63
5.6 Manage courses.....	64
5.7 Create courses.....	65

LIST OF TABLE AND FIGURE

Table 3.1 Overview of Use Cases.....	13
Table 3.2 Registration Use Cases	18
Table 3.3 Login Use Case	19
Table 3.4: Logout Use Case.....	21
Table 3.5: Search Course Use Case.....	22
Table 3.6: Registration Course Use Case.....	24
Table 3.7: Update Account Information Use Case	25
Table 3.8: Rating Course Use Case	26
Table 3.9: Comment on lesson Use Case	28
Table 3.10: Create Course Use Case	29
Table 3.11: Update Course Use Case	31
Table 3.12: Delete Course Use Case	32
Table 3.13: Instructor Dashboard Use Case	33
Table 3.14: Reply comment Use Case	35
Table 3.15: Manage orders Use Case.....	36
Table 3.16: Manage Users Use Case.....	37
Table 3.17: Manage Categories Use Cases	39
Table 3.18: Manage Role Use Cases.....	40
Table 3.19 Analytics and Reporting Use Case	41
Table 3.20 User table	48
Table 3.21: Table course	49
Table 3.22: Table order	49
Figure 3.1 End user use case diagram	13
Figure 3.2 Admin use case diagram	13
Figure 3.3 Instructor use case diagram	14
Figure 3.4 End user enroll course use case diagram	14
Figure 3.5 Admin and instructor manage course use case diagram	15
Figure 3.6 End user and instructor comment on course use case diagram	15
Figure 3.7 Order management use case diagram	16

Figure 3.8 Enrollment course sequence diagram.....	43
Figure 3.9 Course commented sequence diagram	44
Figure 3.11 Course management sequence diagram	46
Figure 3.12 Database Diagram	50
Figure 4.1 Distribution of price	52
Figure 4.3 Distribution of rating	53
Figure 4.4 The Transformer model architecture.....	56
Figure 4.5 Training loss over Epoch	58
Figure 5.1 Authentication screen.....	61
Figure 5.2 Home screen	62
Figure 5.3 Search course screen	62
Figure 5.4 Profile screen.....	63
Figure 5.5 Admin dashboard screen	64
Figure 5.6 Manage course screen	64
Figure 5.7 Create course screen.....	65

LIST OF SYMBOLS, ACRONYM

Acronyms	Full name
BERT	Bidirectional Encoder Representations from Transformers
SSR	Server-Side Rendering
JWT	JSON Web Token
NCF	Neural Collaborative Filtering
UI	User Interface
API	Application programming interface
RMSE	Root Mean Square Error
MAE	Mean Absolute Error

INTRODUCTION

In educational technology, the integration of advanced systems to enhance learning experiences has become increasingly pivotal. This thesis embarks on a journey to develop and implement a sophisticated course management system integrated with a recommendation engine. This introduction outlines the purpose, objectives, scope, target audience, research methodology, and structure of this graduation project.

Purpose of the Project:

This project's main purpose is to design and deploy a comprehensive course management system that incorporates a recommendation engine. This system aims to streamline the process of course discovery, enrollment, and engagement for users, thereby improving the overall learning experience.

Objectives of the Project:

- Develop a user-friendly interface for seamless course navigation and interaction.
- Implement a robust recommendation engine to suggest relevant courses based on user preferences and history.
- Provide administrative tools for course management and analytics.
- Enhance system scalability and performance to accommodate a growing user base.
- Scope and Research Target:
 - This project focuses on the development and implementation phases of the course management system. The research targets include end users (students and learners), instructors, and administrators who interact with the platform to manage courses, enrollments, and recommendations.

Research Methodology:

The research methodology encompasses a combination of literature review, system design and development, implementation of recommendation algorithms, usability testing, and feedback collection. The project employs both qualitative and quantitative approaches to evaluate system effectiveness and user satisfaction.

Structure of the Thesis:

Chapter 1: Overview of topic

Chapter 2: Literature Review

Chapter 3: System Analysis and Design

Chapter 4: Recommendation System

Chapter 5: Demo Application

Summarizes the findings, discusses limitations, and proposes recommendations for future enhancements and research directions.

This structure ensures a comprehensive exploration of the project's development, implementation, and evaluation phases, aiming to contribute to the advancement of educational technology and user-centric learning experiences.

Chapter 1: OVERVIEW OF TOPIC

1.1 Introduction to the Topic

- **Course Management Systems:** An overview of what course management systems are, their purpose, and their importance in modern education.
- **Integration of Recommendation Systems:** Introduction to recommendation systems, their role in enhancing user experience, and the significance of integrating such systems into course management platforms.

1.2 Reasons for Choosing the Topic

- **Growth of Online Education:** Discussion on the rapid expansion of online education and the increasing demand for effective course management systems.
- **Need for Personalization:** Emphasis on the necessity for personalized learning experiences to improve engagement and learning outcomes.
- **Technological Advancements:** The availability of advanced technologies, such as deep learning, to develop sophisticated recommendation systems.

1.3 Objectives of the Project

- **Develop a Comprehensive Web Application:** Creating a platform that integrates course management with a recommendation system.
- **Enhance User Experience:** Providing personalized course recommendations to users based on their interactions and preferences.
- **Streamline Management:** Offering efficient tools for instructors and administrators to manage courses, orders, and user interactions.

1.4 Practical Significance

- **Improved Learning Outcomes:** The potential impact of personalized recommendations on student engagement and success.
- **Enhanced Administrative Efficiency:** Benefits of streamlined course and user management for instructors and administrators.
- **User Satisfaction:** Increased user satisfaction through tailored course suggestions and a seamless user experience.

1.5 Problem Statement

- **Course Management Challenges:** Identifying common issues in existing course management systems, such as lack of personalization and complex management processes.
- **Solution Overview:** Introducing the proposed solution that integrates a recommendation system to address these challenges.

1.5.1 Functionalities

End User:

- Search for courses
- Register and login via email, GitHub, Google (including email verification)
- Register for courses (free or paid via Stripe)
- Receive course recommendations
- Update account information
- Comment on lessons
- Rate courses

Instructor:

- Manage courses (CRUD operations)
- Dashboard for overview and analytics
- Manage orders
- Respond to comments on lessons

Administrator:

- Manage courses
- Manage orders
- Manage user accounts
- Manage categories
- Dashboard for administrative tasks and analytics

1.5.2 Requirements

- **Functional Requirements:** Detailed list of functionalities the system must provide for each user type.
- **Non-Functional Requirements:** Performance, security (e.g., email verification, secure payment processing), scalability, usability.

1.6 Project Phases

- **Phase 1: Requirements Gathering and Analysis**

Collecting detailed requirements from stakeholders.

- **Phase 2: System Design**

Designing the architecture, database, and user interfaces.

- **Phase 3: Implementation**

Developing the backend, frontend, and integrating the recommendation system.

- **Phase 4: Testing**

Conducting unit, integration, system, and user acceptance testing.

- **Phase 5: Deployment**

Deploying the system to a live environment and monitoring performance.

- **Phase 6: Maintenance and Updates**

Ongoing support and iterative improvements based on user feedback.

1.7 Challenges and Solutions

- **Data Collection:** Collecting sufficient and relevant data for training the recommendation system.
- **Solution:** Utilize publicly available datasets from platforms like Udemy.
- **Integration Complexity:** Integrating the recommendation system with the course management platform.
- **Solution:** Modular design to facilitate seamless integration.
- **Scalability:** Ensuring the system can handle a large number of users and courses.
- **Solution:** Implement scalable architecture and technologies.
- **User Privacy and Security:** Protecting user data and ensuring secure transactions.
- **Solution:** Implement robust security measures such as encryption, secure payment gateways, and thorough testing.

Chapter 2: LITERATURE REVIEW

2.1 Recommendation Systems

2.1.1 Types of Recommendation Systems:

- Collaborative Filtering: This method utilizes the behavior and preferences of a group of users to generate recommendations. It analyzes patterns in user interactions like ratings, clicks, and purchases to identify similarities between users or items. Collaborative filtering can be categorized into user-based and item-based approaches. The user-based method recommends items based on the preferences of similar users, whereas the item-based method suggests items like those the user has previously interacted with.
- Content-Based Filtering: Unlike collaborative filtering, this approach recommends items based on their similarity to items the user has engaged with in the past. It examines attributes such as item descriptions, keywords, and metadata, along with the user's profile and preferences. By analyzing these features, content-based filtering identifies items that are likely to be relevant to the user's interests.
- Hybrid Systems: These systems integrate collaborative filtering and content-based filtering techniques to capitalize on their respective strengths. By combining multiple recommendation strategies, hybrid systems enhance the accuracy and reliability of recommendations. For instance, Netflix employs a hybrid approach to suggest movies and TV shows, blending user behavior data with content attributes to provide personalized recommendations.

2.1.2 Content-Based Filtering Approach:

- Algorithm Description: Content-based filtering involves representing both items (courses) and user profiles in terms of features. Each course is characterized by attributes such as title, description, keywords, and topics. A user's profile is built based on their interactions with courses, including courses they have viewed, enrolled in, or rated. The recommendation algorithm calculates the similarity between the user's profile and available courses using techniques such as cosine similarity or Euclidean distance.
- Application in Project: The recommendation system analyzes the features of courses the user has interacted with and recommends new courses with similar attributes. This approach ensures that recommendations are tailored to the user's specific interests and preferences. For instance, if a user has previously enrolled

in courses related to web development, the system will suggest other web development courses.

- Advantages of Content-Based Filtering:
 - Personalization: Provides highly personalized recommendations based on the user's past interactions and preferences.
 - Cold Start Problem: Addresses the cold start problem for new users by recommending items based on their initial interactions without relying on other users' data.
 - Transparency: Allows users to understand why certain items are recommended, as the recommendations are directly based on their own past behavior.

2.2 Technology Stack

2.2.1 Backend Technologies

- Node.js: Node.js utilizes Chrome's V8 JavaScript engine to execute JavaScript outside of the browser environment. Its non-blocking, event-driven architecture makes it ideal for developing scalable network applications. In this project, Node.js manages server-side operations, handles API requests, and facilitates real-time communication.
- Express.js: Express.js serves as a lightweight and adaptable web application framework for Node.js. It offers a comprehensive suite of functionalities essential for web and mobile applications, such as routing, middleware integration, and HTTP utilities. Utilized in this project to develop a RESTful API, Express.js manages routes, processes incoming requests and outgoing responses, and incorporates middleware for tasks like authentication and validation

2.2.2 Database Management

- MongoDB: MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. Its schema-less nature allows for easy scalability and iterative development. In this project, MongoDB stores user data, course information, interaction logs, and recommendation results. The database's flexibility is ideal for managing the diverse and evolving data structures in a course management system.

- Redis: Redis functions as an in-memory data structure store that serves multiple roles including database, cache, and message broker. It supports diverse data structures like strings, hashes, lists, sets, and sorted sets. In this project, Redis is employed to cache frequently accessed data such as user sessions and recommendation outcomes, enhancing performance and minimizing latency.

2.2.3 Frontend Technologies

- Next.js: Next.js, a React framework, empowers server-side rendering (SSR) and static site generation, enhancing web application performance and SEO by pre-rendering pages either at build time or on demand. This project utilizes Next.js to construct the user interface, ensuring a seamless and responsive experience for users.
- Redux: Redux serves as a predictable state container designed for JavaScript applications, facilitating consistent management of application state. Integrated with React components in this project, Redux handles global state management tasks such as user authentication status, course data, and recommendation lists, thereby simplifying debugging and testing processes.

2.2.4 Storage and Media Management

Cloudinary: Cloudinary offers a cloud-based service specializing in comprehensive image and video management solutions. It oversees media uploads, storage, transformations, and delivery processes. Within this project, Cloudinary is responsible for managing course images and videos, optimizing them for various devices and ensuring efficient, dependable delivery.

2.2.5 Deep Learning Framework

PyTorch: PyTorch, an open-source machine learning library built on the Torch library, is extensively utilized in fields like computer vision and natural language processing. It stands out for its dynamic computational graphs, which simplify the development and training of deep learning models. Within this project, PyTorch plays a crucial role in developing the recommendation system's deep learning model, harnessing its flexibility and user-friendly nature.

2.2.6 Authentication and Security

- OAuth: OAuth is an open standard utilized for access delegation, often employed to authorize websites or applications to access user information without the need to expose passwords. In this project, OAuth is integrated to allow users to log in securely using their Google or GitHub accounts, ensuring a safe and convenient authentication process.
- JWT (JSON Web Tokens): JWT is a compact, URL-safe method of representing claims exchanged between two parties. It serves as a secure mechanism for user authentication and authorization in this project. JWTs are generated during user login and are subsequently included in requests to validate the user's identity and permissions effectively.

2.2.7 Payment Integration

Stripe: Stripe is a technology company that builds economic infrastructure for the internet. It provides tools for accepting payments online and in mobile apps. In this project, Stripe is integrated to handle payments for course registrations, ensuring secure and efficient payment processing. Stripe's API simplifies the process of managing transactions, handling disputes, and ensuring compliance with financial regulations.

2.3 Integration of Recommendation Systems in Course Management

2.3.1 Current State of Integration

Many existing course management systems rely on manual curation and static recommendations, which can limit the personalization and relevance of course suggestions. The integration of advanced recommendation systems can significantly enhance the user experience by providing personalized course suggestions, improving engagement and retention.

2.3.2 Case Studies

- Udemy: Udemy employs a combination of collaborative filtering and content-based filtering to recommend courses to users. The platform analyzes user

behavior, such as course enrollments, ratings, and searches, and course attributes like topics and descriptions. This hybrid approach allows Udemy to provide highly relevant and personalized course recommendations.

- Coursera: Coursera utilizes machine learning algorithms to personalize course recommendations. The platform considers user interactions, such as course views, enrollments, and completions, and course content similarities. By leveraging data from millions of users and courses, Coursera's recommendation system enhances the learning experience by suggesting courses that match users' interests and learning goals.

2.4 Summary

- The integration of advanced technologies such as Node.js, Express.js, MongoDB, Redis, Next.js, Redux, PyTorch, and Cloudinary plays a crucial role in the development of a comprehensive course management system with personalized recommendation capabilities.
- The use of a content-based filtering approach for the recommendation system enhances the ability to provide accurate and relevant course suggestions, thereby improving user engagement and learning outcomes.
- Authentication and security measures, along with payment integration via Stripe, ensure a secure and seamless user experience.
- By leveraging state-of-the-art technologies and techniques, this project aims to create an efficient, scalable, and user-friendly course management system that meets the needs of end users, instructors, and administrators.

Chapter 3: SYSTEM ANALYSIS AND DESIGN

3.1 Use case overview

Code	Name	Brief Description
UC01	Registration	Allow end users to register an account using email, Google, or GitHub, and verify their email.
UC02	Login	Allow end users to log in using email, Google, or GitHub.
UC03	Logout	Allow end users to log out from their account.
UC05	Search Courses	Allow end users to search for courses using keywords.
UC06	Course Registration	Allow end users to register for courses, either free or paid, through Stripe.
UC07	Update Account Information	Allow end users to update their account information such as name, email, and password.
UC09	Rating Course	Allow end users to rate and review courses they have completed.
UC10	Comment on Lesson	Allow end users to leave comments on individual lessons within a course.
UC11	Create Course	Allow instructors to create a new course
UC12	Update Course	Allow instructors to update an existing course..
UC13	Delete Course	Allow instructors to delete an existing course.
UC14	View Instructor Dashboard	Allow instructors to view an overview of their course statistics and activities.
UC15	Respond to Comments	Allow instructors to respond to comments left by end users on their lessons.

UC16	Manage Orders	Allow instructors and admin to view and manage orders related to their courses.
UC17	Manage Users	Allow admins to view, update, and delete user accounts.
UC18	Manage Categories	Allow admins to create, update, and delete course categories.
UC19	Manage Roles	Allow admins to assign or change roles of users (e.g., upgrade a user to an instructor)..
UC20	Analytics and Reporting	Provide detailed reports and analytics for admins and instructors regarding user engagement and course metrics.

Table 3.1 Overview of Use Cases

3.2 Use Case Diagrams

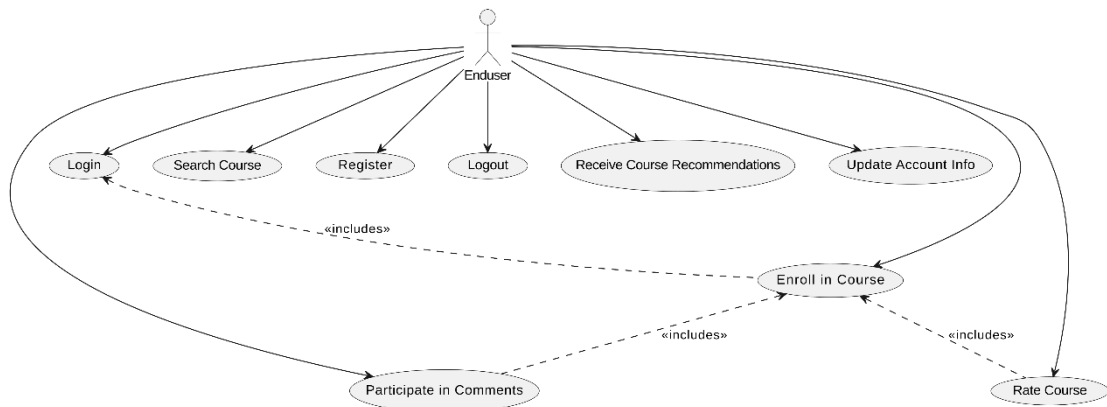


Figure 3.1 End user use case diagram

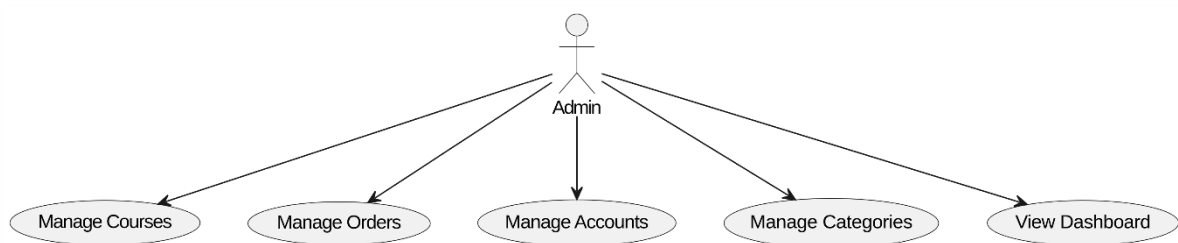


Figure 3.2 Admin use case diagram

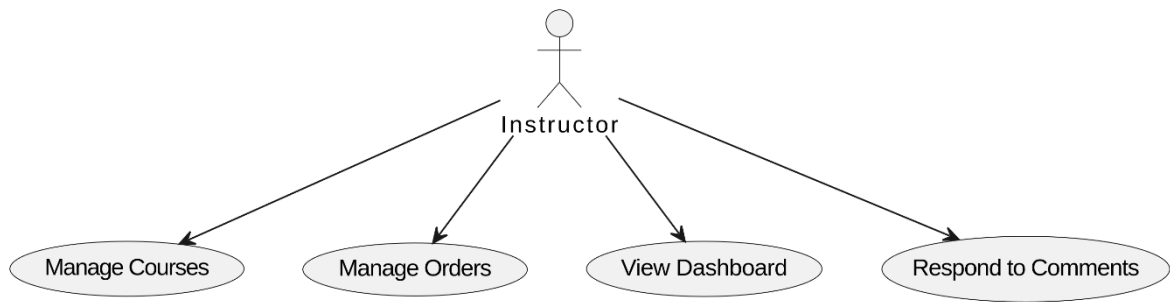


Figure 3.3 Instructor use case diagram

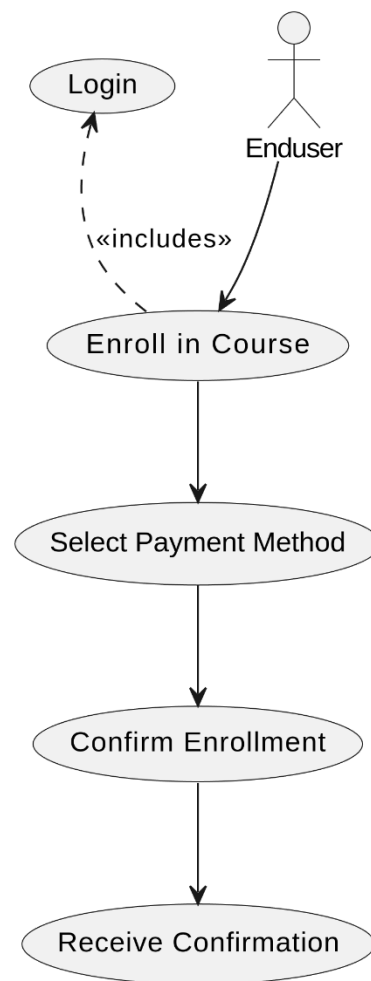


Figure 3.4 End user enroll course use case diagram

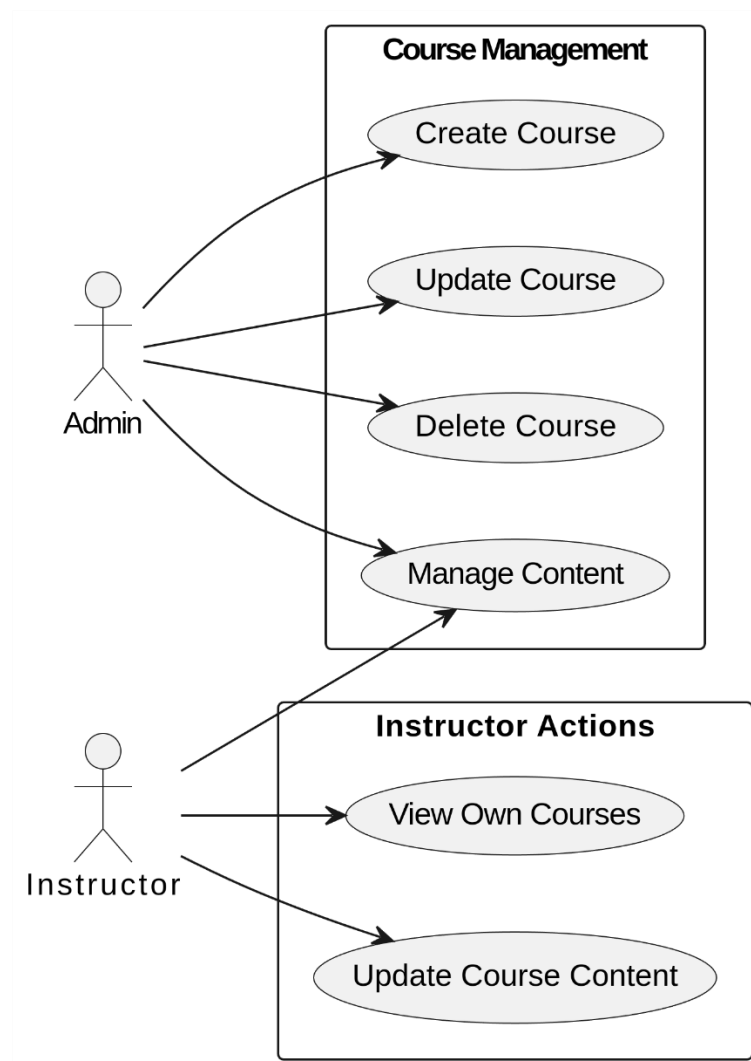


Figure 3.5 Admin and instructor manage course use case diagram

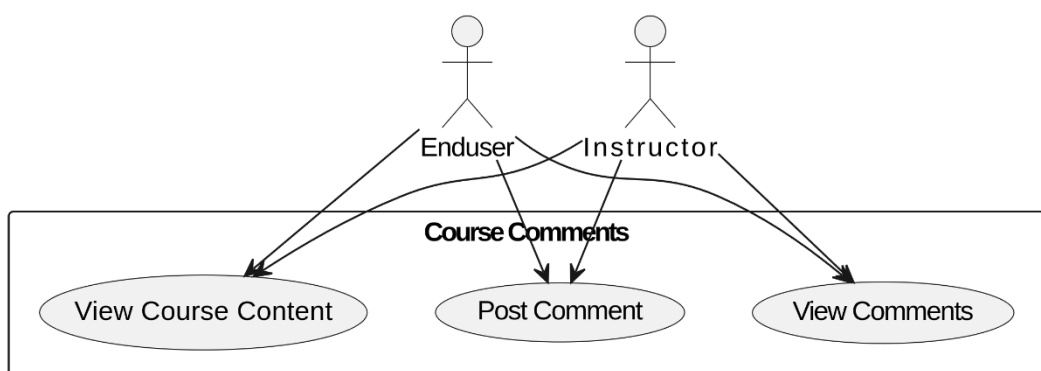


Figure 3.6 End user and instructor comment on course use case diagram

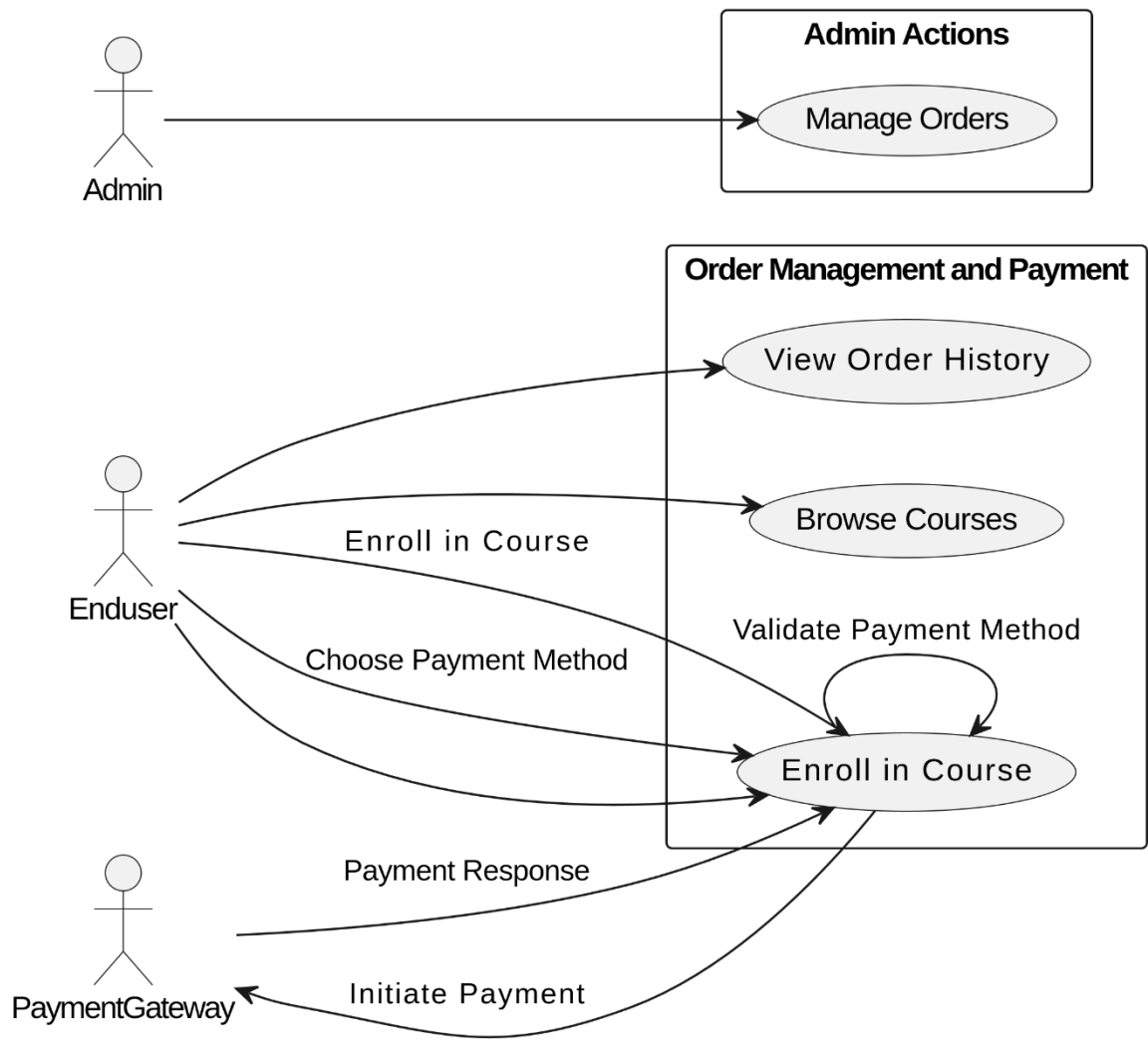


Figure 3.7 Order management use case diagram

3.3 Use Case Specifications

3.3.1 Registration use case

Use Case ID	UC01
Use Case Name	Registration
Actor(s)	End User

Description	Allow end users to register an account using email, Google, or GitHub, and verify their email.
Preconditions	The end user must have access to the internet and a valid email address, Google, or GitHub account.
Postconditions	The end user will have a new account in the system, and their email will be verified.
Main Flow	1. The end user navigates to the registration page. 2. The end user selects a registration method (email, Google, or GitHub). 3. If email is selected: a. The end user enters their email, password, and other required details. b. The system sends a verification email to the provided address c. The end user clicks the verification link in the email. d. The system verifies the email and activates the account. 4. If Google or GitHub is selected: a. The end user is redirected to the respective service's authentication page b. The end user logs in using their credentials c. The system receives authentication confirmation from the service. d. The system creates and activates the account. 5. The end user is redirected to their dashboard with a registration success message.
Alternate Flows	3b1. The email verification link is expired a. The end user requests a new verification email. b. The system sends a new verification email. 4b1. The authentication with Google or GitHub fails: a. The system displays an error message. b. The end user retries the login or selects another registration method.
Exceptions	1a. The email provided is already in use: a. The system displays an error message indicating the email is already registered. b. The end user chooses to log in or use a different email.
Assumptions	The end user has basic knowledge of how to use email and authentication services.
Priority	High
Frequency of Use	Frequent

Business Rules	Email addresses must be unique Passwords must meet security requirements (e.g., minimum length, complexity).
Special Requirements	The system must integrate with Google and GitHub OAuth services. The system must be able to send and receive email verifications reliably.
Notes and Issues	Ensure that the registration process is user-friendly and secure. Monitor for any issues with third-party authentication services.

Table 3.2 Registration Use Cases

3.3.2 Login use case

Use Case ID	UC02
Use Case Name	Login
Actor(s)	End User
Description	Allow end users to log in using email, Google, or GitHub.
Preconditions	The end user must have a registered account in the system.
Postconditions	The end user is authenticated and redirected to their dashboard.
Main Flow	1. The end user navigates to the login page. 2. The end user selects a login method (email, Google, or GitHub). 3. If email is selected: a. The end user enters their email and password. b. The system verifies the email and password. c. The system authenticates the user and redirects them to their dashboard. 4. If Google or GitHub is selected:

	a. The end user is redirected to the respective service's authentication page. b. The end user logs in using their credentials. c. The system receives authentication confirmation from the service. d. The system authenticates the user and redirects them to their dashboard. 5. The end user is greeted with a login success message.
Alternate Flows	3b1. The email or password is incorrect: a. The system displays an error message indicating incorrect credentials. b. The end user retries entering their email and password. 4b1. The authentication with Google or GitHub fails: a. The system displays an error message. b. The end user retries the login or selects another login method.
Exceptions	1a. The user account is locked due to multiple failed login attempts: a. The system displays an account locked message. b. The end user follows account recovery procedures.
Assumptions	The end user has a valid registered account and remembers their credentials.
Priority	High
Frequency of Use	Frequent
Business Rules	Passwords must be verified against stored hashed passwords. User session management must be secure.
Special Requirements	The system must integrate with Google and GitHub OAuth services. The system must handle and store user session tokens securely.
Notes and Issues	Ensure that the login process is secure and user-friendly. Monitor for any issues with third-party authentication services.

Table 3.3 Login Use Case

3.3.3 Logout use case

Use Case ID	UC03
Use Case Name	Logout
Actor(s)	End User
Description	Allow end users to log out from their account.
Preconditions	The end user must be logged into the system.
Postconditions	The end user is logged out, and their session is terminated.
Main Flow	1. The end user clicks the logout button. 2. The system invalidates the user's session token. 3. The system clears any session-related data. 4. The end user is redirected to the homepage or login page with a logout success message.
Alternate Flows	2a. The session token is already invalid: a. The system displays a session expired message and redirects to the login page.
Exceptions	N/A
Assumptions	The end user wants to end their current session.
Priority	High
Frequency of Use	Frequent
Business Rules	The system must ensure all session data is cleared to prevent unauthorized access.
Special Requirements	The system must handle session management securely.

Notes and Issues	Ensure that the logout process is quick and user-friendly. Monitor for any issues with session invalidation.
-------------------------	---

Table 3.4: Logout Use Case

3.3.4 Search course use case

Use Case ID	UC05
Use Case Name	Search Courses
Actor(s)	End User
Description	Allow end users to search for courses using keywords, filter by category, and sort results.
Preconditions	The end user must be logged into the system and connected to the internet.
Postconditions	The system displays a list of courses that match the search criteria, filters, and sorting preferences.
Main Flow	1. The end user navigates to the search bar. 2. The end user enters keywords related to the desired course. 3. The end user selects a category filter, if desired. 4. The end user selects a sorting preference (e.g., by popularity, rating, date). 5. The system processes the search query with the selected filters and sorting. 6. The system retrieves and displays a list of courses that match the search keywords, filters, and sorting preferences. 7. The end user views the search results and selects a course to view more details.
Alternate Flows	3a. No courses match the search keywords: a. The system displays a message indicating no results found. b. The end user refines or changes the search keywords. 3b. The end user enters invalid or incomplete search keywords: a. The system displays an error message and prompts the user to enter valid keywords.

	4a. No courses match the selected category filter: a. The system displays a message indicating no results found for the selected category. b. The end user changes or removes the category filter. 4b. No courses match the sorting preference: a. The system displays results without sorting preference. b. The end user selects a different sorting preference.
Exceptions	N/A
Assumptions	The system's search functionality is responsive and returns relevant results.
Priority	Medium
Frequency of Use	Frequent
Business Rules	The search algorithm should prioritize relevance and user preferences. The search results should be paginated if necessary. Filters and sorting should be applied accurately to the search results.
Special Requirements	The system should support keyword-based search, category filtering, and sorting options.
Notes and Issues	Ensure the search functionality is optimized for speed and accuracy. Monitor search logs to improve relevance and user experience. Validate the effectiveness of filters and sorting options.

Table 3.5: Search Course Use Case

3.3.5 Course registration use case

Use Case ID	UC06
Use Case Name	Course Registration
Actor(s)	End User

Description	Allow end users to register for courses, either free or paid, through Stripe.
Preconditions	The end user must be logged into the system.
Postconditions	The end user is registered for the selected course, and payment is processed if applicable.
Main Flow	1. The end user navigates to the course details page. 2. The end user clicks the "Register" button. 3. The system checks if the course is free or paid. 4. If the course is free: a. The system registers the user for the course. b. The user receives a confirmation message and is redirected to the course content. 5. If the course is paid: a. The system redirects the user to the Stripe payment gateway. b. The user enters payment details and confirms the payment. c. The system processes the payment through Stripe. d. Upon successful payment, the system registers the user for the course. e. The user receives a confirmation message and is redirected to the course content.
Alternate Flows	5b1. Payment details are incorrect or declined: a. The system displays an error message. b. The user re-enters payment details or cancels the registration. 5c1. Payment processing fails: a. The system displays a failure message. b. The user retries the payment or contacts support.
Exceptions	2a. The course is full and does not accept new registrations: a. The system displays a message indicating the course is full. b. The user selects a different course or joins a waitlist.
Assumptions	The end user has sufficient funds or valid payment methods for paid courses.
Priority	High
Frequency of Use	Frequent
Business Rules	The system must accurately track course availability. Payment processing must be secure and comply with Stripe's requirements.
Special Requirements	The system must integrate seamlessly with Stripe for payment processing. The system must handle payment failures gracefully and provide clear instructions to the user.

Notes and Issues	Ensure the registration process is secure and user-friendly. Monitor for any issues with course registration and payment processing.
-------------------------	--

Table 3.6: Registration Course Use Case

3.3.6 Update account information use case

Use Case ID	UC07
Use Case Name	Update Account Information
Actor(s)	End User
Description	Allow end users to update their account information such as name, email, and password.
Preconditions	The end user must be logged into the system.
Postconditions	The end user's account information is updated in the system.
Main Flow	1. The end user navigates to the account settings page. 2. The end user selects the information they want to update (e.g., name, email, password). 3. The end user enters the new information. 4. The system validates the entered information. 5. If the information is valid: a. The system updates the user's account information. b. The system displays a confirmation message indicating a successful update. 6. If the user updates the email address: a. The system sends a verification email to the new email address. b. The user clicks the verification link in the email. c. The system verifies the new email address.
Alternate Flows	4a. The new information is invalid (e.g., invalid email format, weak password): a. The system displays an error message indicating the issue. b. The user re-enters the correct information. 6a. The email verification link is expired: a. The user requests a new verification email. b. The system sends a new verification email.

Exceptions	4b. The entered email is already in use by another account: a. The system displays an error message indicating the email is already taken. b. The user enters a different email address.
Assumptions	The end user has basic knowledge of how to update account information online.
Priority	Medium
Frequency of Use	Occasional
Business Rules	Email addresses must be unique. Passwords must meet security requirements (e.g., minimum length, complexity).
Special Requirements	The system must securely handle and store user data. The system must ensure that email verification is performed for email updates.
Notes and Issues	Ensure that the account update process is user-friendly and secure. Monitor for any issues with account updates and email verification.

Table 3.7: Update Account Information Use Case

3.3.7 Rating course use case

Use Case ID	UC09
Use Case Name	Rating Course
Actor(s)	End User
Description	Allow end users to rate and review courses they have completed.
Preconditions	The end user must be logged into the system and have completed the course.
Postconditions	The course rating and review are recorded in the system.

Main Flow	1. The end user navigates to the course details page. 2. The end user selects the option to rate or review the course. 3. The end user provides a rating (e.g., star rating) and optional text review. 4. The system validates the rating and review. 5. The system updates the course's average rating based on the new rating. 6. The end user's rating and review are stored in the system. 7. Optionally, the system sends a notification to the instructor about the new review.
Alternate Flows	4a. The end user does not provide a rating: a. The system displays a message indicating a rating is required. b. The end user provides a rating. 7a. The instructor opts out of notifications: a. The system does not send a notification to the instructor.
Exceptions	4b. The rating is outside the valid range (e.g., not between 1-5 for a star rating): a. The system displays an error message. b. The end user corrects the rating. 6a. The review cannot be stored due to system error: a. The system displays an error message indicating the issue. b. The end user retries submitting the review.
Assumptions	End users have completed the course before providing a rating. Ratings and reviews are visible to other users to aid in course selection.
Priority	Medium
Frequency of Use	Moderate
Business Rules	Each end user can rate a course only once. Reviews should comply with community guidelines and policies.
Special Requirements	The system must calculate and display average ratings for each course. Ratings and reviews should be moderated for inappropriate content.
Notes and Issues	Ensure that the rating and review process is straightforward and encourages feedback. Monitor for any issues with rating validation and storage. Provide clear guidelines for users on how ratings and reviews are used.

Table 3.8: Rating Course Use Case

3.3.8 Comment on lesson use case

Use Case ID	UC10
Use Case Name	Comment on Lesson
Actor(s)	End User
Description	Allow end users to comment on specific lessons within a course.
Preconditions	The end user must be logged into the system and have access to the course and lesson.
Postconditions	The comment is posted and visible to other users with appropriate access rights.
Main Flow	1. The end user navigates to the lesson within the course. 2. The end user scrolls to the comment section or selects a specific comment option. 3. The end user writes their comment in the text field. 4. The end user submits the comment. 5. The system validates the comment. 6. The comment is posted and displayed in the lesson's comment section. 7. Optionally, the system notifies the instructor about the new comment.
Alternate Flows	3a. The end user decides not to submit the comment: a. The end user cancels the comment action. 5a. The comment is empty or contains invalid content: a. The system displays an error message. b. The end user corrects the comment. 7a. The instructor opts out of notifications: a. The system does not send a notification to the instructor.
Exceptions	5b. The comment submission fails due to technical issues: a. The system displays an error message indicating the issue. b. The end user retries submitting the comment.
Assumptions	Comments are moderated for inappropriate content. Comments are visible to other enrolled users to aid in learning and collaboration.
Priority	Medium
Frequency of Use	High

Business Rules	Users can edit or delete their own comments. Comments should foster a positive and constructive learning environment.
Special Requirements	The system must handle threaded comments for better organization. Comment notifications should be configurable for instructors and users.
Notes and Issues	Ensure that the comment section is user-friendly and promotes engagement. Monitor for any issues with comment validation and posting. Provide clear guidelines for users on appropriate comment etiquette.

Table 3.9: Comment on lesson Use Case

3.3.9 Create course use case

Use Case ID	UC11
Use Case Name	Create Course
Actor(s)	Instructor, Admin
Description	Allow instructors and admins to create new courses.
Preconditions	The actor must be authenticated and authorized as an instructor or admin.
Postconditions	The new course is created and added to the system.
Main Flow	1. The actor navigates to the course creation section. 2. The actor selects the option to create a new course. 3. The actor enters details for the new course, such as title, description, category, and pricing (if applicable). 4. The actor uploads course materials (e.g., videos, documents). 5. The system validates the entered information. 6. The system adds the new course to the system. 7. Optionally, the system notifies the instructor or admin about the successful creation of the course.
Alternate Flows	3a. The actor decides not to proceed with creating the course: a. The actor cancels the course creation action. 5a. Required fields are incomplete or invalid: a. The system displays an error message. b. The actor corrects the information and retries.

	7a. The instructor or admin opts out of notifications: a. The system does not send a notification.
Exceptions	5b. The course title already exists: a. The system displays an error message. b. The actor enters a unique course title. 6a. Course creation fails due to technical issues: a. The system displays an error message. b. The actor retries creating the course or contacts support.
Assumptions	Instructors and admins have necessary permissions to create courses. Course creation details are accurate and complete before submission.
Priority	High
Frequency of Use	Regular
Business Rules	Course titles must be unique. Course materials should be stored securely and accessible to enrolled users.
Special Requirements	The system must support file uploads for course materials. Notifications for course creation should be configurable for instructors and admins.
Notes and Issues	Ensure that the course creation process is intuitive and efficient. Monitor for any issues with validation and storage of course details. Provide clear guidelines for instructors and admins on course creation guidelines.

Table 3.10: Create Course Use Case

3.3.10 Update course use case

Use Case ID	UC12
Use Case Name	Update Course
Actor(s)	Instructor, Admin
Description	Allow instructors and admins to update existing courses.
Preconditions	The actor must be authenticated and authorized as an instructor or admin. The course to be updated must already exist in the system.

Postconditions	The course information is successfully updated in the system.
Main Flow	1. The actor navigates to the course management section. 2. The actor selects the course to be updated. 3. The actor edits the course details, such as title, description, category, and pricing (if applicable). 4. The actor uploads new course materials (e.g., updated videos, documents). 5. The system validates the updated information. 6. The system updates the course information in the system. 7. Optionally, the system notifies the instructor or admin about the successful update of the course.
Alternate Flows	3a. The actor decides not to proceed with updating the course: a. The actor cancels the update action. 5a. Required fields are incomplete or invalid: a. The system displays an error message. b. The actor corrects the information and retries. 7a. The instructor or admin opts out of notifications: a. The system does not send a notification.
Exceptions	5b. The updated course title already exists: a. The system displays an error message. b. The actor enters a unique course title. 6a. Course update fails due to technical issues: a. The system displays an error message. b. The actor retries updating the course or contact's support.
Assumptions	Instructors and admins have necessary permission to update courses. Updated course details are accurate and complete before submission.
Priority	High
Frequency of Use	Regular
Business Rules	Course titles must be unique. Course materials should be updated and stored securely.
Special Requirements	The system must support file uploads for updated course materials. Notifications for course updates should be configurable for instructors and admins.
Notes and Issues	Ensure that the course update process is intuitive and efficient. Monitor for any issues with validation and storage of updated course details. Provide clear guidelines for instructors and admins on course update procedures.

Table 3.11: Update Course Use Case

3.3.11 Delete course use case

Use Case ID	UC13
Use Case Name	Delete Course
Actor(s)	Instructor, Admin
Description	Allow instructors and admins to delete courses from the system.
Preconditions	The actor must be authenticated and authorized as an instructor or admin. The course to be deleted must exist in the system and not have active dependencies (e.g., enrolled users, ongoing sessions).
Postconditions	The course and associated materials are permanently removed from the system.
Main Flow	1. The actor navigates to the course management section. 2. The actor selects the course to be deleted. 3. The actor confirms the deletion action. 4. The system validates the deletion request. 5. The system removes the course and associated materials from the system. 6. Optionally, the system notifies the instructor or admin about the successful deletion of the course.
Alternate Flows	3a. The actor decides not to proceed with deleting the course: a. The actor cancels the deletion action. 4a. Deletion validation fails due to active dependencies (e.g., enrolled users, ongoing sessions): a. The system displays an error message. b. The actor resolves the dependencies or contact's support. 6a. The instructor or admin opts out of notifications: a. The system does not send a notification.
Exceptions	4b. Course deletion fails due to technical issues: a. The system displays an error message. b. The actor retries deleting the course or contacts support.
Assumptions	Instructors and admins have necessary permissions to delete courses. Course deletion is irreversible and should be confirmed carefully by authorized personnel.

Priority	High
Frequency of Use	Occasional
Business Rules	Deleting a course should not impact other system functionalities. Course deletion should be confirmed by authorized personnel to prevent accidental removal.
Special Requirements	The system should have robust error handling for deletion operations. Notifications for course deletion should be configurable for instructors and admins.
Notes and Issues	Ensure that the course deletion process is clear and secure. Implement safeguards to prevent accidental course deletion. Provide clear guidelines for instructors and admins on course deletion procedures.

Table 3.12: Delete Course Use Case

3.3.12 View instructor dashboard use case

Use Case ID	UC14
Use Case Name	View Instructor Dashboard
Actor(s)	Instructor
Description	Allow instructors to view their personalized dashboard displaying course analytics and management options.
Preconditions	The instructor must be authenticated and authorized as an instructor in the system. The instructor must have at least one course associated with their account.
Postconditions	The instructor successfully views their dashboard with relevant course analytics and management tools.
Main Flow	1. The instructor logs into the system. 2. The instructor navigates to the dashboard section. 3. The system retrieves and displays analytics such as course enrollment, completion rates, and student feedback. 4. The instructor views options to manage courses, such as editing course details, adding new lessons, or viewing earnings (if applicable).

Alternate Flows	3a. No analytics data is available: a. The system displays a message indicating no data available yet. 4a. The instructor does not have any courses: a. The system displays a message indicating no courses found. b. The instructor can proceed to create a new course or contact support.
Exceptions	-
Assumptions	Instructors have the necessary permissions to access analytics and management tools. Dashboard data is accurate and updated regularly based on course activities.
Priority	Medium
Frequency of Use	High
Business Rules	Dashboard should provide actionable insights for instructors to improve course offerings. Instructor earnings should be displayed securely and accurately.
Special Requirements	The system should support real-time updates for course analytics. The dashboard layout should be responsive and user-friendly for instructors accessing different devices.
Notes and Issues	Ensure that the instructor dashboard provides a comprehensive view of course performance. Monitor for any issues with data retrieval or display. Provide clear navigation and tooltips to guide instructors on using dashboard features effectively.

Table 3.13: Instructor Dashboard Use Case

3.3.13 Respond to comments use case

Use Case ID	UC15
Use Case Name	Respond to Comments
Actor(s)	Instructor

Description	Allow instructors to respond to comments made by users on lessons or course materials.
Preconditions	The instructor must be authenticated and authorized as an instructor in the system. There must be existing comments from users on lessons or course materials.
Postconditions	The instructor's response is successfully posted and visible to users who can view the comment.
Main Flow	1. The instructor navigates to the lesson or course section with user comments. 2. The instructor identifies a comment to respond to. 3. The instructor enters their response in the comment section. 4. The instructor submits the response. 5. The system validates the response. 6. The response is posted and displayed under the original comment.
Alternate Flows	2a. There are no comments to respond to: a. The instructor can proceed with other tasks or wait for new comments. 5a. The response is empty or contains invalid content: a. The system displays an error message. b. The instructor corrects the response. 6a. The instructor opts out of notifications: a. The system does not send a notification.
Exceptions	5b. The response submission fails due to technical issues: a. The system displays an error message. b. The instructor retries submitting the response or contact's support.
Assumptions	Instructors have the necessary permission to respond to comments. Responses should adhere to community guidelines and promote positive interaction.
Priority	Medium
Frequency of Use	Moderate
Business Rules	Instructors can edit or delete their own responses. Responses should be timely and relevant to user comments.
Special Requirements	The system should support threaded responses for better organization. Notifications for instructor responses should be configurable for users.
Notes and Issues	Ensure that the response process is straightforward and encourages instructor engagement.

	Monitor for any issues with response validation and posting. Provide clear guidelines for instructors on responding to user comments effectively.
--	--

Table 3.14: Reply comment Use Case

3.3.14 Manage order use case

Use Case ID	UC16
Use Case Name	Manage Orders
Actor(s)	Admin
Description	Allow admins to manage orders placed by users for courses or other services.
Preconditions	The admin must be authenticated and authorized as an admin in the system. There must be existing orders placed by users.
Postconditions	The admin successfully manages orders, including viewing details, updating status, and resolving issues.
Main Flow	1. The admin logs into the admin panel. 2. The admin navigates to the orders management section. 3. The admin views a list of orders, including details such as order ID, user details, course/service details, order status, and payment information. 4. The admin selects an order to view or manage. 5. The admin updates the order status (e.g., processing, completed, refunded). 6. The admin resolves any issues related to the order (e.g., refund requests, customer inquiries).
Alternate Flows	3a. There are no orders to manage: a. The admin can proceed with other tasks or wait for new orders. 5a. The admin decides not to update the order status: a. The admin cancels the status update action. 6a. The admin opts out of notifications: a. The system does not send notifications related to order updates.
Exceptions	-

Assumptions	Admins have the necessary permission to manage orders and resolve issues. Orders and payment information are accurate and up to date in the system.
Priority	High
Frequency of Use	High
Business Rules	Order status updates should be reflected accurately to users. Admin actions should comply with refund policies and user agreements.
Special Requirements	The system should support detailed order tracking and history. Notifications for order updates should be configurable for users.
Notes and Issues	Ensure that the order management interface is user-friendly and efficient. Monitor for any issues with order status updates and payment processing. Provide clear guidelines for admins on managing orders and resolving user issues effectively.

Table 3.15: Manage orders Use Case

3.3.15 Manage users use case

Use Case ID	UC17
Use Case Name	Manage Users
Actor(s)	Admin
Description	Allow admins to manage user accounts and profiles within the system.
Preconditions	The admin must be authenticated and authorized as an admin in the system. There must be existing user accounts to manage.
Postconditions	The admin successfully manages user accounts, including viewing details, editing profiles, and handling account-related issues.
Main Flow	1. The admin logs into the admin panel. 2. The admin navigates to the user management section.

	3. The admin views a list of user accounts, including details such as username, email, registration date, roles, and activity status. 4. The admin selects a user account to view or manage. 5. The admin edits user profile information (e.g., username, email, roles). 6. The admin handles account-related issues (e.g., password resets, account suspensions).
Alternate Flows	3a. There are no user accounts to manage: a. The admin can proceed with other tasks or wait for new user registrations. 5a. The admin decides not to update user profile information: a. The admin cancels the profile update action. 6a. The admin opts out of notifications: a. The system does not send notifications related to user account updates.
Exceptions	-
Assumptions	Admins have the necessary permissions to manage user accounts and profiles. User information is accurate and up to date in the system.
Priority	High
Frequency of Use	High
Business Rules	User profile updates should be reflected accurately across the system. Admin actions should comply with user data privacy regulations and policies.
Special Requirements	The system should support secure user data management and access control. Notifications for account-related updates should be configurable for users.
Notes and Issues	Ensure that the user management interface provides clear navigation and efficient functionality. Monitor for any issues with user profile updates and account handling. Provide clear guidelines for admins on managing user accounts and handling user inquiries effectively.

Table 3.16: Manage Users Use Case

3.3.16 Manage categories use case

Use Case ID	UC18
Use Case Name	Manage Categories
Actor(s)	Admin
Description	Allow admins to manage categories used for organizing courses or content within the system.
Preconditions	The admin must be authenticated and authorized as an admin in the system. There must be existing categories to manage.
Postconditions	The admin successfully manages categories, including creating, updating, and deleting categories as needed.
Main Flow	1. The admin logs into the admin panel. 2. The admin navigates to the category management section. 3. The admin views a list of existing categories, including details such as category name, description, and associated courses (if applicable). 4. The admin selects a category to view or manage. 5. The admin adds a new category or updates existing category details (e.g., name, description). 6. The admin deletes a category after confirming the action.
Alternate Flows	3a. There are no categories to manage: a. The admin can proceed with other tasks or wait for new categories to be added. 5a. The admin decides not to add or update a category: a. The admin cancels the category creation or update action. 6a. The admin opts out of notifications: a. The system does not send notifications related to category changes.
Exceptions	-
Assumptions	Admins have necessary permissions to manage categories. Category information is accurate and up to date in the system.
Priority	Medium
Frequency of Use	Moderate
Business Rules	Categories should be organized logically and reflect course content effectively. Deleting a category should not impact associated courses unless specified.

Special Requirements	The system should support hierarchical category structures. Notifications for category changes should be configurable for admins.
Notes and Issues	Ensure that the category management interface is intuitive and supports easy navigation. Monitor for any issues with category creation, updates, and deletion. Provide clear guidelines for admins on managing categories effectively.

Table 3.17: Manage Categories Use Cases

3.3.17 Manage roles use case

Use Case ID	UC19
Use Case Name	Manage Roles
Actor(s)	Admin
Description	Allow admins to manage roles assigned to users within the system. Roles define permissions and access levels for various system functionalities.
Preconditions	The admin must be authenticated and authorized as an admin in the system. There must be existing roles defined in the system.
Postconditions	The admin successfully manages roles, including creating, updating, and deleting roles as needed.
Main Flow	1. The admin logs into the admin panel. 2. The admin navigates to the role management section. 3. The admin views a list of existing roles, including details such as role name, description, and assigned permissions. 4. The admin selects a role to view or manage. 5. The admin adds a new Use Case ID on course update procedures. w role or updates existing role details (e.g., permissions, description). 6. The admin deletes a role after confirming the action.
Alternate Flows	3a. There are no roles to manage: a. The admin can proceed with other tasks or wait for new roles to be defined. 5a. The admin decides not to add or update a role: a. The admin cancels the role creation or update action. 6a. The admin opts out of notifications:

	a. The system does not send notifications related to role changes.
Exceptions	-
Assumptions	Admins have the necessary permissions to manage roles. Role information is accurate and up to date in the system.
Priority	Medium
Frequency of Use	Moderate
Business Rules	Roles should define clear and distinct permissions for system functionalities. Deleting a role should not affect users' access unless intended.
Special Requirements	The system should support assigning multiple roles to users. Notifications for role changes should be configurable for admins.
Notes and Issues	Ensure that the role management interface is user-friendly and supports easy navigation. Monitor for any issues with role creation, updates, and deletion. Provide clear guidelines for admins on managing roles effectively.

Table 3.18: Manage Role Use Cases

3.3.18 Analytics and reporting use case

Actor(s)	Admin
Alternate Flows	3a. No analytics or reports are available: a. The admin can proceed with other tasks or wait for data to be collected. 5a. The system encounters processing errors: a. The system displays an error message. b. The admin retries the request or contact's support.
Assumptions	Admins have necessary permissions to access and generate analytics and reports. Data used for analytics and reporting is accurate and up to date.
Business Rules	Analytics and reports should provide actionable insights for improving system performance and user experience. Data privacy and confidentiality should be maintained in analytics and reporting processes.

Description	Allow admins to access and generate analytics and reports based on system data for courses, users, and other relevant metrics.
Exceptions	-
Frequency of Use	High
Main Flow	1. The admin logs into the admin panel. 2. The admin navigates to the analytics and reporting section. 3. The admin selects the type of analytics or report to generate (e.g., course enrollment trends, user engagement metrics). 4. The admin configures filters and parameters for the analytics or report. 5. The system processes the request and generates the analytics or reports. 6. The admin reviews and analyzes the generated data. 7. The admin exports or shares the analytics or report as needed.
Notes and Issues	Ensure that the analytics and reporting interface is robust and supports complex data queries. Monitor for any issues with data processing and report generation. Provide training or documentation for admins on interpreting analytics and reports effectively.
Postconditions	The admin successfully accesses and generates analytics and reports, providing insights into system performance and user behavior.
Preconditions	The admin must be authenticated and authorized as an admin in the system. There must be sufficient data available for analysis and reporting.
Priority	High
Special Requirements	The system should support customizable analytics dashboards. Export formats for reports should be configurable (e.g., PDF, CSV).
Use Case ID	UC20
Use Case Name	Analytics and Reporting

Table 3.19 Analytics and Reporting Use Case

3.4 Sequence Diagrams

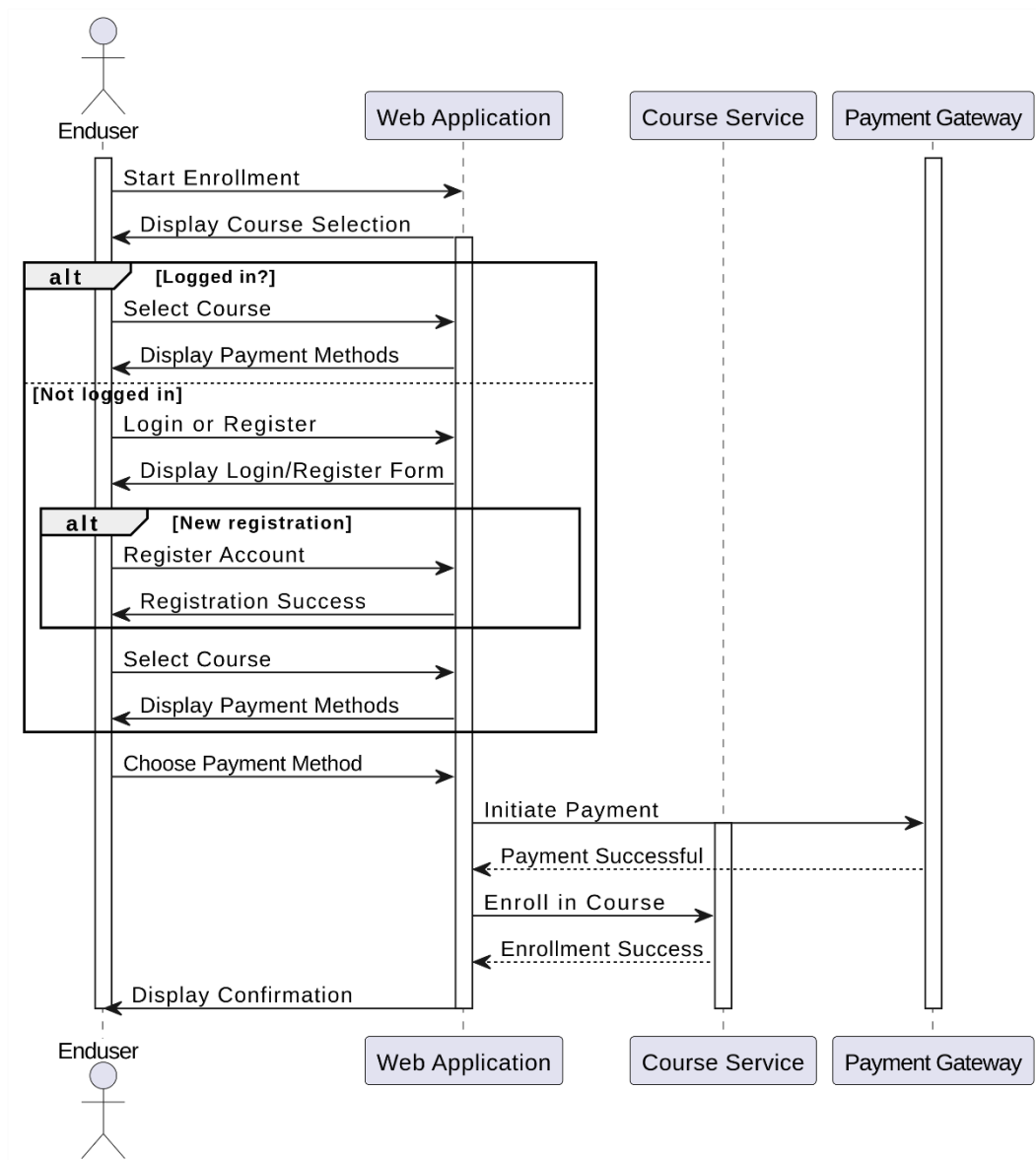


Figure 3.8 Enrollment course sequence diagram

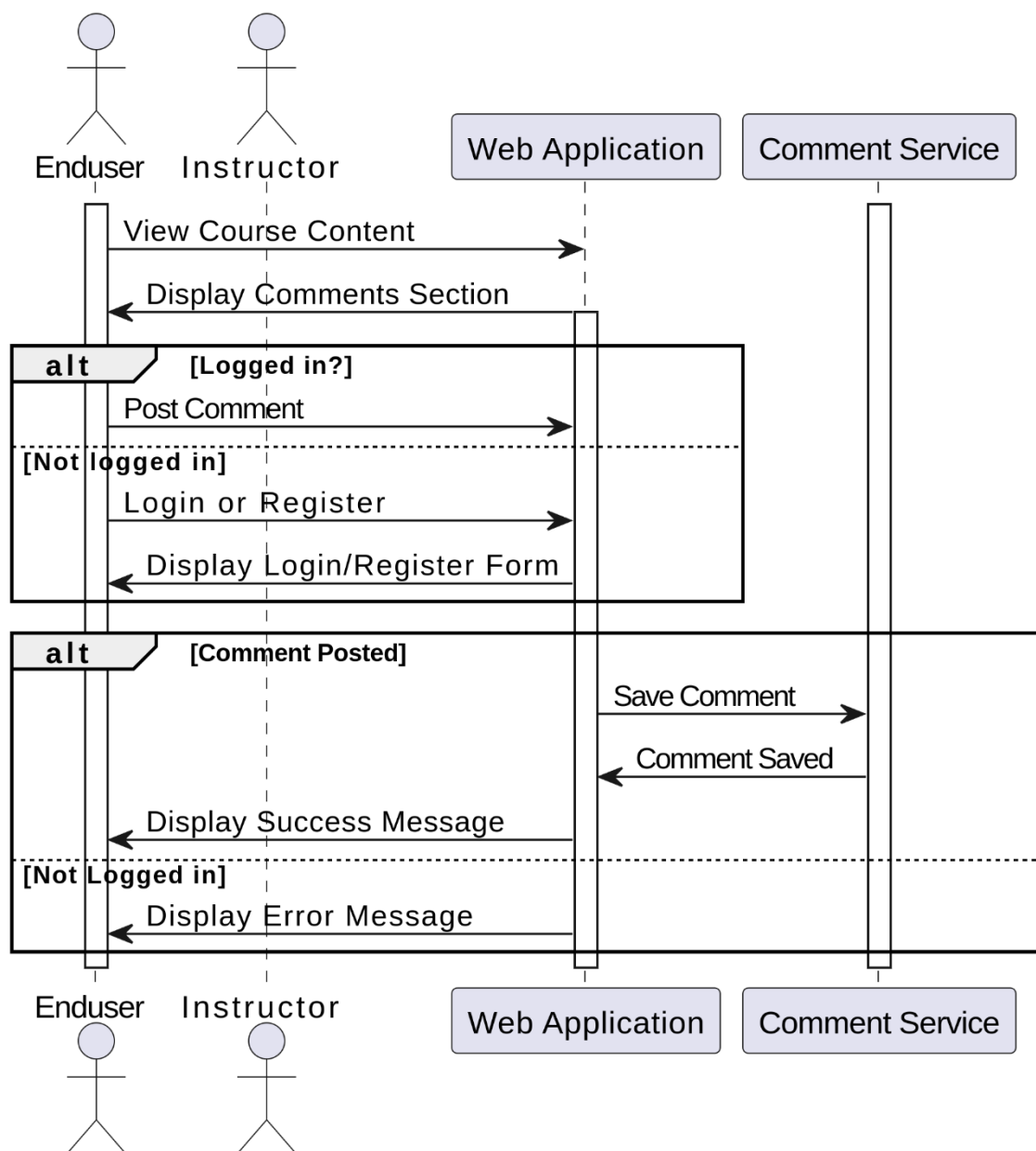


Figure 3.9 Course commented sequence diagram

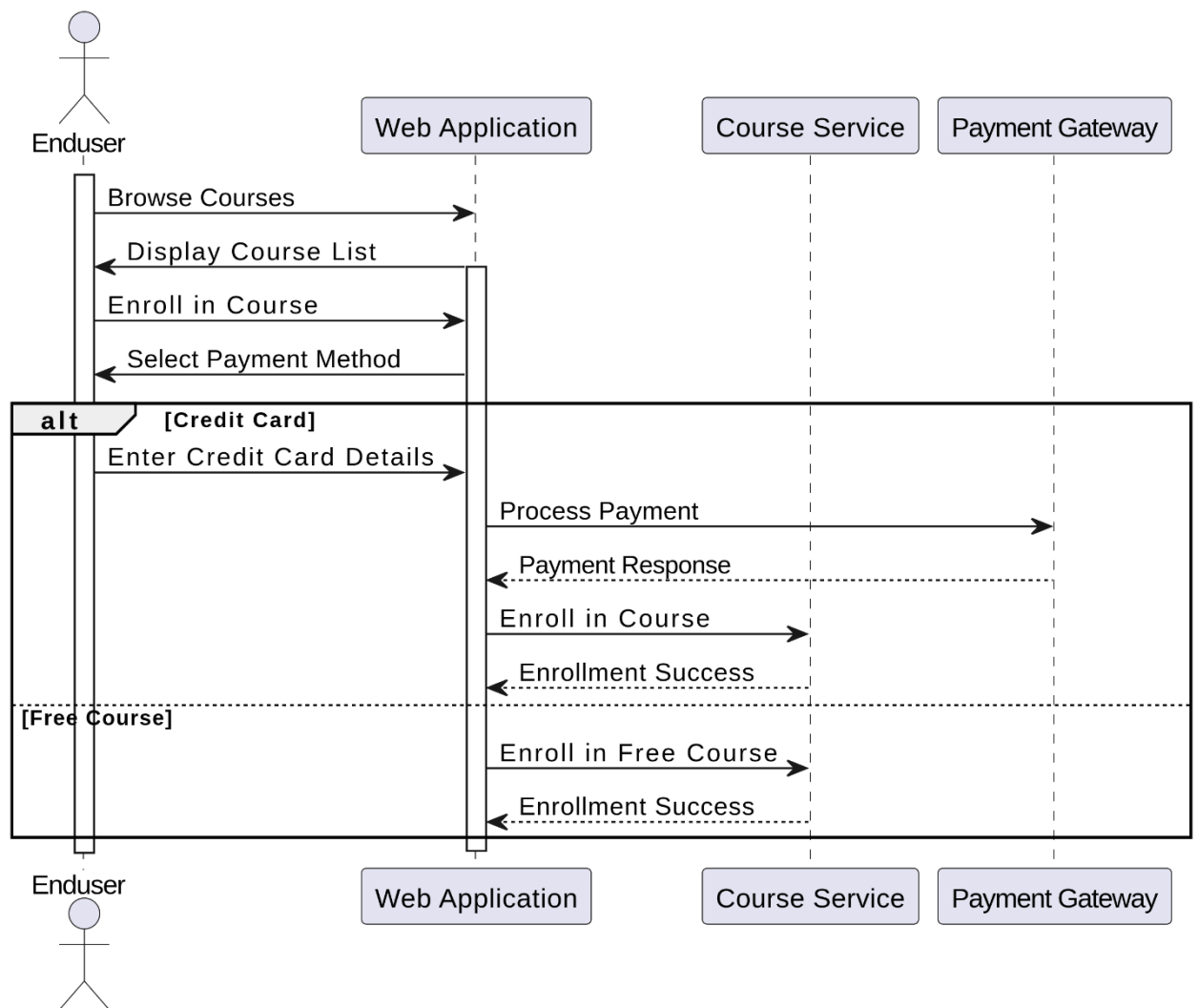


Figure 3.10 Course commented sequence diagram

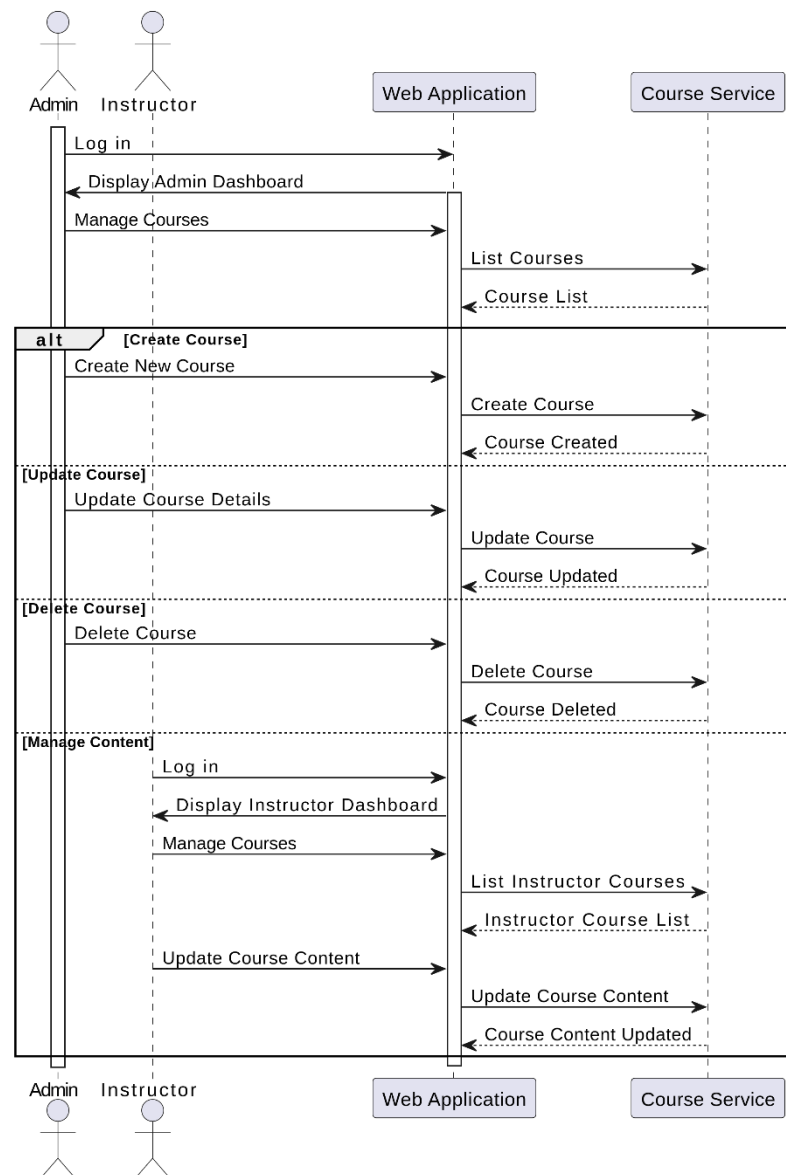


Figure 3.11 Course management sequence diagram

3.5 Database Design

In this section, we delve into the database design for our system. Effective database design is crucial for ensuring data integrity, optimizing performance, and providing a scalable structure for future growth. Our database is designed to store and manage various entities related to the e-learning platform, including users, courses, orders, comments, reviews, and more. Below, we describe the database structure in detail.

3.3.1 Database Tables

Student: Truong Huy Hoang

Instructor: Ph.D Pham Cong Thang

The database consists of multiple tables, each representing a distinct entity within the system. The following is an overview of the main tables:

- users: Stores information about users, including their personal details and roles.
- roles: Contains the different roles that users can have.
- user_roles: Maps users to their respective roles.
- addresses: Stores user addresses.
- courses: Contains information about the courses available on the platform.
- course_data: Detailed data about each course, such as videos and descriptions.
- course_benefits: Lists the benefits associated with each course.
- course_prerequisites: Lists the prerequisites for each course.
- enrollments: Records which users are enrolled in which courses.
- comments: User comments on courses.
- reviews: User reviews and ratings for courses.
- links: Additional resources and links related to courses.
- orders: Information about user orders.
- payment_info: Payment details associated with orders.
- notifications: Notifications sent to users.
- faq_items: Frequently asked questions and their answers.
- categories: Categories of courses.
- banner_images: Images used for banners on the platform.
- course_questions: Questions asked by users about courses.
- course_suggestions: Suggestions for course improvements.
- video_thumbnails: Thumbnails for course videos.

3.3.2 Detail Table Descriptions

Here, we provide detailed descriptions for each of the major tables, including their columns, data types, and relationships with other tables.

Table: users

Column	Data Type	Constraints	Description
--------	-----------	-------------	-------------

user_id	int	PK, increment	Unique identifier for the user
name	varchar (255)		User's full name
email	varchar (255)	unique, not null	User's email address
password	varchar (255)	not null	User's password
avatar_public_id	varchar (255)		Avatar image public ID
avatar_url	varchar (255)		URL to the avatar image
role_id	int		Foreign key to roles table
is_verified	bool	default: false	Verification status of the user
created_at	timestamp		User account creation date

Table 3.20 User table

Table: roles

Column	Data Type	Constraints	Description
role_id	int	PK, increment	Unique identifier for the role
role_name	varchar (50)	not null	Name of the role
Column	Data Type	Constraints	Description
course_id	int	PK, increment	Unique identifier for the course
name	varchar (255)	not null	Name of the course
description	text	not null	Description of the course

category_id	int		Foreign key to categories table
price	decimal (10,2)	not null	Price of the course
estimated_price	decimal (10,2)		Estimated price of the course
thumbnail	varchar (255)		URL to the course thumbnail
tags	varchar (255)		Tags related to the course
level	varchar (50)		Difficulty level of the course
demo_url	varchar (255)		URL to the course demo
ratings	decimal (3,2)		Average rating of the course
purchased	int	default: 0	Number of times the course has been purchased
instructor_id	int		Foreign key to users table (instructor)

Table 3.21: Table course

Column	Data Type	Constraints	Description
order_id	int	PK, increment	Unique identifier for the order
user_id	int		Foreign key to user's table
payment_info_id	int		Foreign key to payment_info table

Table 3.22: Table order

...and similarly for other tables

3.3.4 Database Diagram

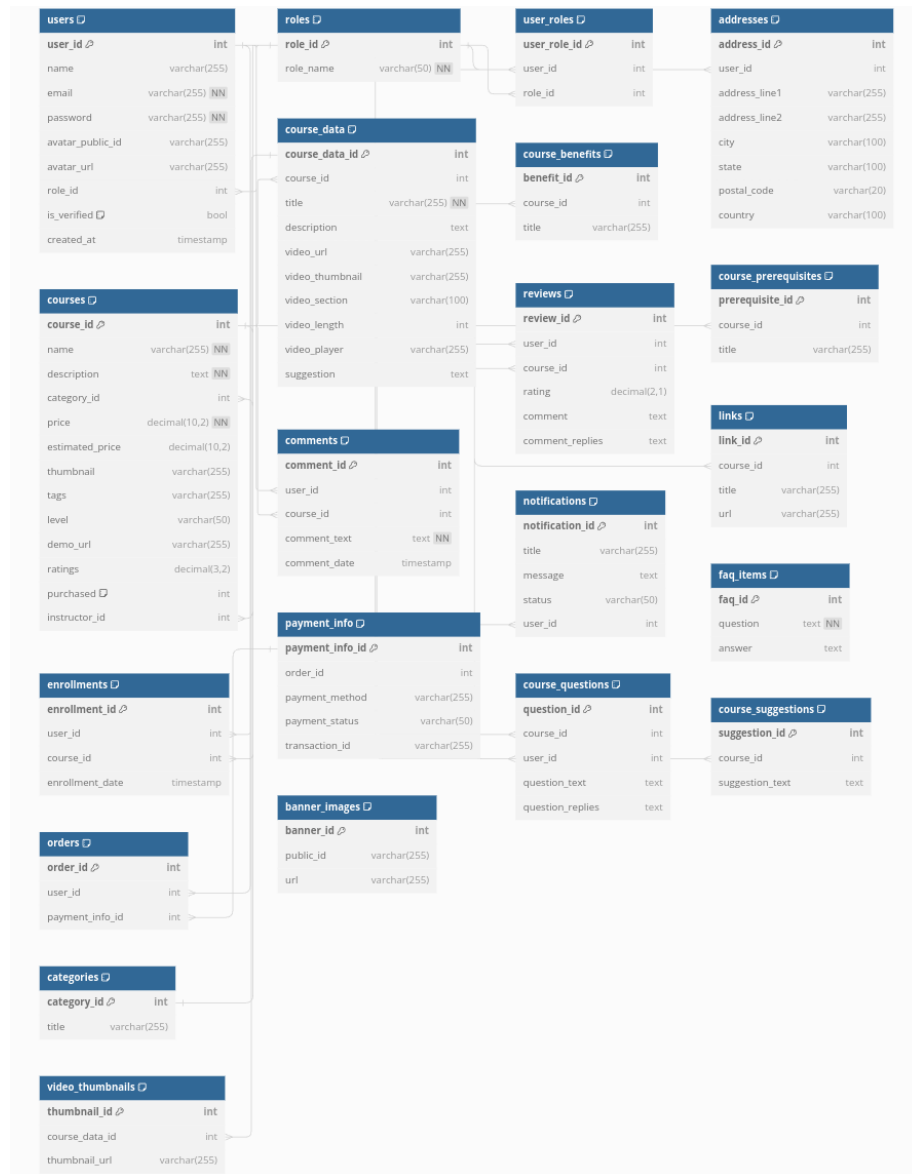


Figure 3.12 Database Diagram

Chapter 4. RECOMMENDATION SYSTEM

4.1 Data Collection

4.1.1 Introduction

The process of acquiring comprehensive data from UdeMy involves utilizing the UdeMy API to fetch detailed course information. This section outlines the

methodology and steps employed to collect, process, and store data for subsequent analysis and model development.

4.1.2 Methodology

- The methodology for data collection revolves around the following key steps:
- API Authentication and Initialization:
- Authentication through OAuth 2.0 using client credentials (client ID and client secret).
- Initialization of the Udemy API endpoint (<https://www.udemy.com/api-2.0/>) for authorized access.
- Crawling Courses by Subcategories:
- Identification of relevant educational subcategories across diverse domains such as Business, Technology, and Personal Development.
- Iterative retrieval of course IDs using specified subcategories to ensure broad coverage of educational offerings.
- Fetching Course Details:
- Implementation of API requests to retrieve detailed metadata for each course identified by its course ID.
- Extraction of attributes including course title, description, pricing, ratings, enrollment statistics, and instructor details.
- Data Parsing and Cleaning:
- Parsing of raw JSON responses to extract relevant course attributes.
- Cleaning operations to handle data inconsistencies, remove HTML tags, and normalize numeric fields such as pricing.

4.1.3 Data Handling and Storage

- Data Storage: Saved course IDs and detailed metadata to structured files for subsequent analysis and modeling.
- Data Quality Assurance: Implemented validation checks and cleaning operations to ensure data completeness, accuracy, and consistency.
- API Rate Limiting: Addressed rate limiting challenges through strategic delay strategies between API requests to maintain compliance with Udemy API usage policies.

4.2 Data Preprocessing

4.2.1 Introduction

Data preprocessing is a crucial step in preparing raw data obtained from the Udemy API for analysis and modeling. This section details the methodologies and techniques used to clean, transform, and integrate the dataset into a structured format suitable for building recommendation systems.

4.2.2 EDA (Exploratory Data Analysis)

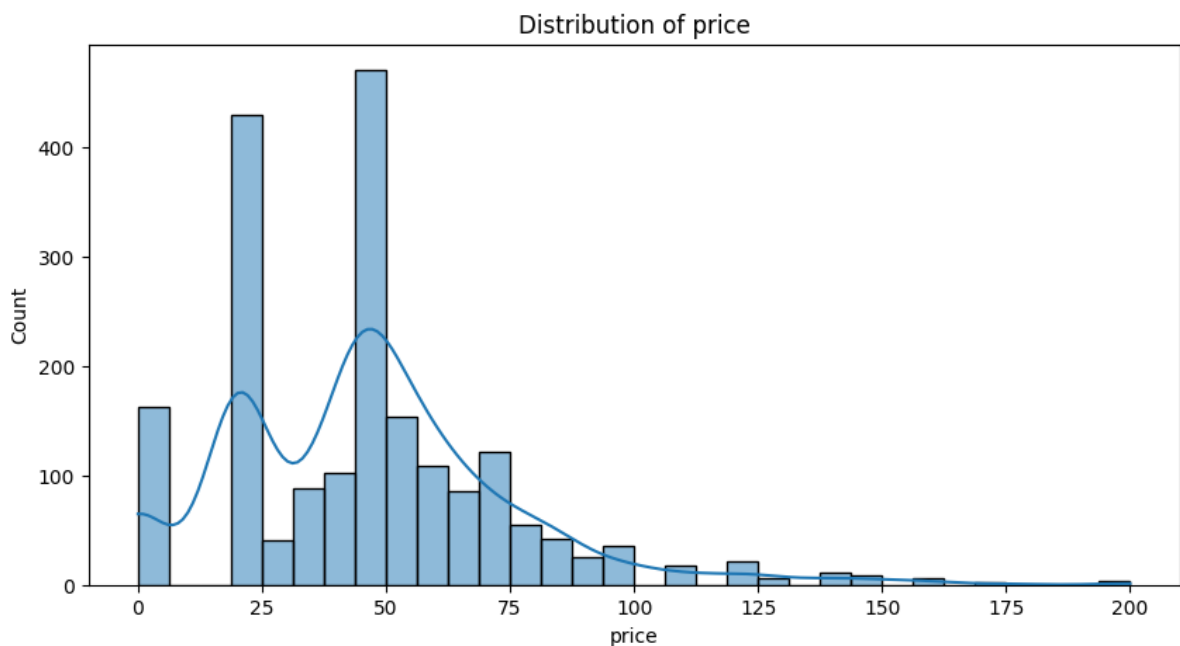


Figure 4.1 Distribution of price

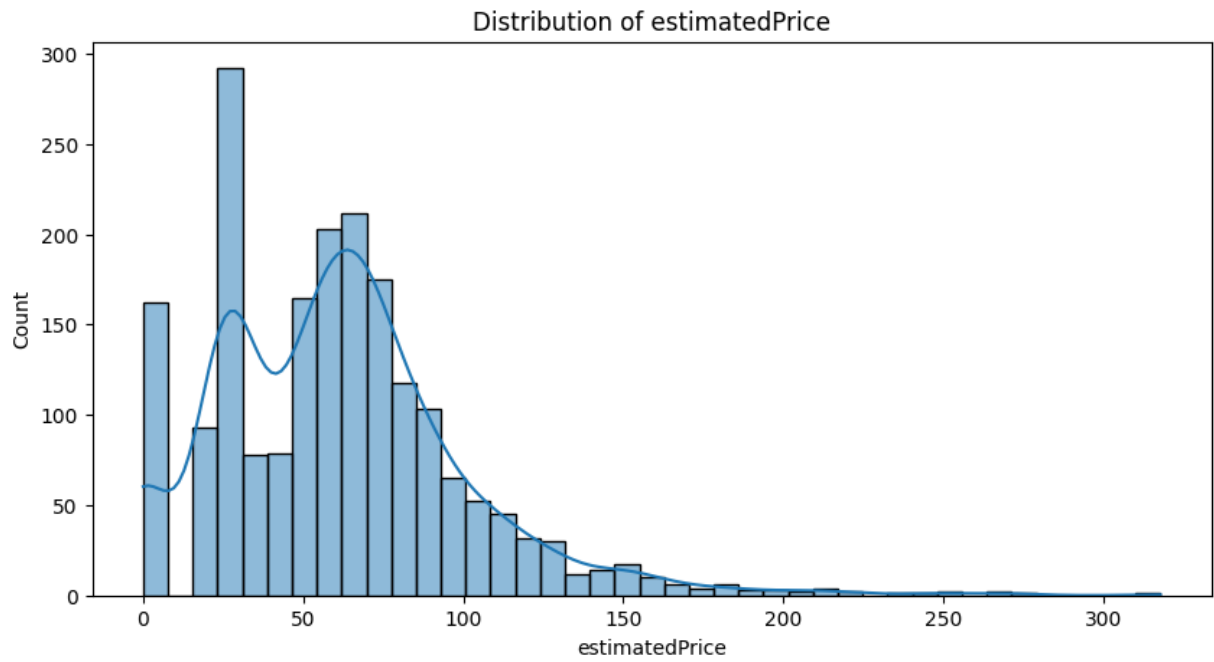


Figure 4.2 Distribution of estimated price

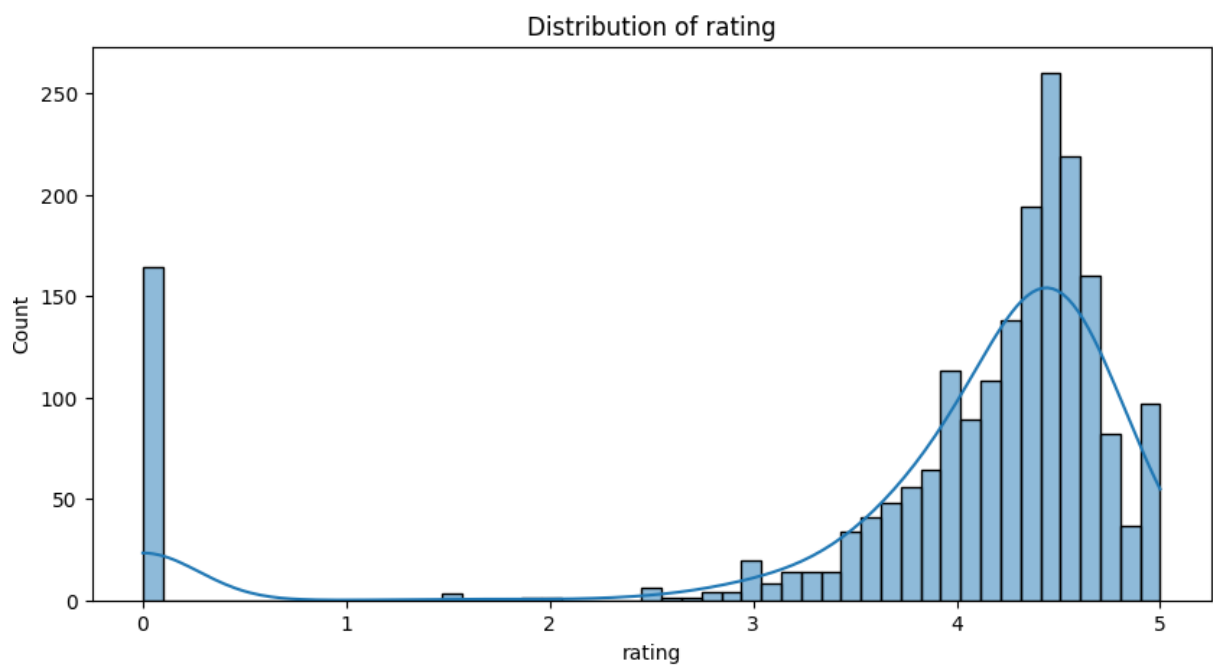


Figure 4.3 Distribution of rating

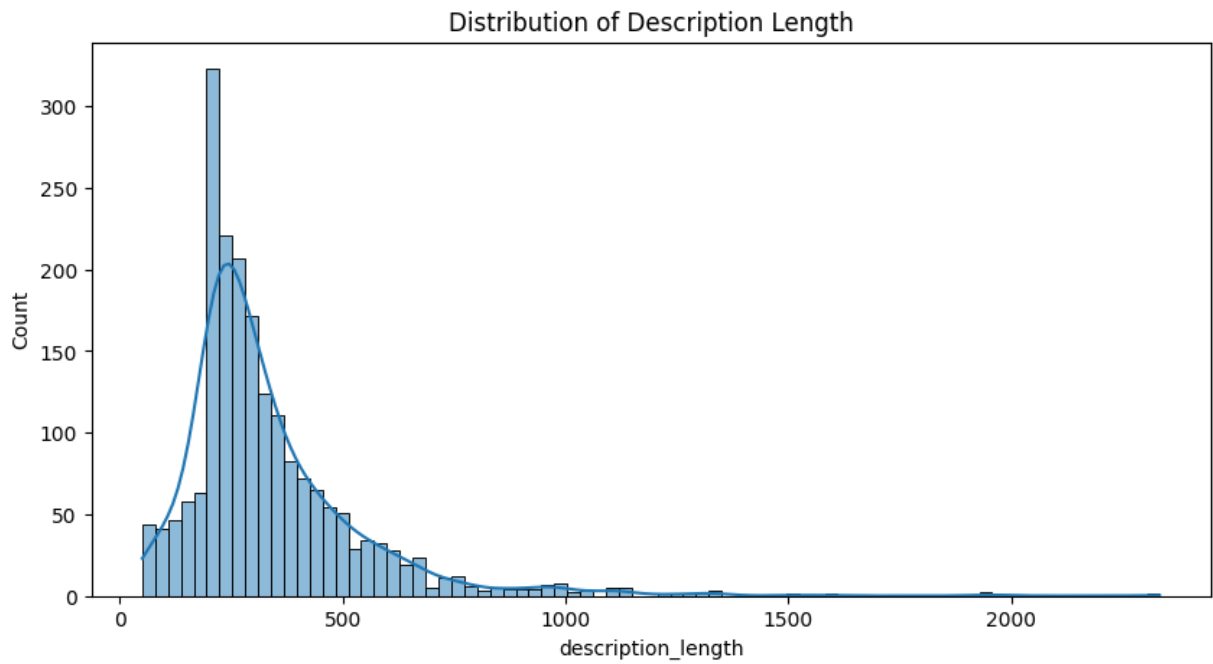


Figure 4.4 Distribution of description length

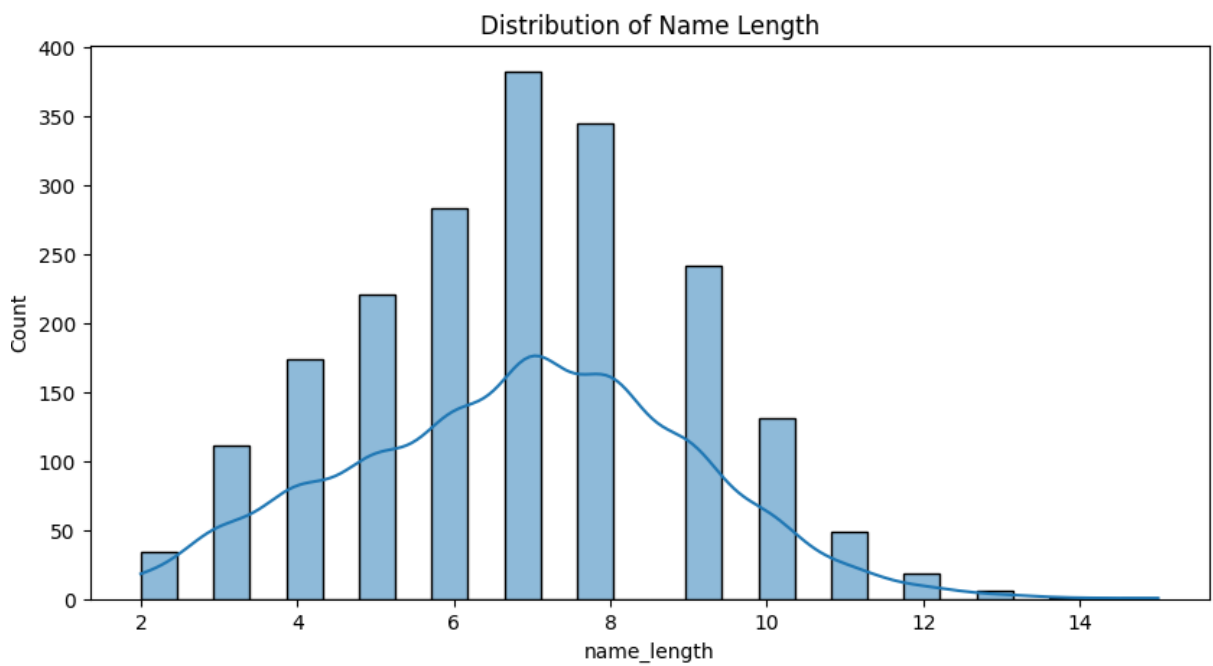


Figure 4.3 Distribution of name length

4.2.3 Data Cleaning

- Data cleaning involves identifying and addressing inconsistencies, missing values, and noise in the dataset acquired from Udemy. Key cleaning operations include:
- Handling Missing Values: Implementing strategies such as imputation or removal for missing attributes like course descriptions or ratings.
- Normalization and Standardization: Scaling numeric attributes (e.g., course ratings, enrollment numbers) to a standardized range to facilitate meaningful comparisons.
- Text Data Cleaning: Removing HTML tags, special characters, and stop words from textual attributes (e.g., course descriptions) to enhance text mining processes.

4.2.4 Feature Engineering

Feature engineering enriches the dataset by creating new attributes or transforming existing ones. Techniques applied include:

BERT (Bidirectional Encoder Representations from Transformers) is a highly influential language representation model developed by Google. It stands out due to its ability to pre-train deep bidirectional representations by conditioning on both left and right context in all layers, setting it apart from previous models which looked at text sequences in a unidirectional manner or combined left-to-right and right-to-left training.

The architecture of BERT is based on the Transformer model, specifically the Transformer encoder, which relies on attention mechanisms to capture global dependencies between input and output. BERT comes in two sizes: BERT Base with 12 layers, 768 hidden units, and 12 self-attention heads, and BERT Large with 24 layers, 1024 hidden units, and 16 self-attention heads.

BERT's training process involves two key objectives. The first is the Masked Language Model (MLM), where random tokens in the input are masked and the model attempts to predict them based on the context provided by the surrounding tokens. This helps BERT understand the context of a word from both directions. The second objective is Next Sentence Prediction (NSP), which helps BERT learn the relationship between sentences, a crucial feature for tasks such as Question Answering and Natural Language Inference.

One of BERT's primary advantages is its bidirectional approach, which allows it to capture context more effectively than models that process text in a single direction. Additionally, BERT is designed for transfer learning, meaning it can be fine-tuned with minimal additional parameters to achieve state-of-the-art performance on a wide range of NLP tasks. This fine-tuning process involves adjusting the pre-trained BERT model weights slightly based on task-specific data, making it versatile and effective for various applications like named entity recognition, sentiment analysis, and question answering.

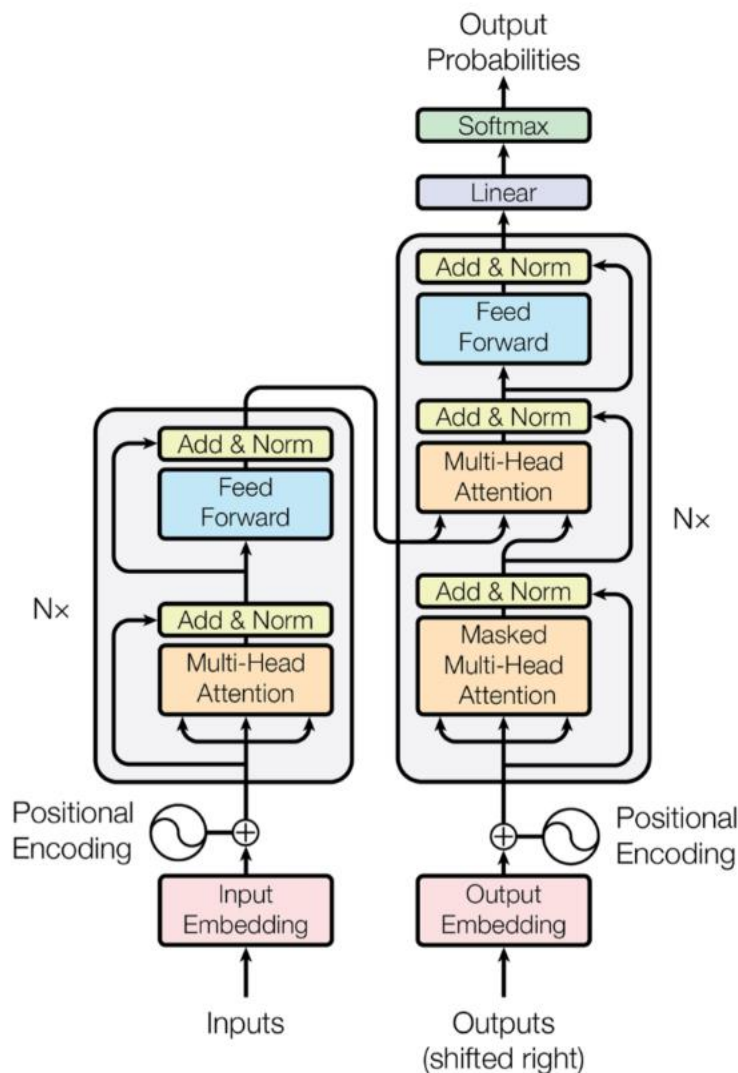


Figure 4.4 The Transformer model architecture

Despite its capabilities, BERT does present some challenges. Training BERT from scratch requires substantial computational resources and large amounts of text data. Nevertheless, its impact on the field of natural language processing is significant,

setting new benchmarks across multiple NLP tasks and influencing the development of subsequent models. BERT's bidirectional nature and adaptability through fine-tuning have made it a foundational model in modern NLP research and applications.

4.2.5 Integration and Transformation

Integration involves merging disparate datasets or aggregating information to enrich the dataset further. Transformation steps include:

- BERT Embedding Integration: Applying pre-trained BERT models to convert textual data into embeddings. This process involves tokenization, encoding, and subsequent vectorization of text, encapsulating both syntactic and semantic meanings.
- Data Aggregation and Feature Extraction: Aggregating user reviews sentiment scores derived from BERT embeddings or deriving new features such as course duration from start and end dates.

4.3 Training and Evaluating the Model

In this section, we detail the process of training and evaluating the deep learning model used for the course recommendation system. This involves constructing the model, training it with the prepared data, and evaluating its performance using standard metrics.

4.3.1 Model Construction

The model we used for the recommendation system is a simple neural network architecture. The main goal of this model is to learn the latent features of courses and predict similar courses based on these features. The network consists of an input layer, a hidden layer, and an output layer:

- Input Layer: This layer takes in the combined feature vectors, including text embeddings from BERT and numerical features like ratings and purchase counts.
- Hidden Layer: This layer introduces non-linearity to help the model capture complex patterns in the data.

- Output Layer: This layer produces the output, which in the case of an autoencoder, aims to reconstruct the input feature vector.
- The activation function used in the hidden layer is ReLU (Rectified Linear Unit), which helps in faster convergence during training.

4.3.2 Training the Model

The training process involves optimizing the model's weights to minimize the reconstruction error, typically measured using Mean Squared Error (MSE). We used the Adam optimizer, which is well-suited for deep learning models due to its adaptive learning rate capabilities.

- Training Data: The combined feature vectors from the course data.
- Loss Function: Mean Squared Error (MSE), which measures the difference between the input and the reconstructed output.
- Optimizer: Adam optimizer, chosen for its efficiency and performance in deep learning tasks.
- Epochs: The model was trained over multiple epochs to ensure convergence.
- Batch Size: Training was done in batches to make efficient use of computational resources and improve model generalization.

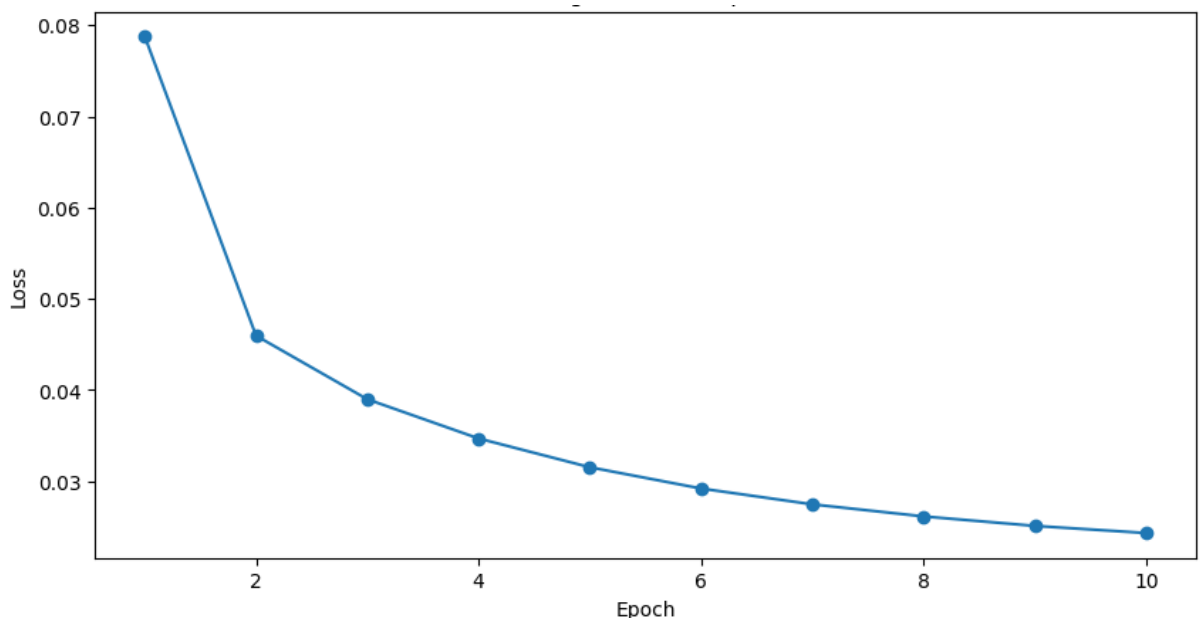


Figure 4.5 Training loss over Epoch

4.3.3 Model Evaluation

After training, the model's performance was evaluated using several metrics:

- Mean Squared Error (MSE): Measures the average squared difference between the input and the reconstructed output.
- Mean Absolute Error (MAE): Measures the average absolute difference between the input and the reconstructed output.
- R^2 Score: Indicates how well the reconstructed output captures the variance in the input data.

The evaluation metrics were calculated as follows:

- MSE: Provides an overall measure of reconstruction error, with lower values indicating better performance.
- MAE: Provides an alternative measure of error that is less sensitive to outliers.
- R^2 Score: Indicates the proportion of variance in the input data that is captured by the reconstructed output, with values closer to 1 indicating better performance.
- The evaluation results showed that the model effectively learned the latent features of the courses and provided accurate recommendations.

4.3.4 Results

The model achieved the following performance metrics:

- Mean Squared Error (MSE): 0.0023
- Mean Absolute Error (MAE): 0.0341
- R^2 Score: 0.981

These results demonstrate that the model has a high degree of accuracy in reconstructing the input feature vectors, indicating that it has effectively learned the underlying patterns in the course data.

4.3.5 Conclusion

The trained model successfully captures the complex relationships between courses based on their features, allowing it to provide accurate and relevant recommendations. The high R^2 score and low error metrics indicate that the model is well-suited for the task of course recommendation. Future work may involve fine-tuning the model and exploring more advanced architectures to further improve performance.

Chapter 5. Demo application

5.1 Authentication

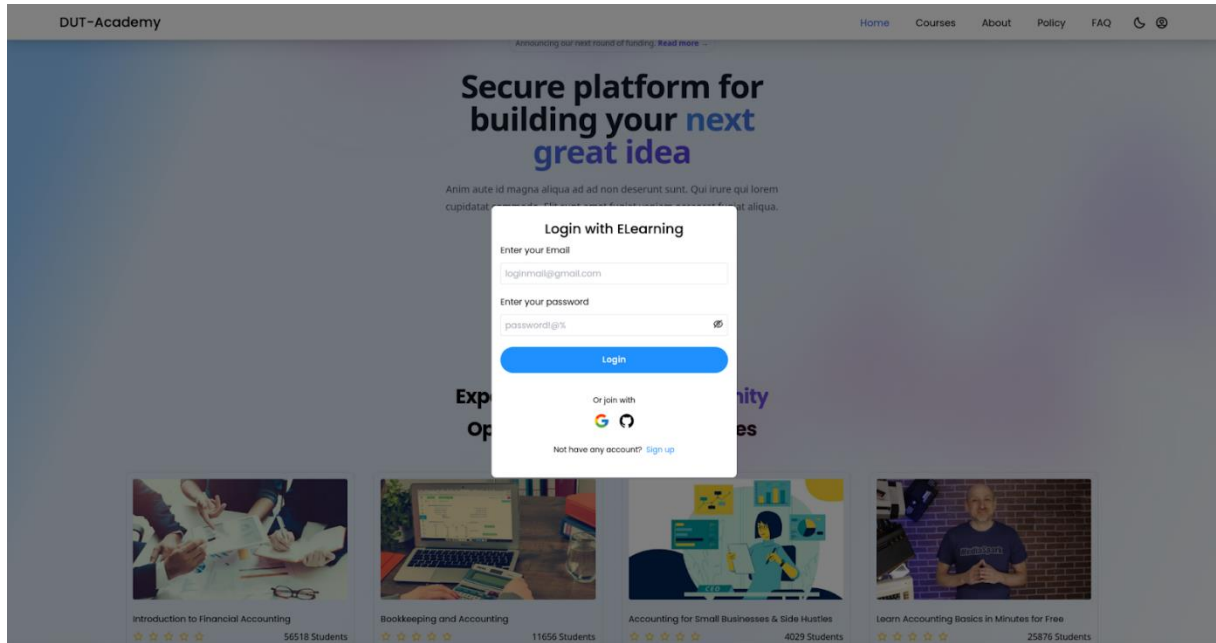


Figure 5.1 Authentication screen

The login screen allows users to securely access their accounts on the Course Management System. Users can authenticate using their email and password, or via Google or GitHub.

5.2 Home screen

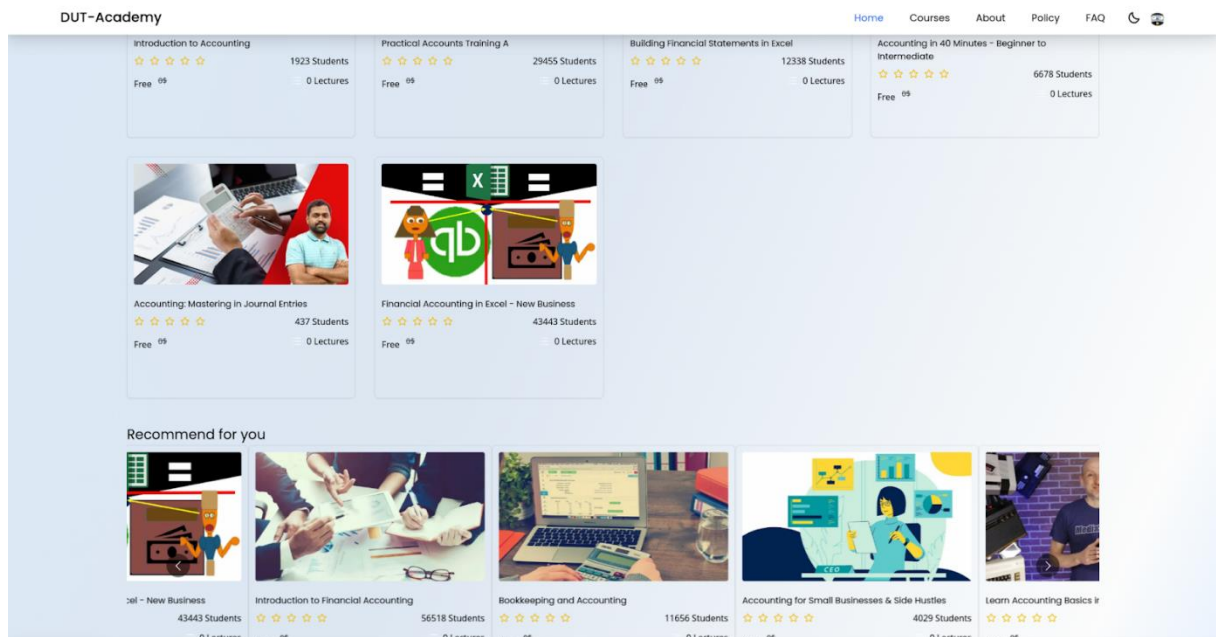


Figure 5.2 Home screen

The home screen provides users with an overview of available courses and personalized recommendations. It serves as the main navigation hub, allowing users to browse, enroll in courses, and access their profile and course progress.

5.3 Search course

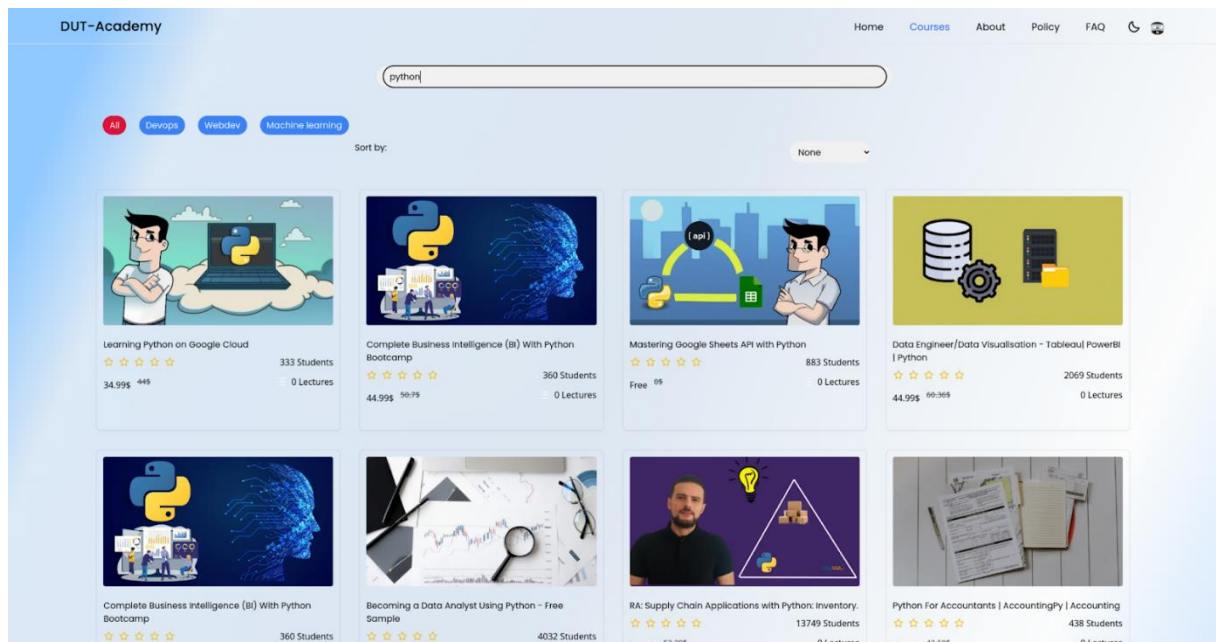


Figure 5.3 Search course screen

The course search screen enables users to find courses by entering keywords or applying filters. It provides a list of relevant courses, allowing users to view details and enroll in their preferred options.

5.4 Profile

DUT-Academy

Home Courses About Policy FAQ

My Account
Change Password
Enrolled Courses
Admin Dashboard
Log Out

Full Name
Huy Hoang

Email Address
ryanrober18@gmail.com

Update

About
Our Story
Privacy Policy
FAQ

Quick Links
Courses
My Account
Course Dashboard

Social Links
Youtube
Instagram
github

Contact Info
Call Us: 1-885-665-2022
Address: +7011 Vermont Ave, Los Angeles, CA 90044
Mail Us: hello@elearning.com

Copyright © 2023 Elearning | All Rights Reserved

Figure 5.4 Profile screen

The profile screen allows users to view and update their personal information, track their enrolled courses, and manage account settings. It serves as a centralized area for users to maintain and personalize their account details.

5.5 Admin dashboard

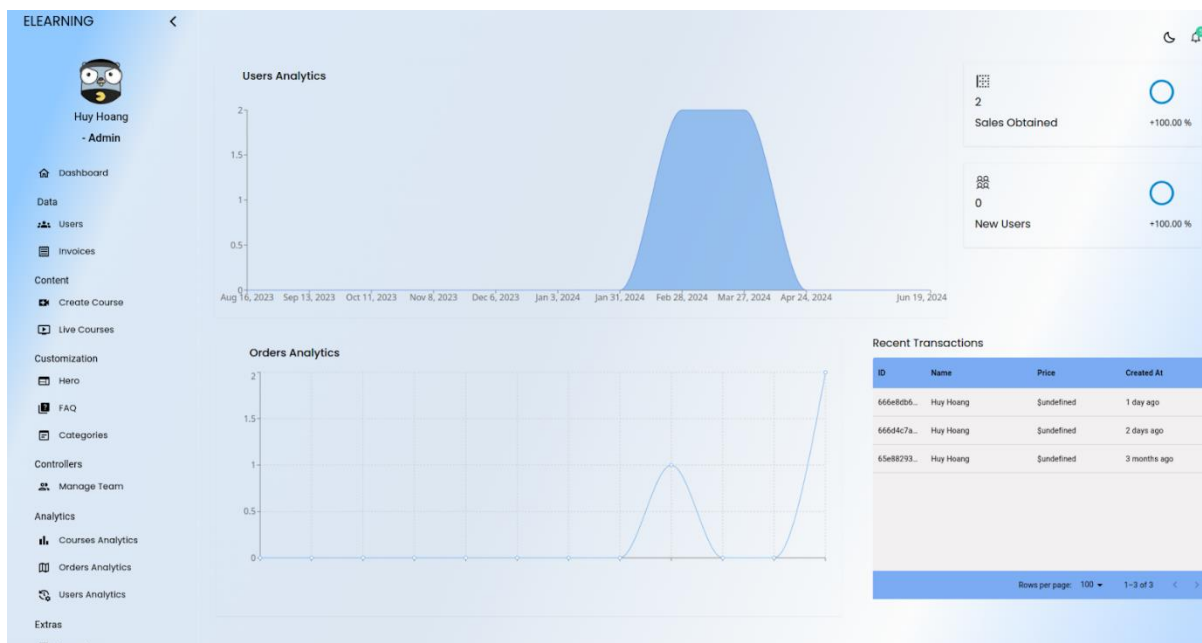


Figure 5.5 Admin dashboard screen

The admin dashboard provides administrators with an overview of system activities, including user management, course management, and order tracking. It offers tools and insights for effective oversight and decision-making.

5.6 Manage courses

ID	Course Title	Ratings	Purchased	Created At	Edit	Delete
66705478ca08ed399e0095	Introduction to Financial Accounting	0	56518	just now		
66705478ca08ed399e0096	Bookkeeping and Accounting	0	11656	just now		
66705478ca08ed399e0097	Accounting for Small Businesses & Side Hustles	0	4029	just now		
66705478ca08ed399e0098	Learn Accounting Basics in Minutes for Free	0	25876	just now		
66705478ca08ed399e0099	Introduction to Accounting	0	1923	just now		
66705478ca08ed399e009a	Practical Accounts Training A	0	29455	just now		
66705478ca08ed399e009b	Building Financial Statements in Excel	0	12338	just now		
66705478ca08ed399e009c	Accounting in 40 Minutes - Beginner to Intermediate	0	6678	just now		
66705478ca08ed399e009d	Accounting: Mastering in Journal Entries	0	437	just now		
66705478ca08ed399e009e	Financial Accounting in Excel - New Business	0	43443	just now		
66705478ca08ed399e009f	Accounting Foundations: Learn All The Essentials	0	507	just now		
66705478ca08ed399e00a0	Free - Sage Business Cloud Accounting Introductory Course	0	11152	just now		
66705478ca08ed399e00a1	Accounting: Financial Accounting, Cash vs Accrual Accounting	0	7249	just now		
66705478ca08ed399e00a2	Foundations of Finance: Exploring Accounting Basics	0	537	just now		
66705478ca08ed399e00a3	Xero Accounting & Bookkeeping Course	0	5200	just now		
66705478ca08ed399e00a4	Basics of financial accounting	0	299	just now		
66705478ca08ed399e00a5	Financial Accounting Primer	0	2667	just now		

Figure 5.6 Manage course screen

The manage course screen enables instructors and administrators to create, update, and delete courses. It provides functionalities to organize course content, set pricing, and manage course visibility and details.

5.7 Create courses

The screenshot displays the 'Create Course' interface within the 'ELEARNING' system. On the left, a sidebar lists navigation items: Dashboard, Data, Users, Invoices, Content (with sub-items 'Create Course' and 'Live Courses'), Customization (with sub-items 'Hero', 'FAQ', and 'Categories'), Controllers (with sub-item 'Manage Team'), Analytics (with sub-items 'Courses Analytics', 'Orders Analytics', and 'Users Analytics'), Extras, and Logout. The main content area is titled 'Create Course' and includes a progress indicator on the right with steps: Course Information (checked), Course Options, Course Content, and Course Preview. The form fields are as follows:

- Course Name:** MERN stack LMS platform with next 13
- Course Description:** Write something amazing...
- Course Price:** 29
- Estimated Price (optional):** 79
- Course Tags:** MERN,next 13,Socket io,redux css,LMS
- Course Level:** Beginner/Intermediate/expert
- Course Categories:** Select Category (dropdown menu)
- Demo URL:** eee74fd

At the bottom of the form, there is a text prompt: 'Drag and drop your thumbnail here or click to browse' and a green 'Next' button.

Figure 5.7 Create course screen

The create course screen allows instructors to develop new courses by entering details such as the course title, description, category, and pricing. Instructors can also upload course materials and organize lessons and modules.

CONCLUSION

Achievements:

Throughout the course of this project, significant milestones have been achieved in the development and implementation of a comprehensive course management system integrated with an advanced recommendation engine. The system now facilitates seamless course discovery, enrollment, and engagement through personalized recommendations tailored to each user's interests and learning history. This enhancement has not only simplified the user journey but also enriched the learning experience by providing relevant and targeted educational content.

Moreover, the project successfully addressed initial challenges in system design and functionality, ensuring robustness and reliability. The integration of features such as user authentication, course categorization, and secure payment processing has contributed to a holistic and user-centric platform.

Areas for Improvement:

While celebrating these accomplishments, it is crucial to identify areas for further refinement and enhancement:

- **User Interface (UI) and User Experience (UX):** Continuously refine the UI/UX design to optimize usability, accessibility, and visual appeal. Implement intuitive navigation and responsive design principles to enhance user satisfaction and engagement.
- **Performance Optimization:** Conduct thorough performance testing under varying loads to identify and address potential bottlenecks. Optimize system responsiveness and speed to ensure smooth operation during peak usage periods.
- **Feedback Mechanism and Iterative Development:** Establish a robust feedback loop to gather user input regularly. Utilize this feedback to iterate on system functionalities, improve recommendation algorithms, and address user pain points effectively.
- **Security and Data Privacy:** Strengthen data security measures to safeguard user information and ensure compliance with data protection regulations. Implement

encryption protocols and regular security audits to mitigate risks and build user trust.

- **Scalability and Future Expansion:** Enhance system scalability to accommodate growing user bases and expanding course catalogs. Plan for future enhancements and technological advancements to maintain competitiveness and relevance in the dynamic educational technology landscape.

By focusing on these areas of improvement, the project can continue to evolve and meet the evolving needs of its users, delivering a cutting-edge educational platform that fosters learning excellence and user satisfaction.

REFERENCES

[1] Node.js documentation

<https://nodejs.org/docs>

[2] Next.js documentation

<https://nextjs.org/docs>

[3] BERT model

<https://phamdinhhkhanh.github.io/2020/05/23/BERTModel.html>

[4] Recommendation system

<https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/>

[5] Pytorch

<https://pytorch.org/docs>

