Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

An Open Source Platform for Course Management

Yifan Chen

Supervisor: Associate Prof. John Waldron

August 16, 2024

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MSc (Computer Science)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

I consent / do not consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement. **Please consult with your supervisor on this last item before agreeing, and delete if you do not consent**

Signed: _____ Date: _____

Abstract

With the advancement of computer technology, particularly in the context of the "Internet + Education" paradigm, course management platforms have become a critical component of educational technology. However, existing platforms face several challenges, including outdated front-end and back-end technologies, proprietary systems with high costs, limited functionality, and a lack of customization, all of which constrain their applicability. This paper proposes and implements a fully open-source course management platform aimed at addressing these technical and functional deficiencies to enhance user experience.

The paper provides a detailed analysis of system requirements and adopts popular web development technologies for technical selection. The platform employs Spring Boot for the backend and Vue for the frontend, achieving complete front-end and back-end separation. The design and implementation of the system encompass data processing architecture, system architecture diagrams, database design, and the detailed design and implementation of individual modules. The system's feasibility was validated through deployment and testing on a local computer.

This platform introduces several innovative features, including a complex multidimensional classification design, dual role design, integration of a rich text editor, support for online video streaming and downloading, user check-in functionality, course content access restrictions, and the use of ECharts for statistical data visualization. These innovations not only enhance the platform's functionality but also cater to the personalized needs of users and provide administrators with robust data analysis tools.

Lay Abstract

In the era of the internet, course management platforms are a crucial part of online education. However, existing platforms often face issues such as outdated technology, high system costs, limited functionality, and difficulty adapting to user needs. To address these issues, this paper introduces a brand new open-source course management platform designed to enhance user experience by incorporating updated technology and expanded features.

By analyzing the needs of users and the system, this platform has adopted modern web technologies to build a better teaching and management system. The design of the platform took into consideration data processing, system architecture, and database configuration, and it meticulously planned the implementation of each functional module. The platform's effectiveness was confirmed through local testing.

Furthermore, this platform has introduced several new features, such as a more complex classification system, multi-role user design, an integrated text editor, online video viewing and downloading, user check-in, course access controls, and visualized data analysis capabilities. These added features not only make the platform more powerful but also better meet the personalized needs of various users, providing administrators with effective data analysis tools. Such a design greatly enhances the user experience.

Acknowledgements

I would like to express my sincere gratitude to my mentor, Dr. John Waldron, for his patient guidance and invaluable support throughout my studies. Additionally, I am thankful for my classmates Peng Xueyuan, Jiang Yifan, Sun Peichen, Yang Yi, and others for their help and companionship during this time. Special thanks to Deng Jiaming for his assistance with my coursework and job search. Lastly, I cherish the enriching and fulfilling learning experiences I've had over the past year.

Contents

List of Figures

List of Tables

1 Introduction

In the context of ongoing advancements in computer technology, various industries have become inextricably linked with the application of computers. The benefits of computer technology in the education sector are particularly evident. Against the backdrop of "Internet+ Education", the timely emergence of course management platforms represents a key innovation in educational technology. These evolving applications are enhancing the quality, efficiency, and adaptability of education, particularly during emergencies such as public health crises. They transcend traditional temporal and spatial limitations, enabling users to study courses from any location, while allowing teachers and administrators to guide and monitor student progress remotely. However, current course management platforms are hindered by the use of outdated front-end and back-end frameworks and technologies, which result in an incomplete separation of front-end and back-end, low operational efficiency, closed-source systems with high costs, and limited functionality with a lack of customization, thus reducing their applicability and degrading the user experience [1].

Addressing these pain points of current course management platforms, this paper proposes a comprehensive design and implementation plan for a course management platform that effectively addresses the existing technological and functional issues, significantly enhancing the user experience. The paper begins with an analysis of system requirements, followed by a selection of technologies for the front-end, back-end, and database of the course management platform, in line with current popular web development technologies. Subsequently, the paper details the system design, including the data processing architecture, system architecture diagram, database design, and the design of the sub-modules that constitute the platform. Following this, the implementation of these sub-modules based on the system design is discussed. After implementing the system, it will be deployed on a local computer to test the functionality of its modules, ensuring the system operates effectively [2]. Finally, the paper concludes by summarizing the design and implementation process of the course management platform and discussing future work.

2 Background

2.1 Existing Platform Overview

Drawing on information from the literature [3] and web sources [4], this paper selects the six most popular course management platforms: Moodle, Open edX, Canvas LMS, Chamilo, Sakai, and Totara Learn. These platforms are compared based on four main criteria: open-source status, technology stack used, separation of front-end and back-end, and functionality. The functionality criterion is further subdivided into customization, ease of use, course creation, and content management. The specific comparison information is listed in Table 2.1.

From the comparison table above, it can be observed that although most of the popular course management platforms are open-source, issues such as the use of outdated web development technologies and the lack of complete separation between front-end and back-end are prevalent. Except for Open edX, the other five course management platforms do not achieve full separation between front-end and back-end. Furthermore, Open edX suffers from low customization capability and usability challenges. Additionally, its backend is based on Django, which lacks the performance, scalability, and robust ecosystem offered by alternatives like Spring Boot. Therefore, there is a pressing need for a new open-source course management platform that achieves complete front-end and back-end separation, offers powerful functionality, and is user-friendly to fill the gap in this field.

2.2 Requirements Analysis

Upon understanding and analyzing the deficiencies and shortcomings of existing mainstream course management platforms, this article will further analyze and organize the user requirements for course management platforms. User requirements can specifically be categorized into functional requirements, non-functional requirements, and design constraints [5]. This paper will also analyze user needs for this platform from these three perspectives.

2.2.1 Functional Requirements

The users of this platform can generally be divided into two categories: standard users and administrators. In my vision, teachers providing online courses can act directly as platform administrators; alternatively, administrators could purchase the ownership of online courses from the teachers, then upload these courses via the admin interface to the course management platform for ordinary users to buy and study. Thus, administrators are responsible for managing user information, and online courses and resources on the platform. A significant portion of the resources on the platform originates directly from standard users, who upload their content. Once approved by administrators, these resources are made available on the user interface. If other users wish to download these resources, they must spend points, which are then credited back to the users who uploaded the resources. Therefore, users can not only purchase and view various courses and resources but can also upload their own resources and set the number of points required for others to download them, thereby earning points from downloaders. All courses and resources, except for the free ones, require purchase for access. There are two ways to make a purchase: using money or redeeming points.

2.2.2 Non-Functional Requirements

As a supplement to functional requirements, users also present some non-functional requirements. For this platform, the main non-functional requirements are as follows:

1. When developing backend APIs, adherence to RESTful API standards is required. This enhances the maintainability and scalability of the APIs. Due to the standardization and simplicity of RESTful APIs, developers can understand their workings more rapidly, making it easier for new developers on this platform to utilize these APIs.

2. This project adopts Vue.js as the frontend framework. In coding the page components, i.e., writing code in the views folder, it is essential to organize the components properly: user interface components should be placed in the front folder, administrator interface components in the manager folder, and common components for both user and administrator interfaces should be placed directly under the views folder, belonging neither to front nor to manager. This arrangement facilitates a clear frontend code structure and enhances the reusability of page components.

3. The frontend pages displayed by the final product of this project should be neat and aesthetically pleasing, while aligning with the usage habits of most users.

2.2.3 Design Constraints

For this platform, the following design constraints are primarily considered: Given that this project aims to develop a web application, which needs to be deployed on a cloud server for user access and utilization, it is imperative that the project operates on a Linux environment, and that the technology stack utilized within this project is compatible with the Linux environment.

This concludes the requirements analysis for the project. The design and implementation of these requirements will be discussed in detail in the chapter on system design and implementation.

2.3 Technology Stack

2.3.1 Frameworks and Database

The backend framework adopted for this project is Spring Boot, version 2.5.9; the SQL mapping framework is MyBatis, version 2.2.1. The frontend framework used is Vue.js, version 2.6.14; the frontend UI component library is ElementUI, version 2.15.14. The database used in this project is MySQL, version 8.0. The following will provide a brief introduction to these frameworks and the database, explaining the rationale behind these technical selections.

Spring Boot [6] is an open-source framework based on Java, designed to facilitate rapid construction and deployment of microservice applications. It represents an integration of the Spring platform with third-party libraries, offering a highly convenient way to launch and run Spring applications. By adopting the principle of "convention over configuration," Spring Boot automatically provides default configurations, significantly simplifying the programming model and reducing the configuration effort for developers. Moreover, it includes various embedded web servers, such as Tomcat and Jetty, allowing developers to run applications without the need for separate server deployments. Particularly focused on the rapid, autonomous deployment of microservices, Spring Boot is well-suited for the cloud service environment and ideal for building scalable internet applications, enabling developers to concentrate on the functional development of applications rather than underlying configurations. During the technology selection process, I considered Django and Flask; however, both presented specific issues. Django does not support complete front-end/back-end separation without introducing the Django REST framework to create a complete RESTful API, which can be cumbersome; hence, it was not chosen as the backend framework. Flask, while straightforward and user-friendly, offers limited functionality and could not provide the robust backend support needed for the complex requirements of this

project, thus it was also not selected. After careful consideration, I chose Spring Boot as the project's backend framework due to its:

1. Suitability for applications requiring high scalability and robust performance.

2. Comprehensive ecosystem.

3. Extensive documentation and community support.

4. Dependency injection feature that leads to cleaner code, simplifying testing and maintenance.

MyBatis [7] is a popular Java persistence framework that offers a semi-automated ORM (Object-Relational Mapping) implementation. Unlike fully automated ORM frameworks such as Hibernate, MyBatis allows developers to write SQL statements through XML or annotations and map these statements to Java objects. This approach provides greater flexibility, enabling developers to control SQL execution directly and optimize database operation performance. Additionally, MyBatis is relatively simple to configure and use, facilitating integration with popular frameworks such as Spring, and is suitable for projects requiring direct SQL operations for granular control over database interactions. MyBatis-Plus [8] was also considered for this project, but due to its limitations in direct SQL manipulation, which resulted in increased usability but decreased flexibility, MyBatis was ultimately chosen to bridge Spring Boot and the database.

Vue.js [9] is an open-source JavaScript framework primarily used for building user interfaces and single-page applications (SPAs). It is centered around data-driven and component-based philosophies, allowing developers to create richly interactive interfaces through simple APIs. Vue.js is designed to be incremental, meaning its core library focuses solely on the view layer but can be easily integrated with other libraries or existing projects, or used in complex applications. This flexibility makes Vue.js not only easy to learn for beginners but also capable of meeting the development needs of complex projects, making it a favorite among developers and companies. React.js [10] was considered as an alternative frontend framework, but due to Vue.js's simplicity, intuitive component construction, and easier integration features, it was chosen as the frontend framework for the project.

ElementUI [11] is a frontend UI component library based on Vue 2.0, specifically designed for desktop applications. It provides a rich set of UI elements, including buttons, forms, notifications, navigation menus, tables, and dialogs, all aimed at simplifying the development process and enhancing interface consistency. ElementUI's modern design and rich functionality make it easy to integrate and extend, ideally suited for rapidly building high-quality enterprise-level application interfaces. With comprehensive documentation and community support, ElementUI has become one of the preferred UI frameworks among Vue.js developers. As Vue.js was selected as the frontend framework for this project,

ElementUI was naturally chosen as the accompanying UI framework.

MySQL [12] is a widely used open-source relational database management system (RDBMS) initially developed by the Swedish company MySQL AB and now maintained by Oracle Corporation. It uses Structured Query Language (SQL) for database management and is popular for its high performance, reliability, simplicity, and open-source nature. MySQL is capable of handling large-scale data sets, making it an ideal choice for building websites and online transaction processing applications. It supports multi-user access and multi-threading, allowing MySQL to run efficiently on various operating systems, widely used in internet data processing and storage. Given the need in this project to persistently store critical data such as user information, course information, and order details, the chosen database must support the ACID properties, necessitating the use of an SQL database over a NoSQL database. Based on these considerations, MySQL, a classic among SQL databases, was selected.

2.3.2 Software and Environment

In this project, the software utilized includes IntelliJ IDEA and Navicat, with the former at version 2023.3 and the latter at version 17. The runtime environment comprises JDK 1.8 and Node.js 16. The roles that these software applications and runtime environments play in the development and operation of the project will be briefly introduced.

IntelliJ IDEA [13], developed by JetBrains, is a robust Integrated Development Environment (IDE) designed primarily for the Java language, yet it also supports a variety of other programming languages such as Kotlin, Groovy, and Scala. It offers an array of intelligent features, including code auto-completion, instant error detection, code refactoring tools, and version control system integration, all aimed at enhancing development efficiency. Furthermore, IntelliJ IDEA supports web and mobile application development, incorporating support for various popular frameworks and technologies. Its user-friendly interface can be extended through plugins, making it the tool of choice for professional developers and large project teams. Both the frontend and backend components of this project were developed in IntelliJ IDEA, leveraging its capabilities to simplify Spring Boot configurations, auto-complete code, and run projects with a single click.

Navicat [14] is a multi-database management tool that supports various database systems, including MySQL, PostgreSQL, Oracle, SQLite, and MariaDB. It is widely acclaimed for its intuitive graphical user interface and powerful features, facilitating easy database management, development, and maintenance. Navicat offers advanced features such as data migration, import/export tools, data synchronization, backup, and report generation, suitable for businesses and individual users of varying sizes. Additionally, Navicat supports cloud database connections, allowing users to connect directly to cloud databases via HTTP/HTTPS protocols, which is beneficial for remote work and team collaboration.

During the local development phase of this project, Navicat's visual interface was primarily used to create and modify tables in the MySQL database, utilizing its graphical interface to configure table details such as fields, data types, and key types. After development, I used Navicat to export the table structure to SQL files, a feature that aids in database backup and migration, helping developers to quickly replicate databases in the cloud and deploy services.

JDK 1.8 [15] (Java Development Kit 1.8), also known as Java 8, is a major release of Oracle Corporation's Java programming environment. Launched in 2014, this version introduced numerous significant enhancements and new features such as lambda expressions, default and static methods in interfaces, and a new date-time API, which greatly simplified Java programming and enhanced development efficiency. Java 8 also improved support for collection operations with the introduction of the Stream API, making collection processing more efficient and concise. Furthermore, Java 8 optimized and strengthened aspects of the JVM, JavaScript engine, and security to meet the performance and security demands of modern applications. In this project, JDK 1.8 primarily serves as a compiler and provides the runtime environment for Spring Boot.

Node.js 16 [16] is a significant version of Node.js, a JavaScript runtime built on Chrome's V8 engine that enables the execution of JavaScript code on the server-side. This version introduced several new features and improvements, including native support for Apple Silicon, performance enhancements in timers, better package manager configurations, and support for new JavaScript language features. Node.js 16 also enhanced security and the module ecosystem, providing additional APIs to support the development of modern web applications and services. These improvements make Node.js 16 a preferred platform for developing high-performance, efficient web applications. In this project, Node.js 16 primarily serves the following functions:

1. Compiles Vue single-file components (.vue files) into JavaScript, CSS, and HTML that browsers can understand.

2. Acts as a package management tool, allowing developers to install, update, and manage required libraries and frameworks.

3. Use for server-side rendering of Vue applications.

Table 2.1: Comparison of Features Across Different Course Management Platforms

Features	**Moodle**	**Open edX**	**Canvas LMS**	**Chamilo**	**Sakai**	**Totara Learn**
Open Source or not	Yes	Yes	Yes	Yes	Yes	No
Technology Stack	Backend: PHP Frontend: Javascript	Backend: Python (Django) Frontend: Javascript (React)	Backend: Ruby on Rails Frontend: Javascript (React)	Backend: PHP Frontend: Javascript	Backend: Java (Spring) Frontend: Javascript (JSF, JQuery)	Backend: PHP Frontend: Javascript
Separation between Frontend and Backend	No	Yes	Mostly integrated, partial separation	No	No	No
Customization	High (Extensive plugins)	Low	Medium	Medium	Medium	High (Moodle fork)
Ease of Use	Medium	Low	High	High	Medium	Medium
Course Creation	Flexible	Powerful	Streamlined	Easy	Modular	Flexible
Content Management	Yes	Yes	Yes	Yes	Yes	Yes

3 System Design and Implementation

3.1 Pipeline

The platform's pipeline consists of two components: the front end, which is the browser interface accessible to all users, and the back end, which supports the front end with server capabilities, as depicted in Figure 3.1.

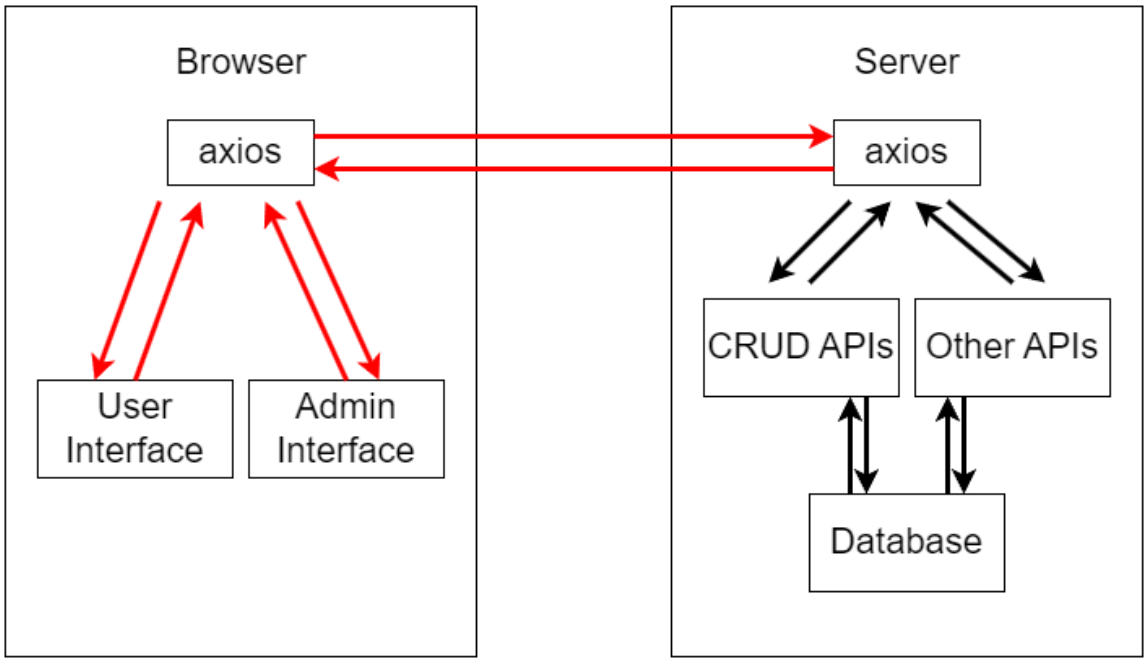

Figure 3.1: Pipeline

Initially, users must register an account on the registration page. It is important to note that this page only supports the registration of standard user accounts, not administrator accounts. Administrator accounts are added directly through the Admin Info page on the administrator interface using an initial administrator account, ensuring that administrative privileges are not misused and securing the platform's integrity. Following registration, users

may log in on the login page. Using either an administrator or a standard user account, users can access respective interfaces. These interfaces send HTTP requests to the server via axios and handle all received responses. Upon receiving a request, the server invokes APIs to process it, which can be broadly categorized into two types: APIs responsible for database CRUD operations and others that facilitate login, registration, current user information retrieval, and upload and download functions. These APIs interact directly with the backend database, the direct source of data operations. Once the backend has processed a request, it generates an HTTP response, including status codes, headers, and a body (the data to be returned to the frontend). This response is then sent back to the requesting frontend. The frontend's axios request awaits and eventually receives the HTTP response from the backend. Finally, axios processes this response, converting the data in the response body into JavaScript objects for rendering and display on the frontend interface.

3.2 Use Case Diagram

This system is an online course management platform designed to provide users with functionalities for online course participation and document upload/download, and to enable platform administrators to manage user information and all online resources. Following a comprehensive analysis of the overall system requirements, we have delineated the functional requirements of the system based on user roles, categorizing them into administrator and standard user functionalities.

The functional requirements for administrators are illustrated in Figure 3.2. Specifically, the administrator interface should encompass the following capabilities:

1. Administrator Login

2. System Home page, featuring a Notice List and Platform Resource Statistics Charts Information Management functionality, which should include pages for Announcements, Course Information, Points Area, Document Review, and Orders. Each of these five pages should facilitate operations such as Add New, Batch Delete, Edit, Delete, Search, and Reset. Additionally, the Course Information and Points Area pages should offer a One-Click View option for all courses, and the Document Review page should include a Review function, allowing administrators to determine whether user-uploaded documents can be displayed on the users' massive resources page by selecting Approved, Pending, or Not Approved.

3. User Management functionality, which should cover management of both administrator and standard user information. Management of information for both user categories should also support Add New, Batch Delete, Edit, Delete, Search, and Reset operations.

The functional requirements for standard users are illustrated in Figure 3.3. Specifically, the standard user interface should encompass the following capabilities:

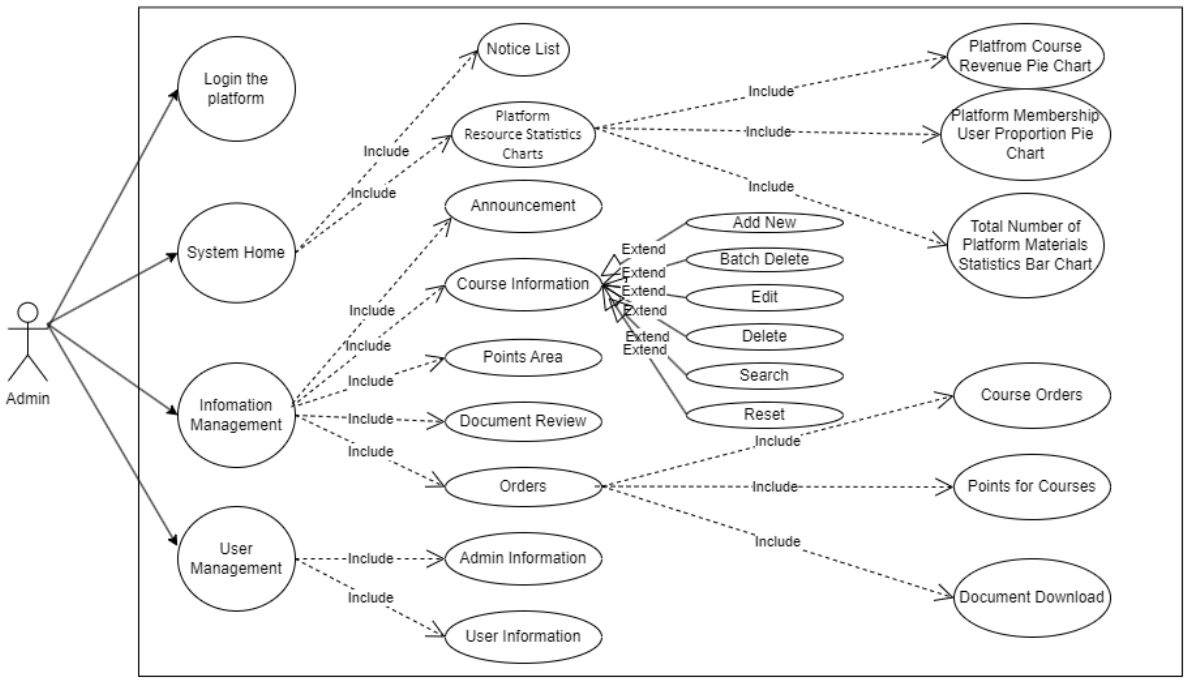

Figure 3.2: System Use Case Diagram(Admin)

1. User Registration

2. User Login

3. Home Page, which includes sections for Online Courses and Online Resources, with the former further divided into Video Courses, Points Area, and Image and Text Courses

4. All Courses, Points Area, and Massive Resources sections, each supporting Search and Reset operations

5. Personal Center, where users can view their personal information, including username, phone number, email, points, balance, and membership status. Additionally, this section should facilitate functionalities such as password modification, simulation of account recharging, and saving of updated personal information.

3.3 System Architecture Diagram

The system architecture of the Online Course Management Platform is depicted in Figure 3.4. The platform's Application Expression Layer is segmented into administrator and user interfaces. Within the Operation Logic Layer, the administrator module is further divided into the System Home module, Information Management module, and User Management module; the user module is further segmented into the Home Page module, All Courses module, Points Area module, Massive Resources module, and Personal Center module. In the Data Processing Layer, there are primarily two categories of APIs: CRUD APIs and

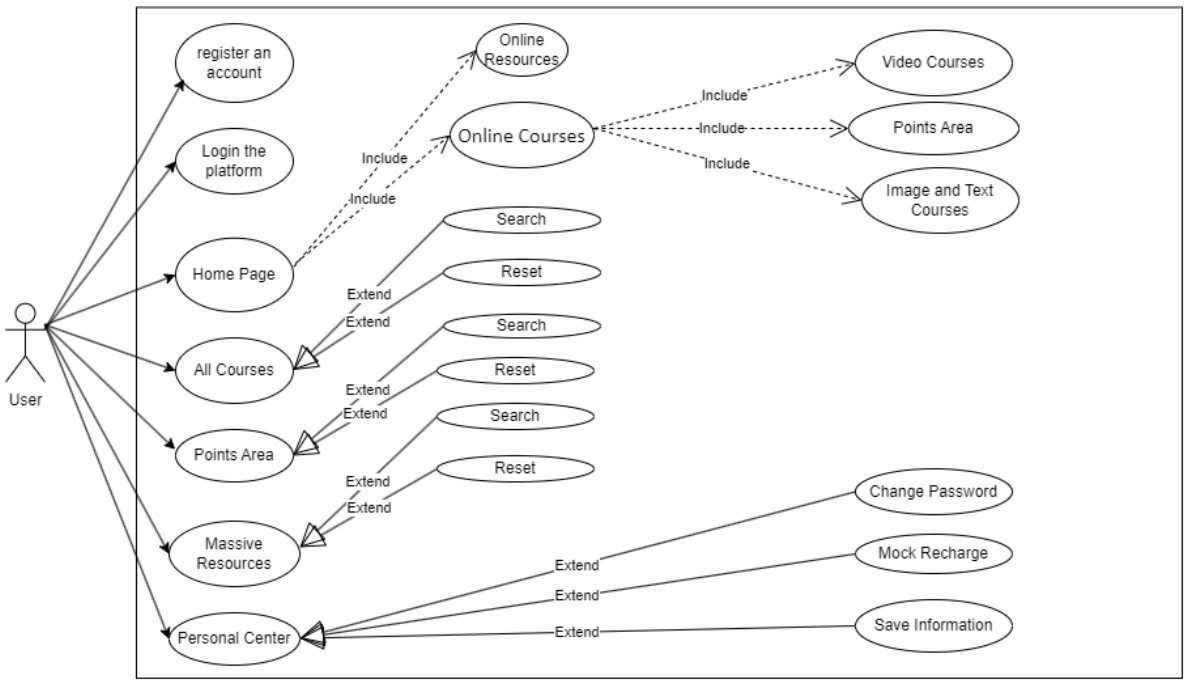

Figure 3.3: System Use Case Diagram(User)

Other APIs, which are collectively utilized by the aforementioned modules to provide data support. The final layer is the Data Storage Layer, where the system's data is stored in three types of tables: User Information Tables, Course and Resource Information Tables, and Order Information Tables.

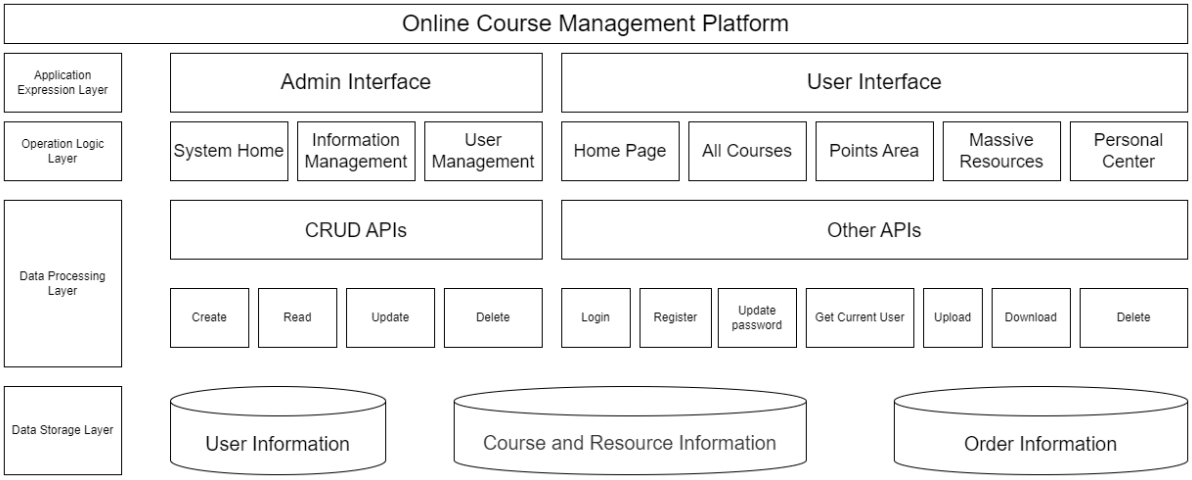

Figure 3.4: System Architecture Design

Based on the System Architecture described, I have identified the technology stack used for each layer, as illustrated in Figure 3.5. For the Application Expression Layer, I have selected Vue.js as the framework, paired with the ElementUI component library, utilizing HTML, CSS, and JavaScript as the programming languages. In the Network Communication Layer, I have opted for HTTP and HTTPS as the communication protocols. For the Operation

Logic Layer, the backend framework chosen is Spring Boot, through which I have defined several RESTful APIs for frontend utilization. In the Data Processing Layer, the persistence framework employed is MyBatis, which interacts with database data using both annotations and XML. For the Data Storage Layer, data intended for permanent storage is housed in a MySQL database, while transient data, such as JWTs, is stored in the Browser Cache.

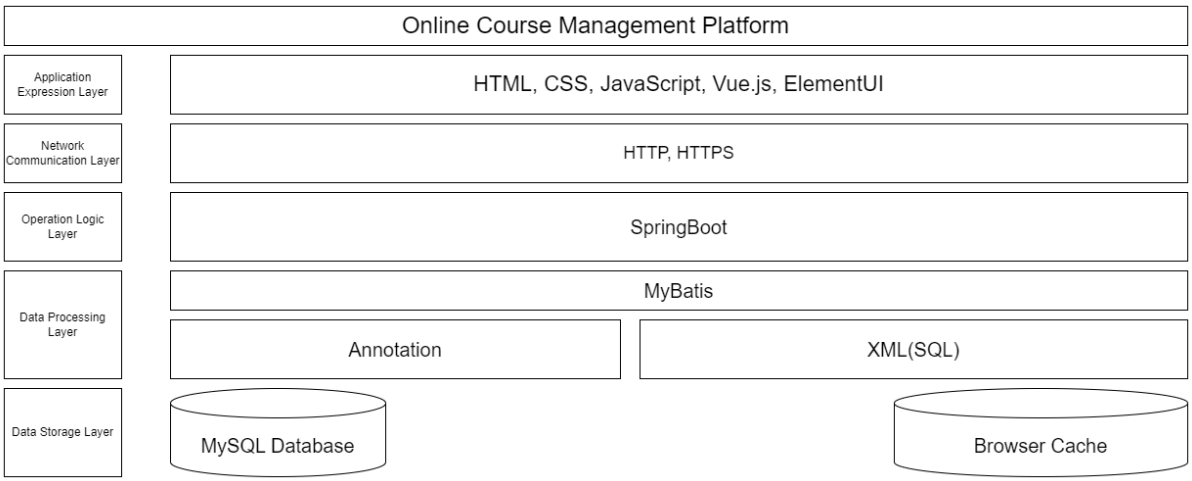

Figure 3.5: System Technical Architecture Diagram

The development of both the frontend and backend of this project was completed in IntelliJ IDEA Ultimate 2023.3. Database-related operations were conducted using Navicat Premium 17, which supports visual creation and modification of database tables and allows users to convert table structures directly into SQL files, facilitating database backup and migration. The runtime environment for this project includes Node.js 16 and JDK 1.8.

3.4 Database Design

Database design is fundamental to the user management, course management, and order management modules of this platform. The project employs MySQL, necessitating the storage of data belonging to different modules in separate tables. Through an analysis of the functional requirements of each module on the platform, I have created 11 tables in MySQL, which can broadly be classified into three categories: User Information Tables, Course Information Tables, and Order Information Tables. Next, I will present the design details of each table.

3.4.1 User Information Table Design

In my database design, there are three tables related to user information, which respectively document the information of administrators, standard users, and the check-in times of standard users. The detailed attributes of these three tables are listed below.

It is important to note that Table 3.3 is necessary because it records the specific check-in times of each user, which is crucial for implementing a daily check-in feature. Only by recording the check-in times can it be determined whether a user is checking in for the first time on a given day. If it is the first check-in of the day, the user is awarded 10 points; otherwise, the user is reminded that they have already checked in for the day and should return the following day to check in and earn points.

3.4.2 Course and Resources Information Table Design

In the database design of this platform, there are five tables related to course and resource information. The platform offers two types of courses: the first type comprises online courses that can only be purchased with euros, and the second type includes courses that can only be redeemed with points; the information for these courses is stored in Tables 3.4 and 3.5 respectively. The platform also facilitates the display of a substantial amount of user-uploaded resources, necessitating Table 3.6 for recording information related to these resources. For all online courses, users can leave comments in the comment section after purchasing a course, and other users can reply to these comments, thus Table 3.7 is required to document the comment interactions of different users under different courses. Administrators have the capability to post announcements on the admin interface, which are then displayed on the user interface, necessitating Table 3.7 for recording information related to these announcements. The detailed attributes of these five tables are listed below.

3.4.3 Order Information Table Design

In the course management platform, three types of resources are available for purchase using euros or points: online courses, courses in the points area, and resources uploaded by users. For these purchasable resources, it is necessary to separately record the order information corresponding to each type, which is stored in three distinct tables in MySQL. The detailed information for these three tables is listed below.

Table 3.1: Admin Information Table

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
username	varchar	255	NO	YES	Username
password	varchar	255	NO	YES	Password
name	varchar	255	NO	YES	Name
avatar	varchar	255	NO	YES	Avatar
role	varchar	255	NO	YES	Role: ADMIN
phone	varchar	255	NO	YES	Phone Number
email	varchar	255	NO	YES	Email

Table 3.2: User Information Table

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
username	varchar	255	NO	YES	Username
password	varchar	255	NO	YES	Password
name	varchar	255	NO	YES	Name
avatar	varchar	255	NO	YES	Avatar
role	varchar	255	NO	YES	Role: ADMIN
phone	varchar	255	NO	YES	Phone Number
email	varchar	255	NO	YES	Email
member	varchar	255	NO	YES	A Member or not
score	int	NA	NO	YES	Points
account	double	10	NO	YES	Balance

Table 3.3: User Check-in Information Table

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
userid	int	NA	NO	YES	Username
time	varchar	255	NO	YES	Check-in time
day	varchar	255	NO	YES	Check-in date

Figure 3.6 presents the system's entity diagram, showcasing all 11 entities utilized in this project. During the database design process, these abstract entities are converted into concrete database tables, the design of which has been detailed in the previous sections. Figure 3.7 illustrates the system's ER diagram, primarily highlighting the relationships between different entities. As depicted in the diagram, administrators are responsible for uploading notices, online courses, and courses available for points redemption, as well as reviewing user-uploaded resources. The order information tables for courses or resources purchased by users are automatically generated and made accessible to administrators. Users can leave comments in the comment section under courses and can also click the check-in button to earn points; information related to these two actions is respectively recorded in the comment table and the signin table.

In modern database design, the use of foreign keys is generally discouraged. Foreign keys can lead to performance bottlenecks on the database server and hinder system scalability, particularly in high-traffic, data-intensive internet applications where the use of foreign keys can be even more problematic. To optimize database server performance, all foreign key constraints should be implemented by the application, even though this approach may consume more application server resources. In this project, due to the large number of tables in the database and the complexity of the relationships between these tables, I did not use foreign keys during the database design process. Instead, the relationships between tables

Table 3.4: Online Courses Information Table

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
img	varchar	255	NO	YES	Course Cover
name	varchar	255	NO	YES	Course Name
content	varchar	255	NO	YES	Course Description
type	varchar	255	NO	YES	Course Type
price	double	10	NO	YES	Course Price
video	varchar	255	NO	YES	Course Video
file	varchar	255	NO	YES	Material Link
discount	double	10	NO	YES	Course Discount
recommend	varchar	255	NO	YES	Recommend or Not
time	varchar	255	NO	YES	Release Time

Table 3.5: Points Area Courses Information Table

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
img	varchar	255	NO	YES	Course Cover
name	varchar	255	NO	YES	Course Name
content	longtext	NA	NO	YES	Course Description
type	varchar	255	NO	YES	Course Type
price	int	NA	NO	YES	Course Price
video	varchar	255	NO	YES	Course Video
file	varchar	255	NO	YES	Material Link
recommend	varchar	255	NO	YES	Recommend or Not
time	varchar	255	NO	YES	Release Time

were defined in the backend code. The key relationships are as follows:

1. In the orders table, the course_id field is a foreign key that references the id field in the course table.

2. In the scoreorder table, the score_id field is a foreign key that references the id field in the score table.

3. In the fileorder table, the file_id field is a foreign key that references the id field in the information table.

4. In the comment table, the user_id field is a foreign key that references the id field in the user table.

5. In the signin table, the user_id field is a foreign key that references the id field in the user table.

Table 3.6: Resources Information Table

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
name	varchar	255	NO	YES	Resource Name
img	varchar	255	NO	YES	Resource Cover
score	int	NA	NO	YES	Points Required
time	varchar	255	NO	YES	Upload Time
recommend	varchar	255	NO	YES	Recommend or Not
user_id	int	NA	NO	YES	Uploader ID
status	varchar	255	NO	YES	Review Status
descr	varchar	255	NO	YES	Review Remarks
content	longtext	NA	NO	YES	Resource Description
file	varchar	255	NO	YES	Resource Link

Table 3.7: Table recording user comments

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
user_id	int	NA	NO	YES	Student ID
course_id	varchar	255	NO	YES	Course ID
time	varchar	255	NO	YES	Comment Time
content	varchar	255	NO	YES	Comment Content
parent_id	int	NA	NO	YES	Parent ID

3.5 System Module Design and Implementation

This section will describe the system design approach for each module of the platform, using sequence diagrams, activity diagrams, or textual descriptions. The system is composed of the User Account Management Module, Course Information Module, User Interface, and Administrator Interface, with the User Interface and Administrator Interface further divided into several submodules. The following sections will provide a detailed explanation of the design and implementation process for each system module. The description of each system module is mainly divided into three parts: user operations on the system frontend, backend processing logic, and database operations.

Overall, the backend logic and database operations of this platform are relatively standardized, primarily involving combinations and reuse of CRUD (Create, Read, Update, Delete) APIs, occasionally with minor modifications or the addition of new features. However, the frontend implementation of this platform is more flexible, providing a user-friendly, cleverly designed, and functionally diverse interaction model. In both the User Interface and Administrator Interface, some submodules share similar functionalities, which will not be redundantly explained.

Table 3.8: Table of announcements added by the administrator

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
title	varchar	255	NO	YES	Announcement Title
content	varchar	255	NO	YES	Announcement Content
time	varchar	255	NO	YES	Creation Time
user	varchar	255	NO	YES	Creator

Table 3.9: Table recording information on the paid online courses purchased by users

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
course_id	int	NA	NO	YES	Course ID
price	double	10	NO	YES	Course Price
order_id	varchar	255	NO	YES	Order Number
time	varchar	255	NO	YES	Order Time
user_id	int	NA	NO	YES	Ordering User

3.5.1 User Account Management Module

This module is primarily responsible for implementing user registration, login, and password modification functionalities. After registering and logging into the platform, users can change their passwords in the "Change Password" section of the personal center. The functionality of this module is identical for both administrators and regular users; therefore, the module has been abstracted and made independent of the administrator and user interfaces. The sequence diagram for the user registration process and the activity diagram for password modification are shown in Figures 3.8 and 3.9, respectively.

The specific implementation process of this module is as follows:

User Login and Registration

1. Frontend

In the frontend, the el-form-item and el-select components from the Element UI library are used to allow users to select their login role (administrator or regular user). Based on the selected role, the frontend logic determines the appropriate redirection to the corresponding homepage.

2. Backend

The login process involves the login method, where the backend first searches for the user by username. If the user does not exist, an exception is thrown. Upon verifying the correct password, the backend generates a Token containing the user ID and role, which is then

Table 3.10: Table recording information on courses redeemed by users with points

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
score_id	int	NA	NO	YES	Points Course ID
score	int	NA	NO	YES	Points Required
order_id	varchar	255	NO	YES	Order ID
time	varchar	255	NO	YES	Redemption Time
user_id	int	NA	NO	YES	Redeeming User

Table 3.11: Table recording information on resources redeemed by users with points

Field Name	Type	Length	Primary Key	Empty	Description
id	int	NA	YES	NO	Auto-increment
file_id	int	NA	NO	YES	Resources ID
score	int	NA	NO	YES	Points Required
order_id	varchar	255	NO	YES	Order ID
time	varchar	255	NO	YES	Redemption Time
user_id	int	NA	NO	YES	Redeeming User

returned to the frontend.

During registration, the account information submitted by the user is copied into a new User object. The add method is then called to add the user data to the database.

3. Database

During login, the backend queries the database to verify the user's existence and password correctness. During registration, the new user data is inserted into the database.

Password Modification

1. Frontend

Users can submit their old password and new password to initiate the password modification process. The frontend verifies the user's role and triggers the password change workflow.

2. Backend

The password modification feature first checks if the old password provided by the user is correct. If incorrect, an error is thrown. If correct, the new password is updated in the database.

3. Database

Password modification involves querying the current user's information to verify the old

Figure 3.6: System Entity Diagram

password, and upon successful verification, updating the password field in the user's record.

The module's implemented effects are shown in Figures 3.10 and 3.11:

3.5.2 Course Information Module

1. Frontend

Users can perform several operations through the frontend interface, including viewing course information, adding new courses, editing course details, deleting courses, and bulk deleting courses. The frontend is implemented using the Vue.js framework, with the specific functionalities as follows:

- Search and Filter: Users can search for courses by entering the course name and clicking the "Search" button, which calls the backend API to retrieve data. A "Reset" button is also provided to clear the search box and refresh the course list.

- Add and Edit Courses: By clicking the "Add" button, users can add new course information. The edit feature allows users to modify existing course information. These operations are carried out via a pop-up form that includes various course attributes such as

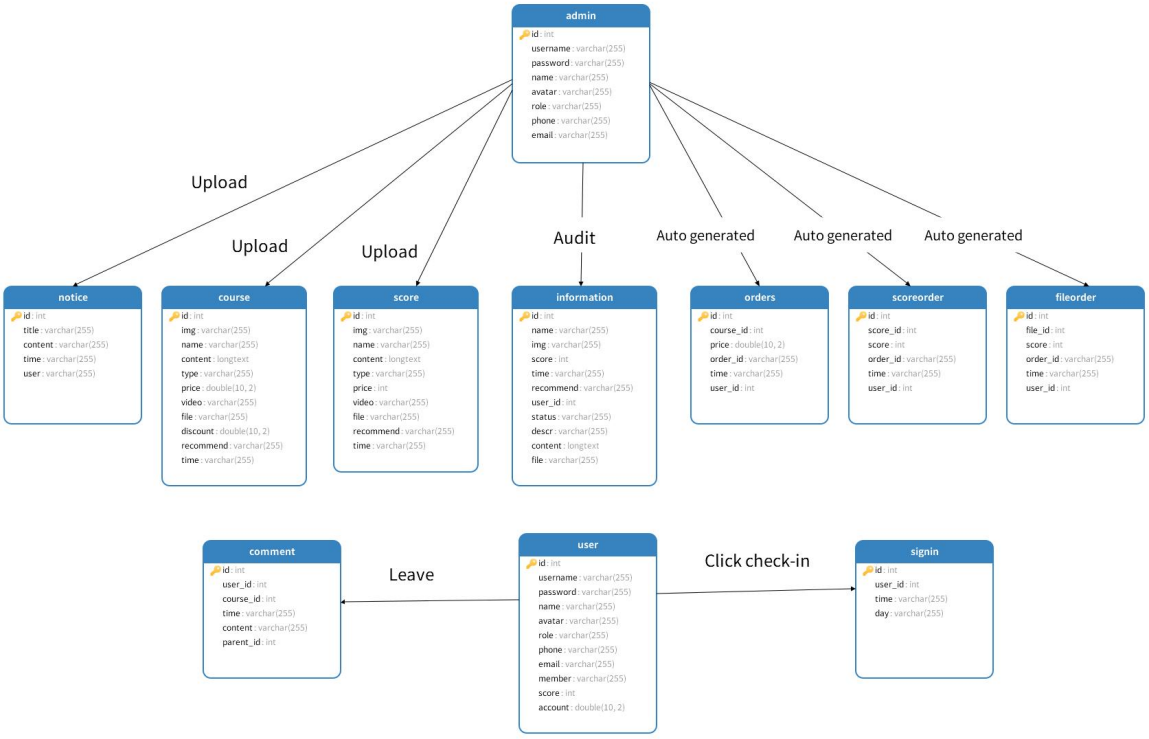

Figure 3.7: Entity Relationship Diagram

name, type, price, etc.

- Delete Operations: Users can delete selected courses individually or delete multiple selected courses in bulk.

- Pagination: The course list supports pagination, allowing users to browse different sets of courses by selecting the page number.

2. Backend

The system backend is built using the Spring Boot framework, with the specific processing logic as follows:

- Course Information Management: The backend provides a complete set of CRUD operations, including adding, deleting, updating, and querying course information.

- Business Logic Handling: For instance, when adding or updating course information, the backend performs data validation to ensure that all required information is correctly filled out.

- Pagination and Filtering: The backend supports pagination of course information and filtering based on course names.

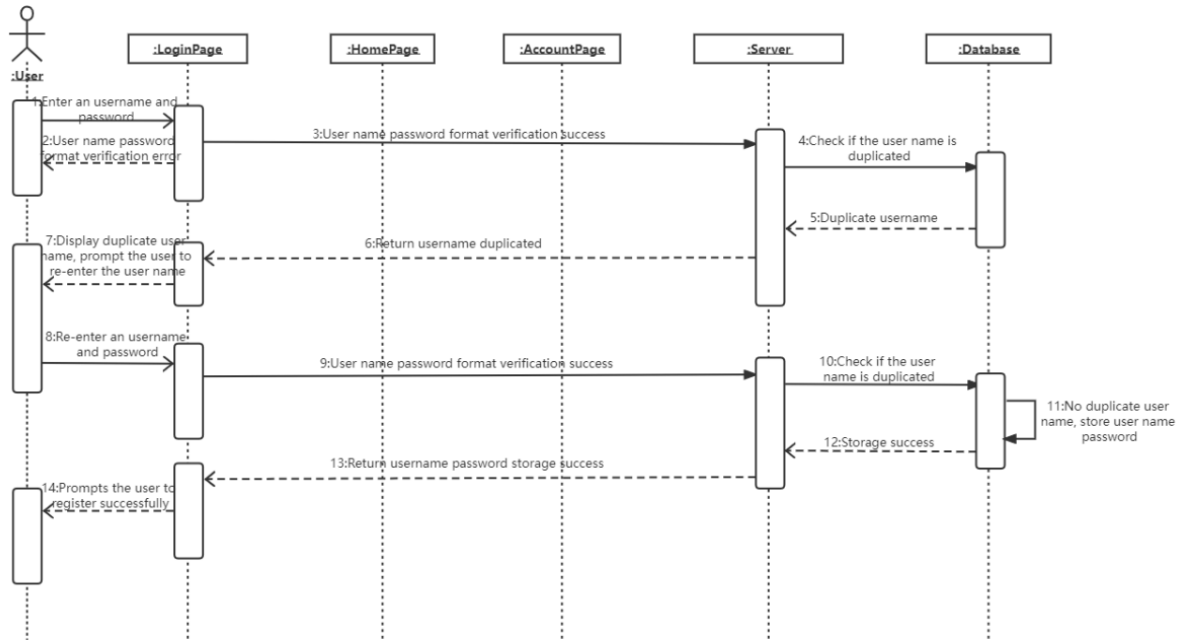

Figure 3.8: Sequence Diagram for User Registration Process

3. Database

The database uses MySQL for data storage, with the following key operations:

- Course Information Table Design: The course table includes basic attributes of courses such as course name, type, price, etc.

- SQL Mapping Configuration: Using the MyBatis framework, SQL operations on the course table are defined through XML configuration files, including inserting new records, updating records, deleting records, and querying records.

- Transaction Management: Ensures data consistency and integrity; for example, in a bulk delete operation, the backend must ensure that all specified courses are deleted.

The implemented results of this module are shown in Figures 3.12 and 3.13.

It is worth noting that the platform's course information module integrates the open-source rich text editor, WangEditor [17]. This rich text editor allows administrators to modify text attributes such as format and color while editing course descriptions, and also supports uploading images and videos, using code blocks, and more. The steps for integrating this rich text editor are as follows:

1. Implement an image upload API for the rich text editor on the backend.

2. Import the WangEditor package in the frontend.

3. Incorporate the rich text editor into the frontend and initialize it when in use.

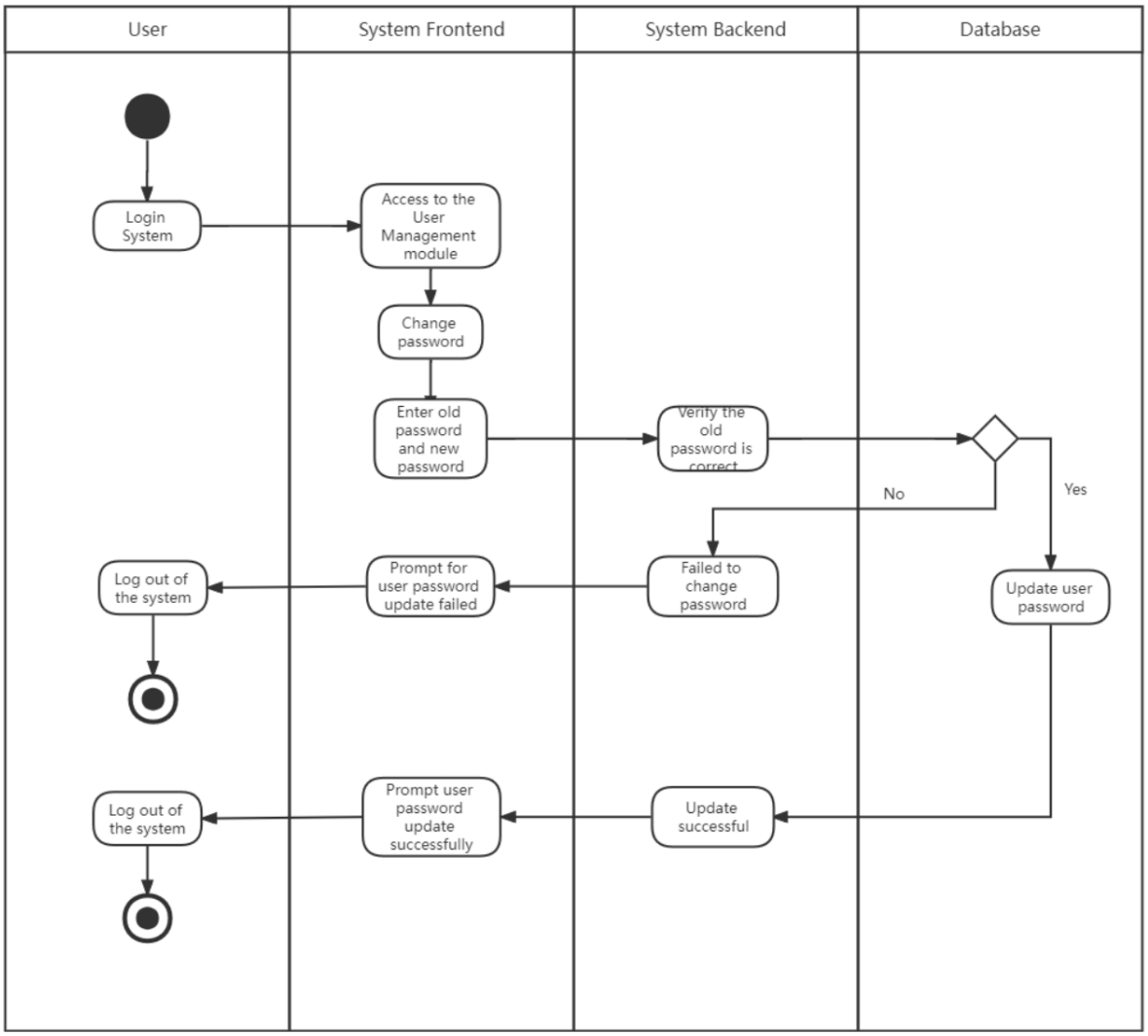

Figure 3.9: Activity Diagram for User Password Modification

4. Save the content after editing.

5. Render the content in the frontend.

The effect of the rich text editor is shown in Figure 3.14.

3.5.3 User Interface

Home Page Module

1. Frontend

The home page module of the system presents a visually appealing interface that combines menus, a carousel, and dynamic content areas such as online courses and resources. The key elements include:

- Navigation Bar and Header: The top of the homepage features a navigation bar and

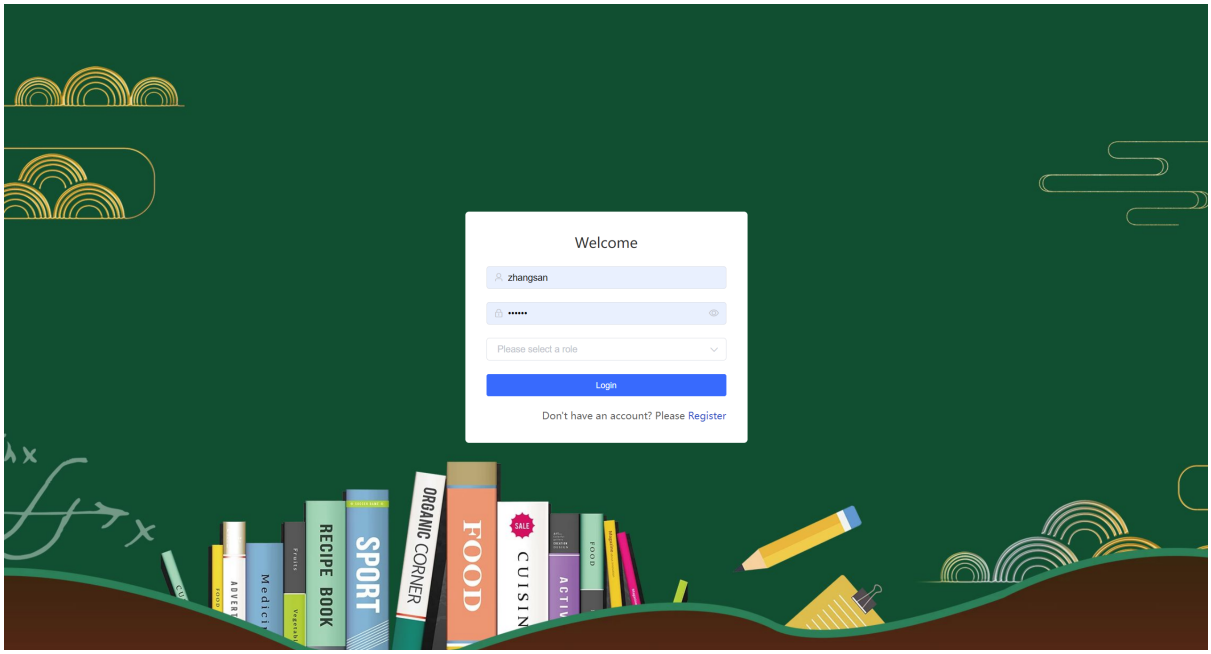

Figure 3.10: Login and Registration Page

header area, designed using Vue.js and Element UI, displaying the platform's logo, navigation links, and dropdown menus for user interaction.

- Carousel: A prominent carousel is located in a prime spot on the homepage, with each image in the carousel carefully selected as an advertisement.

- Course and Resource Display: The main content area dynamically showcases online courses and resources. Buttons such as "Video Courses," "Points Area," and "Image and Text Courses" allow users to browse various options. Each course or resource is represented by a thumbnail and a brief description, with periodic updates achieved by calling backend APIs.

2. Backend

The backend of the homepage module is responsible for serving the dynamic content visible on the frontend. It handles user requests and provides the corresponding data. Key functionalities include:

- Dynamic Content Loading: During initialization (using the mounted lifecycle hook in Vue.js), the frontend sends requests to backend endpoints like /course/getRecommend and /course/selectTop8 to retrieve data for the main carousel and course information.

- API Endpoints:

GET /getRecommend: Retrieves a single recommended course to be highlighted on the homepage.

GET /selectTop8: Retrieves the top 8 courses based on predefined criteria, displayed under

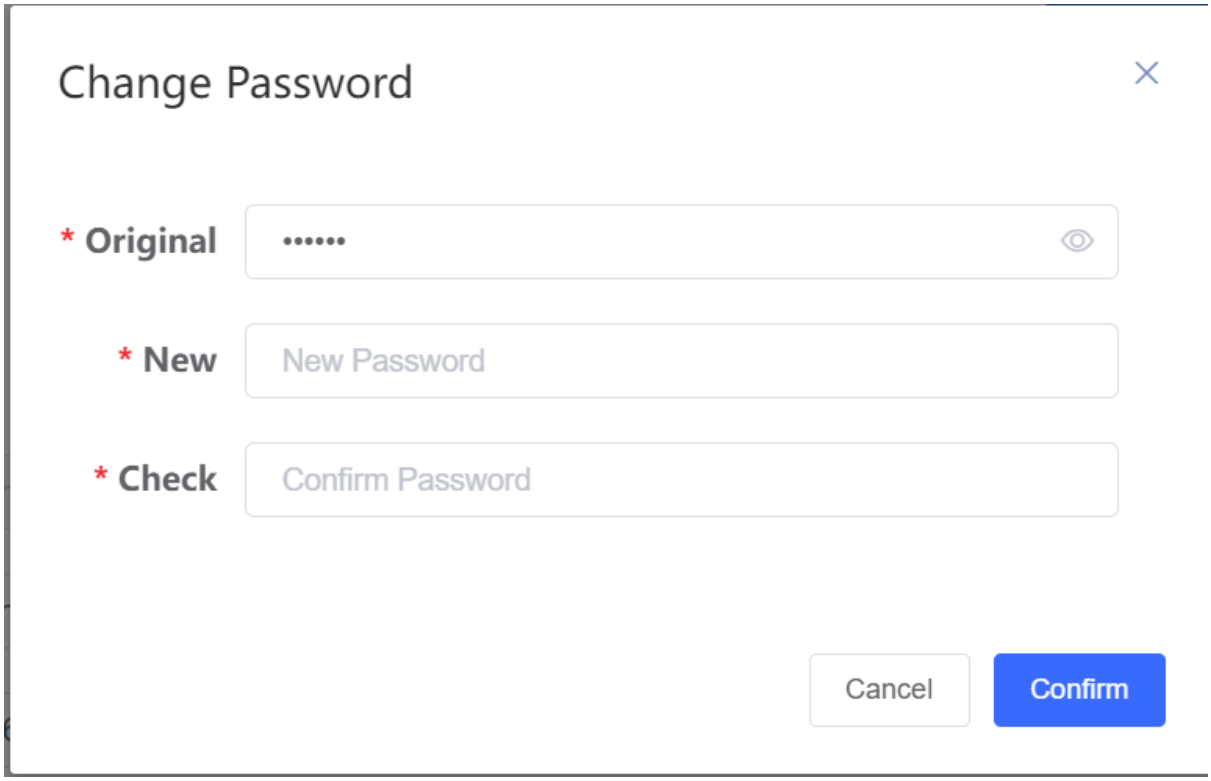

Figure 3.11: User Changing Password Page

different categories.

- Error Handling: Responses from these endpoints include status codes and messages. If there are errors in data retrieval or on the server side, mechanisms are triggered to provide immediate feedback to the user and log the specific errors in the system logs.

3. Database

The interaction between the homepage and the database determines the content displayed on the homepage. The operations include:

- Recommendation Logic: Database queries are used to fetch courses marked as recommended. For instance, an SQL query like SELECT * FROM course WHERE recommend = 'YES' can retrieve courses designated as recommended.

- Data Insertion and Updates: Insertion and update operations ensure that the course listings are up-to-date and accurate. Before a course is inserted or updated as recommended, the system checks to ensure no other courses are currently recommended, preventing duplicate recommendations.

- Structured Query Processing: The system uses prepared statements and transactions to manage database operations, enhancing security and efficiency.

Overall, the homepage module is designed to provide a seamless user experience through

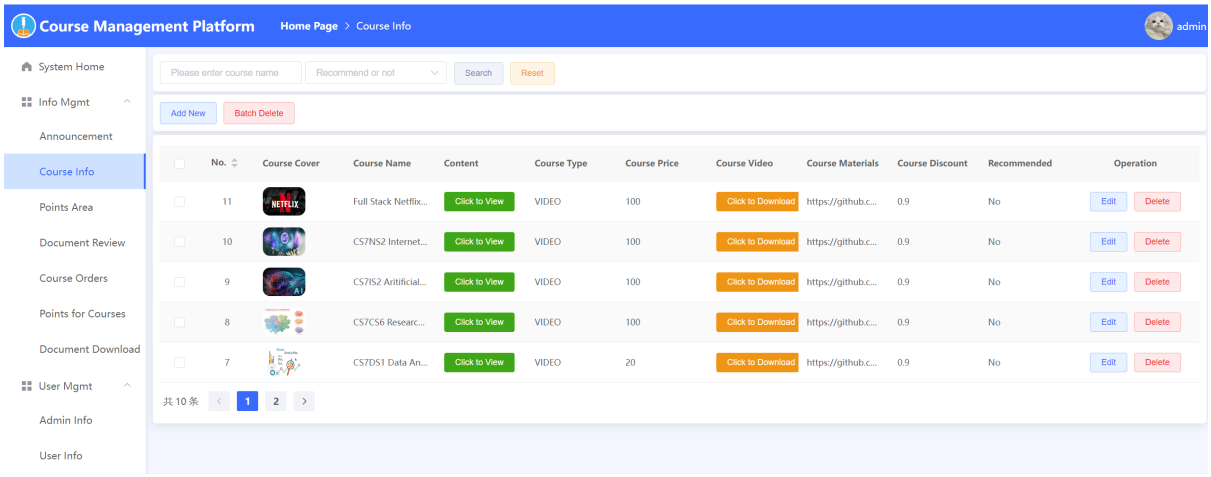

Figure 3.12: Course Information Module in the Admin Interface

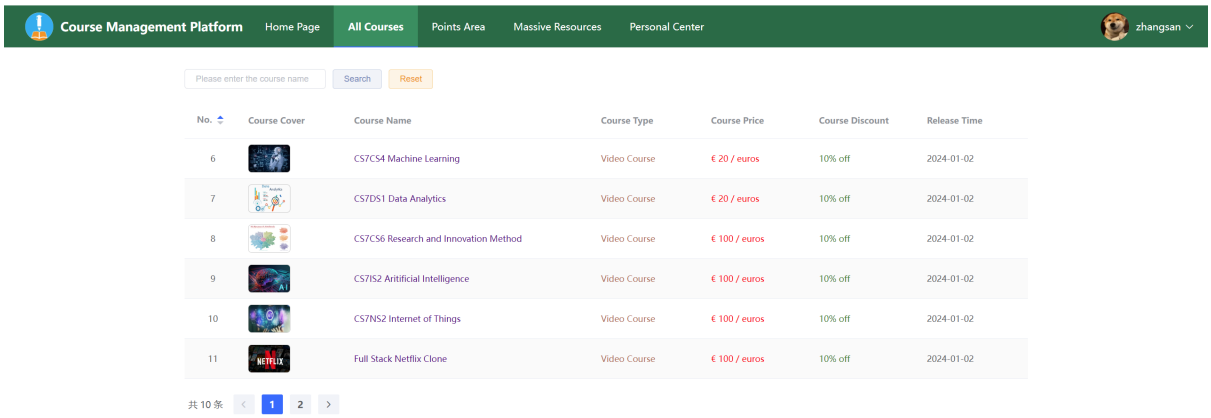

Figure 3.13: Course Information Module in the User Interface

effective frontend design, robust backend processing, and efficient database operations, ensuring that users have access to the most relevant and engaging content. The implementation effect of the Home Page is shown in Figure 3.15.

Three-type Course Switching Module

1. Frontend

At the system frontend, users can switch between different types of course displays through three buttons on the interface. These buttons are linked to click events (e.g., @click="initValue('VIDEO')"), allowing users to select video courses, points-based courses, or text and image courses. The following are the user interaction steps:

- Video Course Button: When the user clicks this button, the system sets the course type by executing the initValue('VIDEO') method, triggering data retrieval.

- Points Area Button: After clicking this button, initValue('SCORE') is executed to specifically handle the retrieval of points-related course data.

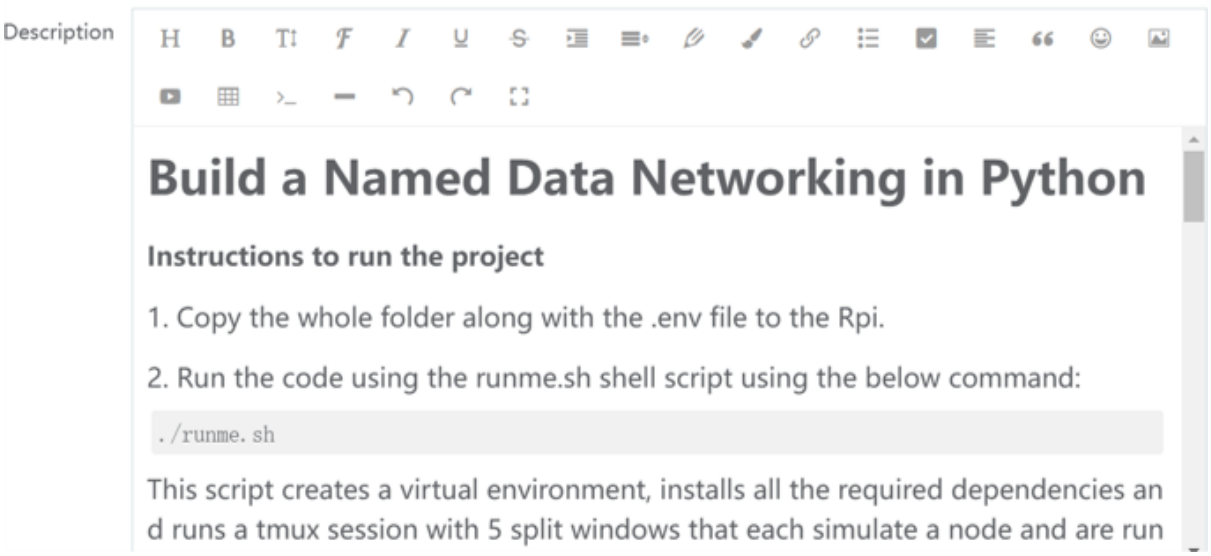

Figure 3.14: Rich Text Editor Renderings

- Text and Image Course Button: This button sets the course type to text and image through initValue('TEXT') and updates the display data.

The functionality of these buttons is supported by the initValue method, which first sets the course type and then calls the getData() method to fetch the corresponding data. The getData() method requests recommended courses and additional course lists from the backend based on the current course type.

2. Backend

In the backend, in response to frontend requests, the system processes and returns data through different API endpoints:

- Fetching Recommended Courses: Handled by @GetMapping("/getRecommend"), this endpoint processes requests for different types of recommended courses. Depending on the type parameter, it calls either courseService.getRecommend(type) or scoreService.getRecommend() to fetch recommended courses.

- Fetching Additional Courses: The @GetMapping("/selectTop8") endpoint manages requests for additional courses, returning the latest eight courses of the specified type.

These methods involve calling the corresponding service layer methods, which then interact with the data access layer to complete the retrieval and packaging of course information.

3. Database

Database operations are primarily carried out through SQL queries:

- Recommended Courses Query: A SQL query such as select * from course where

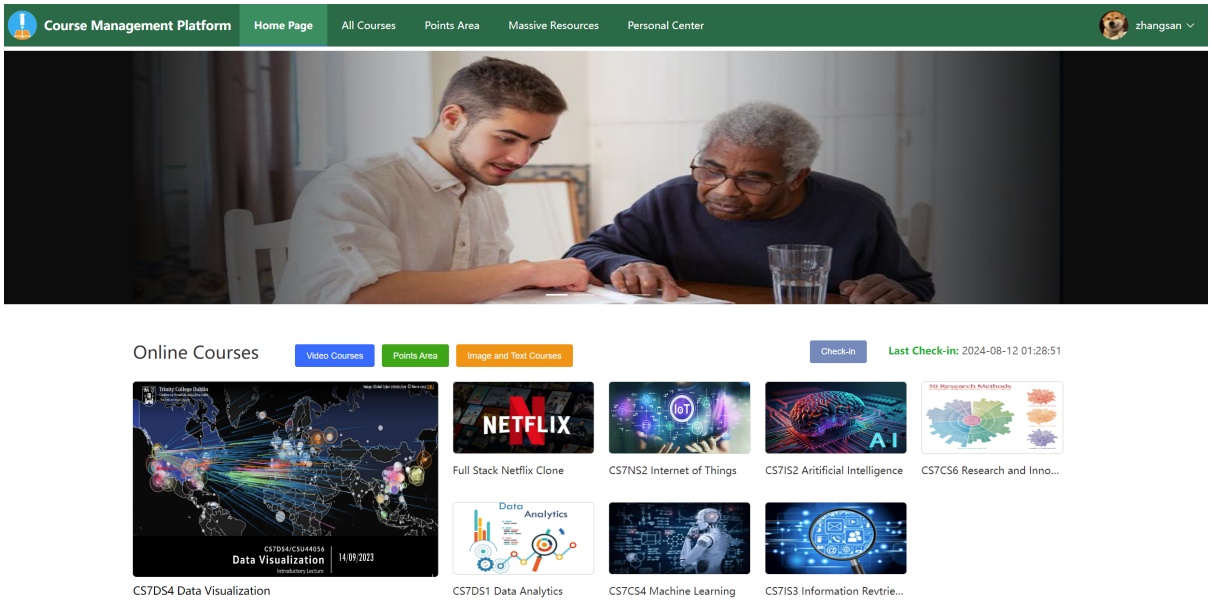

Figure 3.15: Implementation Effect Diagram of the Home Page

recommend = 'YES' and type = #type is used to fetch recommended courses of a specified type.

- Additional Courses Query: Queries like select * from course where recommend = 'NO' and type = #type order by id desc limit 8 are executed to obtain the latest eight courses that are not recommended.

Points Redemption and Course Unlocking Module

1. Frontend

Users engage in course redemption via points through the frontend interface. The frontend offers two main functionalities: order checking and course redemption.

- Check Orders

Users can verify whether they have already redeemed a specific course.

The system retrieves relevant order information by calling the /scoreorder/selectAll API, passing the user ID and course ID.

If a corresponding order is found, an indicator of redemption is displayed on the interface.

- Points Redemption

Before redeeming a course, the system displays the required number of points.

Upon user confirmation of redemption, the frontend sends a POST request to /scoreorder/add, including the course ID, required points, and user ID.

Following successful redemption, the system notifies the user of success and refreshes the course information and re-checks the order status to update the user interface.

2. Backend

The backend specifically includes the following processing logic:

- Add Order:

The service first verifies whether the user has sufficient points.

If points are insufficient, a custom exception is thrown, and the frontend captures and displays an error message.

When points are sufficient, an order number and record time are generated, then the order information is saved to the database.

Simultaneously, the user's points balance is updated.

- Query and Manage Orders:

APIs are provided for searching by ID, batch deletion, modification, and pagination.

These interfaces allow the frontend to retrieve, update, or delete order records according to different requirements.

3. Database

- Database Table Design:

The scoreorder table stores points redemption orders and includes fields such as order ID, points course ID, required points, redemption time, and user ID. Database operations, including inserting new records, updating records, and deleting records, are performed using the MyBatis framework.

- Data Access Objects (DAO):

Database-specific operations are implemented through methods defined by the ScoreorderMapper interface.

These operations include adding new records, deleting records, updating records, and querying records.

The distinction between the course purchasing and course unlocking modules and this module lies solely in the replacement of account points with account balance, thus the design and implementation of the course purchasing and unlocking modules are not further elaborated. The visual representation of the effects after implementing this module is shown in Figures 3.16 and 3.17 .

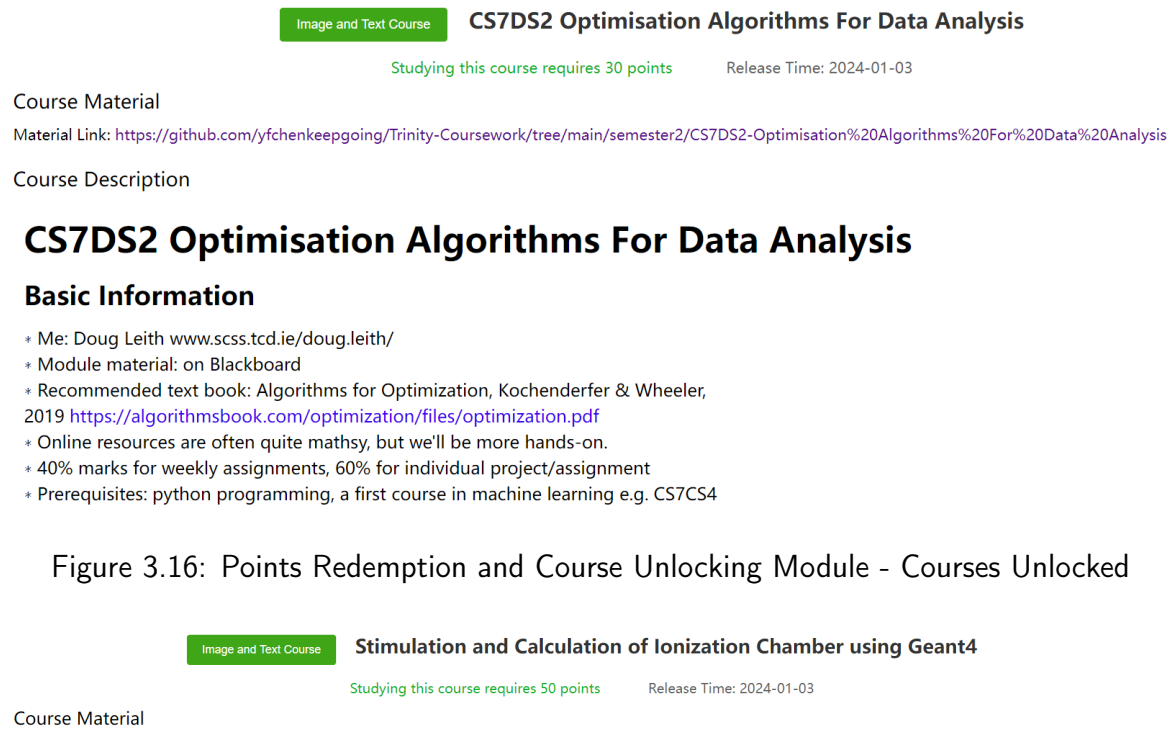

Figure 3.16: Points Redemption and Course Unlocking Module - Courses Unlocked

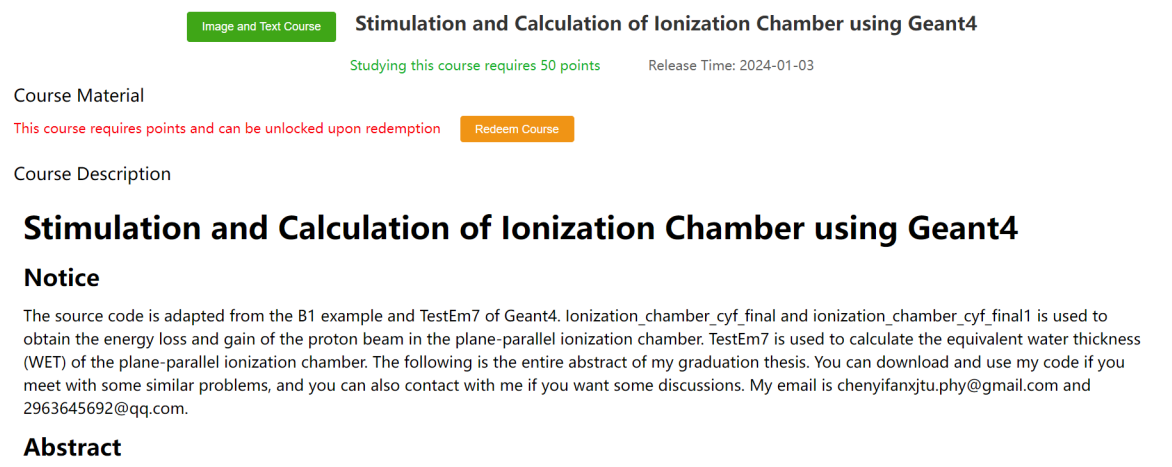

Figure 3.17: Points Redemption and Course Unlocking Module - Courses Locked

My Resources Module

1. Frontend

At the system frontend, users can manage their data through an intuitive interface. The frontend page includes multiple components such as buttons, input fields, data tables, and pagination controls, facilitating the easy addition, deletion, editing, and querying of data.

- Data Addition: Users can click the "Publish Data" button to open a form dialog where they can enter various details about the data, such as name, cover image, required points, etc., and submit it.

- Search and Reset: Users can search for data by entering its name in the input field or reset the search criteria by clicking the "Reset" button. The system displays relevant data based on the entered keywords.

- Edit and Delete Data: In the data table, each row of data is equipped with "Edit" and "Delete" buttons, allowing users to modify or delete data individually or in batches.

- Pagination Display: The data list supports pagination, enabling users to browse different pages of data by selecting page numbers, which enhances the user interaction experience.

2. Backend

The backend logic of the system consists of the service layer and controller layer, ensuring effective data flow and processing.

- Service Layer: Responsible for handling specific business logic, such as invoking mapper methods to operate on the database. The service layer processes functions like data addition, deletion, and querying, and calls utility classes for date formatting when appropriate.

- Controller Layer: Manages HTTP requests from the frontend, invokes methods from the service layer, and returns the results of operations. The controller layer defines multiple routes to handle different types of requests (such as GET and POST) and manages the validation and response of request data.

3. Database

Database operations are performed using MyBatis mappers, involving various CRUD operations on the information table.

- Creating and Updating Data: New or updated data records can be added to the database, including basic information about the uploaded data such as name, image, upload time, etc.

- Deleting Data: The database supports the deletion of individual data records by ID or the batch deletion of multiple records.

- Querying Data: Data can be queried in the database based on various criteria (such as data name, user ID, etc.), and supports pagination to accommodate the display of large volumes of data.

The effects of my Data Module are illustrated in Figure 3.18.

Online Resources Module

1. Frontend

In the frontend design of the Online Resources Module, the User Interface (UI) employs the Vue.js framework to dynamically display online materials. The frontend page layout utilizes Flexbox to achieve a responsive user interface. The main display area is divided into two

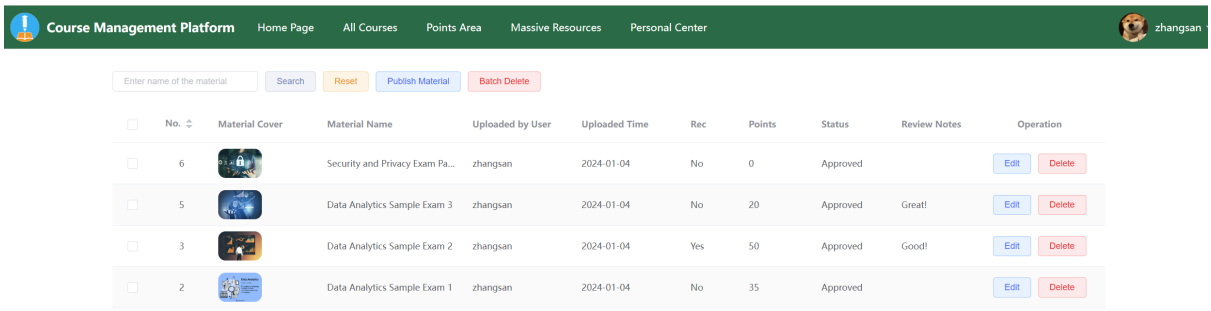

Figure 3.18: The visual representation of My Resources Module

parts: the left side lists multiple resource thumbnails and summaries, while the right side displays a recommended single resource.

- Resource List Rendering: The left side uses a loop to render each resource item in the list. The image and name of each resource are dynamically updated by binding to data properties of the Vue instance.

- Recommended Resource Display: The right side displays a specially recommended resource, handling potential undefined states through conditional rendering.

- Data Retrieval: The frontend interacts with the backend by invoking the encapsulated $request.get method to fetch both recommended resources and list resources. These operations are completed through asynchronous requests, with page content updated following data retrieval.

2. Backend

The backend, utilizing the Spring Boot framework, provides RESTful API interfaces to handle frontend requests. For the data requests of the Online Resources Module, the backend defines two main API interfaces:

- Fetching Recommended Resources: The route set by @GetMapping("/getRecommend") handles requests for recommended resources. This method invokes informationService.getRecommend(), retrieving resources marked as recommended from the database.

- Fetching Resource List: The route set by @GetMapping("/selectTop8") handles requests for a list of frontend resources. This method invokes informationService.selectTop8(), retrieving the latest eight non-recommended resource records from the database.

3. Database

Database operations are encapsulated and implemented using the MyBatis framework, primarily involving the querying of information resources.

- Recommended Resources Query: A SQL query statement defined by the @Select

annotation retrieves recommended resources marked as "YES" and records with a status of "review approved" from the information table.

- Resource List Query: Another SQL query statement defined by the @Select annotation retrieves the latest eight non-recommended resource records, which are sorted in descending order by ID to ensure the display of the newest available resources.

The visual representation of the effects after implementing the Online Resources Module is shown in Figure 3.19.

Figure 3.19: The visual representation of the Online Resources Module

Check-in Module

1. Frontend

The user interface provides a check-in functionality through a simple check-in button. This button is implemented using the Vue.js frontend framework. When users click the check-in button, it triggers a call to the backend /signin/add interface, sending the user's ID. Upon successful check-in, users receive a system notification indicating "Check-in successful, congratulations on earning 10 points." Simultaneously, the system updates to display the user's most recent check-in time. Additionally, upon page load, the /signin/selectByUserId interface is invoked to query and display the user's last check-in time, enhancing user experience.

2. Backend

In the backend, the check-in functionality is handled by the SigninController and SigninService classes. When the add method of SigninService is called:

- The system first retrieves the current date and checks if the user has already checked in on that day to prevent duplicate check-ins.

- If the user has not yet checked in, the system records the check-in time and date and stores this information in the database.

- If there is an existing record for the user, the system updates their check-in information.

- Upon successful check-in, the user immediately earns 10 points, accomplished by updating the points field in the user data table.

Furthermore, the SigninController provides RESTful API interfaces, including adding, deleting, and querying check-in records, supporting various data interaction needs of the frontend.

3. Database

The database includes a table named signin, which stores check-in information, including user ID, check-in time, and date. Through the MyBatis framework, the SigninMapper interface defines SQL operations related to the check-in data table, such as inserting new records, updating records, and querying a specific user's check-in records. Additionally, the system utilizes transaction management to ensure data consistency and integrity.

- When a user checks in for the first time each day, the system inserts a new record in the signin table.

- If a user attempts to check in repeatedly on the same day, the system will refuse the operation and return an error.

- Updates to the user's points are achieved by modifying the corresponding record in the user table.

The visual representations of both successful and failed check-in attempts are shown in Figures 3.20 and 3.21.

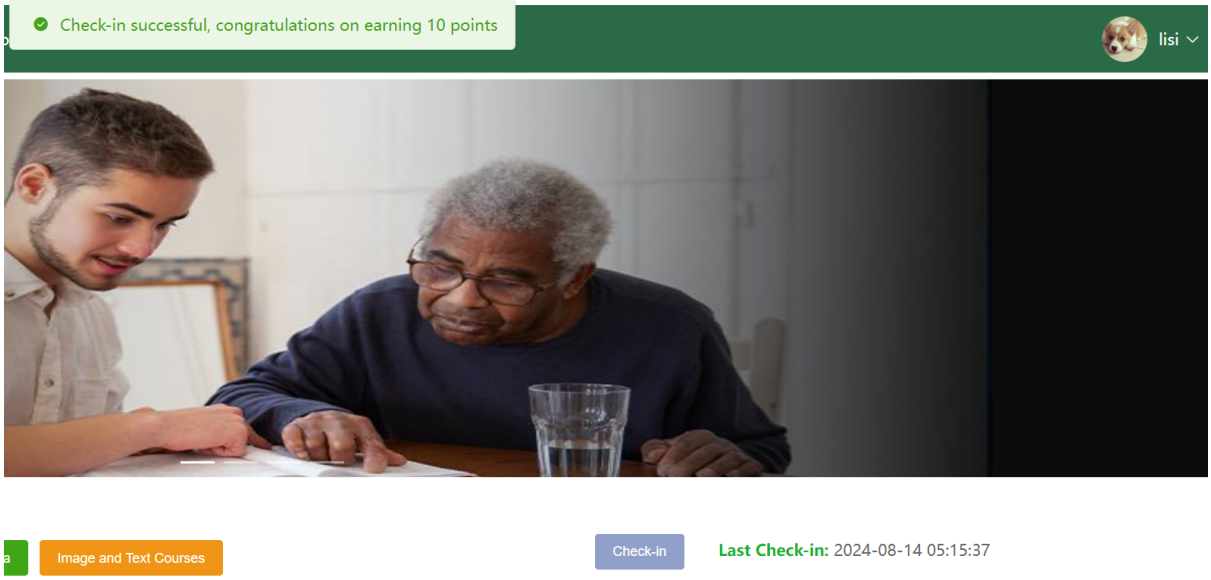

Figure 3.20: Visual Representation of a Successful Check-in

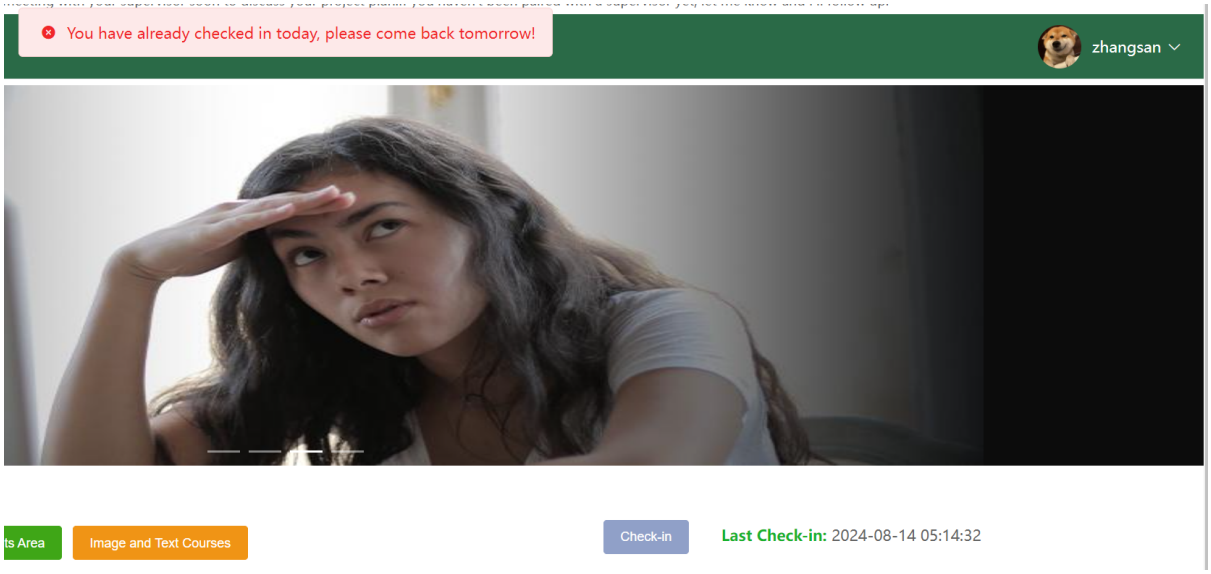

Figure 3.21: Visual Representation of a Failed Check-in

Mock Recharge Module

1. Frontend

On the personal center page, users can perform recharge operations through a recharge dialog. This dialog consists of a form that includes fields for entering the recharge amount, selecting the payment method, and confirming the recharge operation.

- Recharge Dialog Interface: An el-dialog component creates a modal dialog titled "Personal Recharge." The dialog contains a form el-form with the following elements:

Recharge Instructions: An el-form-item displays the recharge rules, noting that a one-time recharge of at least 500 euros automatically grants the user membership status.

Recharge Amount Input: An el-input allows users to enter the amount they wish to recharge.

Payment Method Selection: An el-radio component lets users choose between PayPal or Revolut as their payment method.

- Interactive Operations: After filling out the form, clicking the "Confirm" button triggers the recharge method to process the recharge, or the "Cancel" button to close the dialog.

2. Backend

When the user clicks the recharge button, the frontend sends a recharge request to the backend, where the following logic is implemented:

- Receiving Recharge Requests: The backend defines a recharge method that receives the

recharge amount from the frontend.

- User Authentication and Status Update: The system first retrieves user information using the current user's token, then updates the user's account balance based on the provided recharge amount.

- Membership Status Update: If the recharge amount reaches or exceeds 500 euros, the user's membership status is set to member.

- Persisting User Data: After updating the user's account balance and membership status, these changes are saved to the database.

3. Database

At the database level, the following key operations are involved:

- User Data Retrieval: Retrieve detailed information about the current user by user ID to enable updates.

- Update Operations: Update the user's account balance and membership status based on the recharge results. If the recharge amount meets the membership criteria, the membership field in the user record is updated to "yes."

- Data Consistency Assurance: All database operations must ensure data consistency and integrity, especially when updating financial data.

This module does not invoke real PayPal and Revolut interfaces; therefore, it simulates the actual recharge functionality. The visual representation of the effects after implementing this module is shown in Figure 3.22.

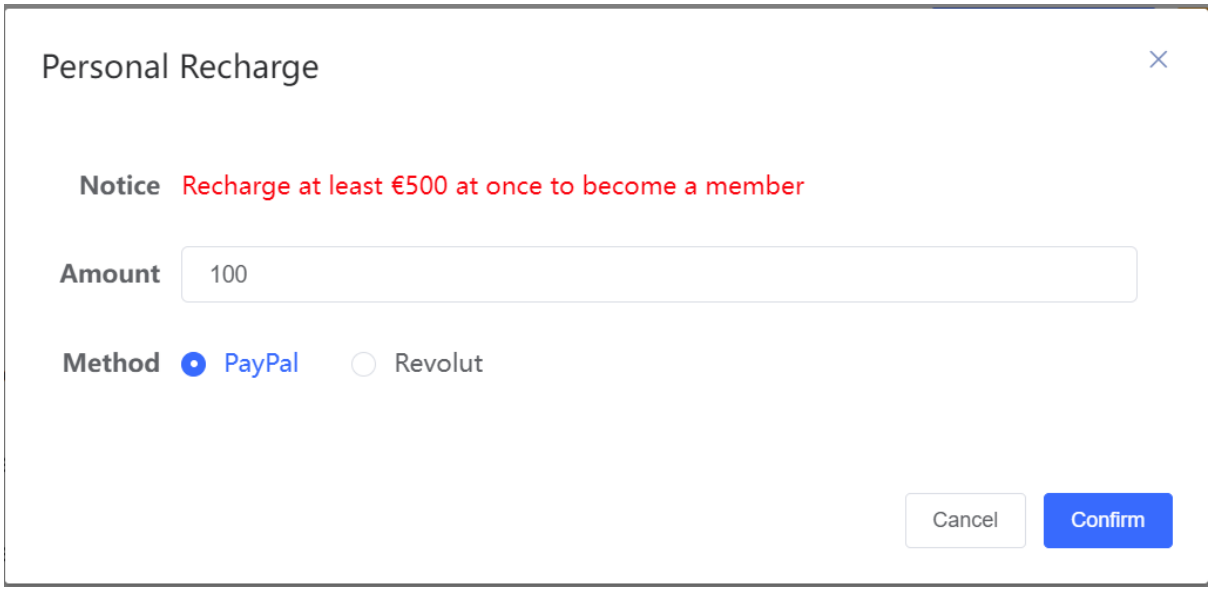

Figure 3.22: Visual Representation of the Mock Recharge Module

Course Order Module

1. Frontend

Users can browse and purchase courses through the frontend page of the system. A dedicated page on the user end lists the courses that have been purchased, displaying detailed information about each course in a dynamic table. This information includes the course cover, course name, course type, order price, order number, and order date. Users can navigate their order history using pagination controls. The course page provides direct links to each course's detail page, facilitating easy access to further course details.

The frontend page is built using the Vue.js framework and interacts with the backend services through asynchronous requests (AJAX) to fetch order data. Each operation on the order page, such as page navigation or filtering, triggers a request to the backend to ensure real-time data updates.

2. Backend

In the backend services, the order module primarily involves operations such as adding, querying, modifying, and deleting orders. The OrdersService class, implemented using the Spring Boot framework, handles specific business logic:

- Adding Orders: When a user completes a course purchase, the backend calculates the course price and checks if the user's account balance is sufficient. Order details, including user ID, course ID, order creation time, and price, are recorded in the database.

- Querying Orders: Users can access information about all or specific orders. The backend filters order data based on user ID and course ID.

- Modifying and Deleting Orders: Users have the authority to modify or cancel incomplete orders. The backend provides corresponding API interfaces to support modifications and deletions of orders.

3. Database

Order information is stored in a MySQL database, specifically in a table named orders. This table contains fields such as order ID, course ID, user ID, order price, order number, and order time.

- Database Table Design: The design includes an auto-incrementing primary key ID and foreign key references for course ID and user ID to ensure data integrity and consistency.

- SQL Operations: Includes basic CRUD operations. For example, adding an order inserts a new record, while deleting an order removes a record based on the order ID.

- Index Optimization: Indexes are set for commonly queried fields such as user ID and course ID to optimize query performance.

The visual representation of the effects after implementing this module is shown in Figure 3.23.

My Purchased Courses (6)

No.	Cover	Name	Type	Order Price	Order Number	Order Time
12		CS7DS3 Applied Statistcal Modelling	Image and Text Course	40	20240727032721	2024-07-27 03:27:21
11		CS7DS1 Data Analytics	Video Course	18	20240727024648	2024-07-27 02:46:48
10		CS7CS4 Machine Learning	Video Course	18	20240726034335	2024-07-26 03:43:35
9		CS7NS2 Internet of Things	Video Course	90	20240726033940	2024-07-26 03:39:40
8		Full Stack Netflix Clone	Video Course	90	20240726033903	2024-07-26 03:39:03
2		CS7CS6 Research and Innovation Method	Video Course	100	20240110151136	2024-01-10 15:11:36

共 6 条 1

Figure 3.23: Visual Representation of the Course Order Module

Course Comment Module

1. Frontend

The frontend of the Course Review Module is constructed using the Vue framework in conjunction with the Element UI library. The user interface includes functionalities such as video playback, course information display, comment display, and comment submission. Upon entering the course detail page, course information is initially loaded, and depending on the user's purchase status, the corresponding course materials and videos are displayed. If the course requires payment, users can click the purchase button to buy the course, which upon successful transaction, will unlock the related content.

The comment section allows users to view all comments and reply to specific ones. Users can enter their comments in a text box and post them by clicking the submit button. The system also supports the real-time display of comments, allowing users to see the most recent comments immediately after submission.

2. Backend

The backend services primarily consist of the CommentController and CommentService. The CommentService is responsible for handling the business logic related to comment data, such as adding, deleting, modifying, and querying comments. Comment timestamps are generated in the backend to ensure data consistency and accuracy.

The CommentController processes requests from the frontend, invokes methods of the CommentService to perform specific operations, and returns the results. These operations include the addition, deletion, querying, and modification of comments. All data interactions are completed through RESTful APIs, ensuring decoupling between the frontend and backend and standardization of interfaces.

3. Database

Comment data is stored in a database table named comment, which includes fields such as comment ID, user ID, course ID, comment content, and comment timestamp. Interactions with the database are facilitated using the MyBatis framework, with specific data operations implemented through SQL statements defined by the CommentMapper interface.

Database operations include inserting new comments, updating comments, deleting comments, and querying comments. Specifically, the querying operation supports multiple filtering conditions (such as course ID, user ID, etc.) and supports pagination to accommodate the processing of large volumes of data.

The visual representation of the effects after implementing this module is shown in Figure 3.24.

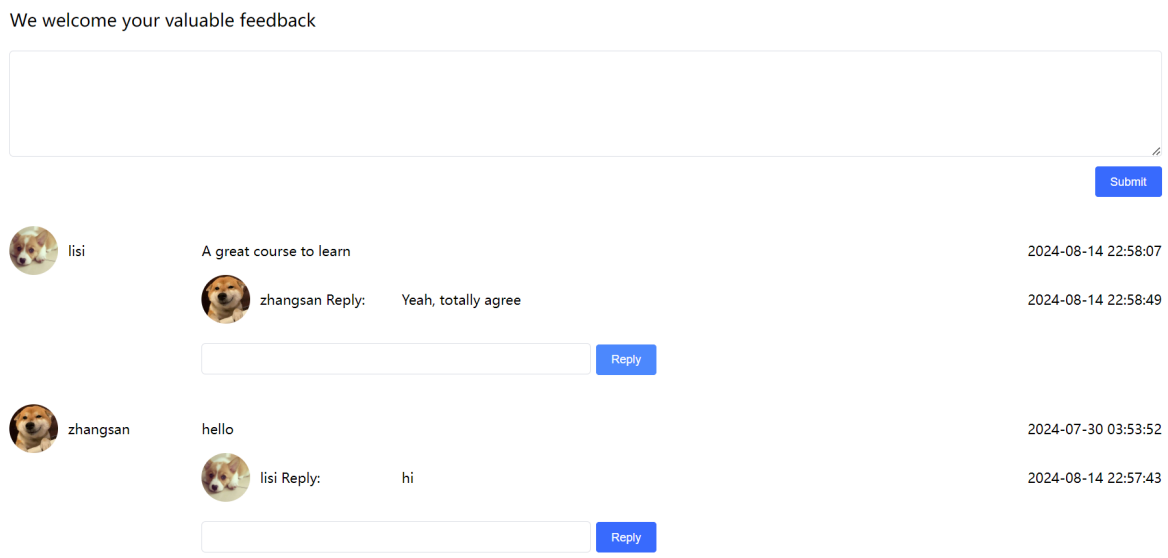

Figure 3.24: Visual Representation of the Course Comment Module

3.5.4 Admin Interface

Points Area Module

1. Frontend

Users interact with the Points Area module through a frontend interface constructed using

the Vue.js framework and Element UI component library. The primary functionalities include:

- Search and Filter: Users can search for courses by name and choose to display only recommended courses. This functionality is implemented via data binding using v-model for two-way data binding.

- Data Display: Course data is presented in a tabular format with pagination support. The table displays various information about the courses such as cover, name, content, and type.

- Add and Edit Courses: Users can enter course information through a form and add or edit via a dialog box. Form validation ensures the completeness of necessary information.

- Delete Courses: Users have the option to delete courses individually or in bulk. A confirmation dialog box appears before deletion to prevent accidental operations.

- Content Management: Users can view and edit course content using the WangEditor rich text editor to enhance content editing capabilities.

All these interactions are carried out through asynchronous requests to communicate with the server, ensuring the responsiveness and immediacy of the user interface.

2. Backend

The main processing logic in the backend includes:

- Course Management: Provides APIs for adding, deleting, modifying, and querying courses. The ScoreService class encapsulates business logic, handling the CRUD operations for course data.

- Data Validation: Data validation is performed at the backend during the addition and updating of courses, such as checking for existing recommended courses to prevent data conflicts.

- Pagination and Filtering: Utilizes the MyBatis pagination plugin PageHelper to implement data paging queries, optimizing the loading and display of large data volumes.

- Exception Handling: Employs a custom exception handling mechanism to capture and manage potential errors, returning clear error messages to the frontend.

3. Database

The main operations in the database include:

- Database Table Design: The score table stores course information, including fields such as course cover, name, type, and points.

- Data Access Layer: Utilizes the MyBatis framework's Mapper interface ScoreMapper to encapsulate database operations. SQL queries and commands are defined through XML configuration files to implement data persistence operations.

- Data Security and Integrity: Database tables are configured with primary keys and necessary indexes to ensure data uniqueness and query efficiency. Transaction control is used to ensure the integrity and consistency of data operations.

The only distinction between the Online Courses Module and this module is the replacement of account points with account balances; therefore, the design and implementation of the Online Courses Module are not further elaborated. The visual representation of the effects after implementing this module is shown in Figure 3.25.

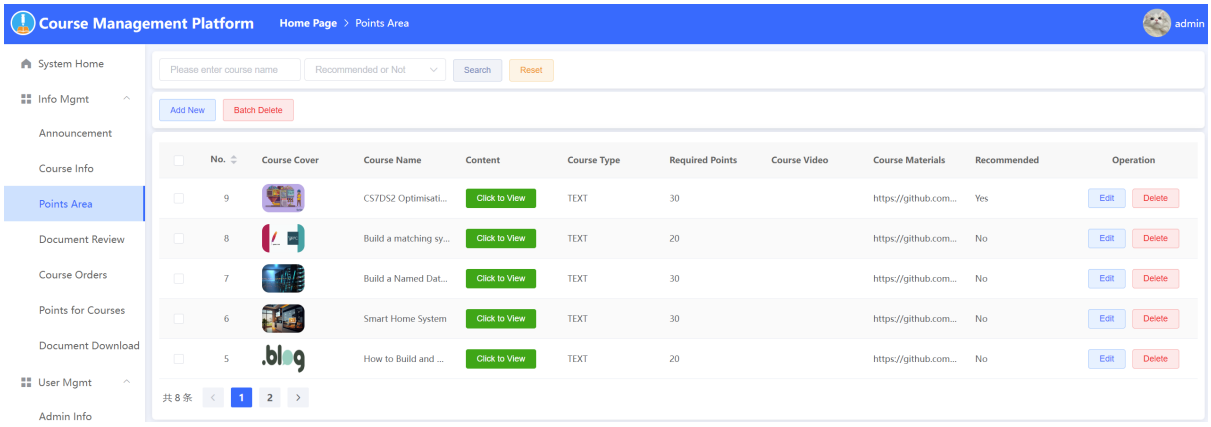

Figure 3.25: Visual Representation of the Points Area Module

Document Review Module

1. Frontend

This module's frontend allows users to search, review, and manage recommendations for materials. The user interface provides filtering options through a search box and dropdown menus, while displaying a detailed list of materials that includes information such as cover, name, description, uploading user, and upload time. For each material, users can click the "Review" button to access an editing interface to update the review status and recommendation options.

2. Backend

Backend processing includes receiving users' query requests, returning the corresponding list of materials, and handling updates for material reviews and recommendations. Notably, the system backend includes logical checks to ensure that changes in recommendation status do not violate the unique recommendation rules. If violated, the system will throw a custom exception and notify the user of the error.

3. Database

Database operations primarily involve the storage and management of material information, including basic details and statuses (such as review and recommendation statuses). The system uses SQL statements to perform data queries and updates.

The visual representation of the effects after implementing this module is shown in Figure 3.26.

Figure 3.26: Visual Representation of the Document Review Module

Data Statistics Module

1. Frontend

The system frontend utilizes ECharts tools to present various statistical charts, such as pie charts and bar graphs, providing intuitive data views. Users initiate data requests through the graphical interface, and the frontend application interacts with the backend via AJAX technology, dynamically loading and rendering statistical data. For example, the frontend page provides options for administrators to view statistical charts of platform course revenues or the proportion of platform membership users, with corresponding charts updating in real-time based on data returned from the backend.

2. Backend

The backend, implemented using the Spring Boot framework, defines RESTful API interfaces to respond to frontend requests. These interfaces access the database through the service layer, perform queries, and aggregate data, such as sales volumes of different types of courses or user distributions. For instance, the bar graph data interface aggregates the total number of various resources on the platform and returns related statistical results for use in frontend charts. The backend is responsible for data processing and transformation, ensuring that the frontend receives data in the correct format.

3. Database

Database operations include data storage, querying, and aggregation, managed by the backend service layer through an ORM framework. The database stores information such as courses and users, supporting complex data queries such as aggregating the total order prices by course type. These operations provide raw data for the system, which is then processed by backend logic to produce the chart data required by the frontend.

The visual representation of the effects after implementing this module is shown in Figure 3.27.

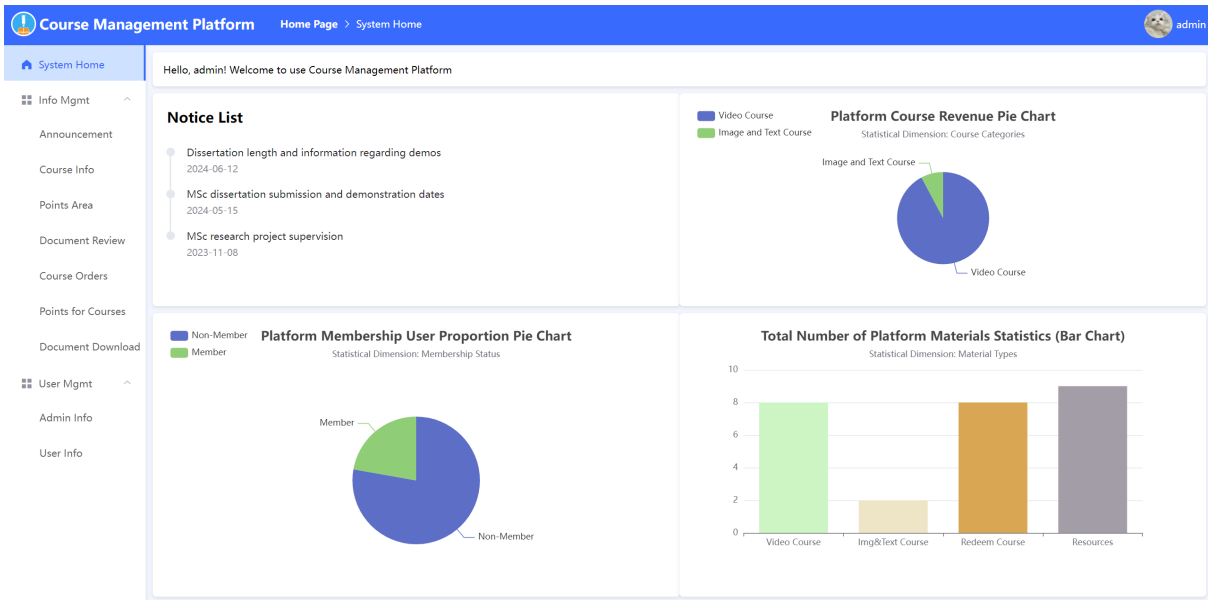

Figure 3.27: Visual Representation of the Data Statistics Module

Points Area Order Management Module

1. Frontend

The interface design for the administration side enables efficient querying and management of point orders. The interface includes a query module, featuring an input box and two buttons (search and reset), as well as an order data display module equipped with pagination functionality.

- Query Module: Administrators can perform searches by entering order numbers, using the search button to initiate the search or the reset button to clear entered search criteria and reload all data.

- Data Display Module: Search results are displayed in a table, including information such as course cover, course name, required points, order number, redemption time, and the redeeming user. The table supports pagination, enhancing the efficiency of data browsing and management.

2. Backend

The system backend is responsible for handling requests from the frontend, executing business logic, and interacting with the database.

- Pagination Query Handling: The backend retrieves relevant data from the database based on the incoming pagination parameters (page number and page size) and order number.

- Data Response: After processing, the backend returns the query results to the frontend for use in the table module.

- Reset Function Implementation: When an administrator clicks the reset button, the backend clears the current search criteria and reloads the initial page data.

3. Database

The database design supports the needs of frontend display and backend logic, with key operations including data querying and management.

- Database Table Design: The order table stores key information such as order numbers, user IDs, course IDs, and redemption times. Additionally, the user and course tables support the order table by providing complete order details through related queries.

- Query Operations: Queries are executed based on order numbers; if no number is specified, all order data is loaded. The query operation takes pagination needs into account, using SQL statements to precisely control data retrieval.

The visual representation of the effects after implementing this module is shown in Figure 3.28.

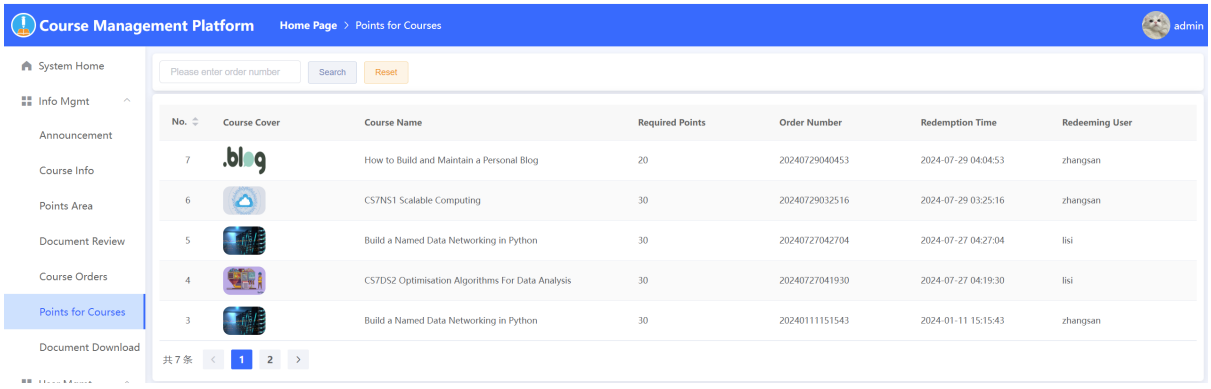

Figure 3.28: Visual Representation of the Points Area Order Management Module

Massive Resources Order Management Module

1. Frontend

Users can perform points redemption and resource unlocking operations through the system's frontend. The specific process is as follows:

- Points Redemption: Users select the resources they wish to redeem and initiate the process by clicking the "Redeem" button. The system first checks if the user has enough points for the redemption. If there are sufficient points, the system completes the redemption, deducts the appropriate points, and updates the resource's unlocked status. Users then receive a notification of successful redemption.

- Resource Unlock Check: The system regularly checks the status of unlocked resources to confirm whether users have access to these resources. Such checks are conducted each time a user accesses a resource.

- Historical Downloads View: Users can view records of all resources they have historically unlocked, including resource name, redemption points, order number, and redemption time, facilitating easy management and review.

2. Backend

The backend services are primarily responsible for handling user requests, including resource redemption and queries, as well as the deduction and awarding of points. The detailed process includes:

- Resource Redemption Processing: When a user initiates a redemption request, the backend first verifies whether the user has sufficient points. If the points are insufficient, an error is returned; if sufficient, the user's points are deducted, a new order is generated, and the original author of the resource is awarded points.

- Order Management: The backend provides complete order management functionality, including adding new orders, deleting orders, modifying order information, and querying orders.

- Pagination Query Function: The backend supports pagination queries for order data, making it easier for users to manage large volumes of order data.

3. Database

The database design is foundational to the efficient operation of the entire system. Here are the main database operations:

1. Creating Resource Order Table: The database includes a specific table to store resource order information, including order number, user ID, resource ID, redemption points, and redemption time.

2. Operation Definitions: Various database operations are defined in the MyBatis mapping files, including inserting new orders, updating order information, deleting orders, and

querying orders.

3. Data Association Queries: To retrieve information about resources and users simultaneously when querying orders, the system has designed complex SQL queries that achieve data association through multi-table joins.

The visual representation of the effects after implementing this module is shown in Figure 3.29.

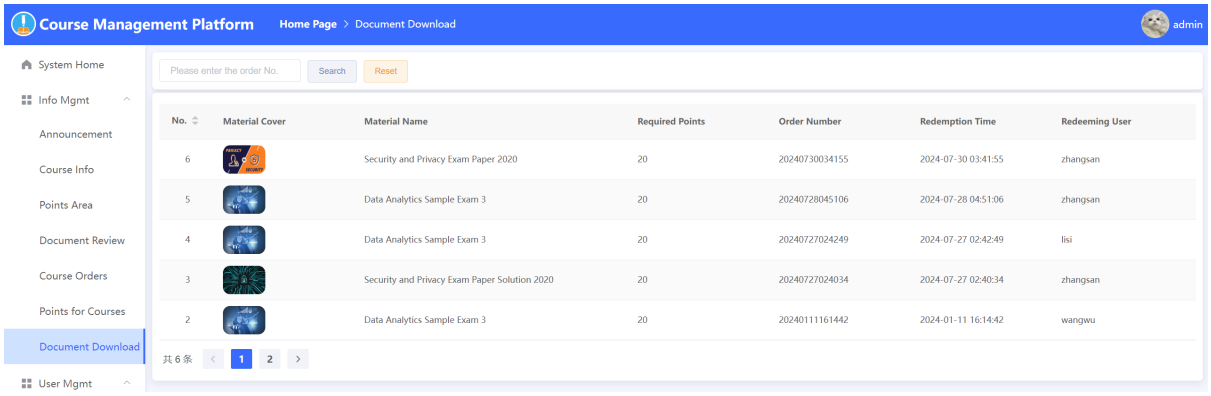

Figure 3.29: Visual Representation of the Massive Resource Order Module

4 Results

4.1 Deployment

Prior to system testing, the online course management platform must first be deployed. This platform can be deployed on a local computer and accessed via localhost and a port number; alternatively, it can be deployed on a remote server (such as AWS or Azure) [18], and accessed via a domain name or the remote server's IP address and port number. For convenience and cost efficiency, I have chosen to deploy this project on my local computer. Once deployed, the platform is tested by opening a browser and navigating to http://localhost:8080/. The testing environment is on a Windows 11 system using the Chrome browser, version 119.0.6045.205.

4.2 System Testing

The testing consists of three parts. The first part involves testing the regular user module, the second part focuses on testing the administrator module, and the third part examines the interaction between the regular user and administrator modules [19]. The results of the tests for all three parts will be summarized in a series of tables, while the actual screenshots of the tests will be included in the appendix.

4.2.1 Testing of the Standard User Module

Login and Registeration Module

The screenshots of the test results for this module are shown in Figures 1 to 3 in the appendix, with the testing information summarized in table 4.1. As indicated by the table, the functionality of the login and registration module is fully operational and meets the expectations.

Table 4.1: Login and Registeration Module Testing

Testing Items	Testing Data	Testing Results
Incorrect user login information	The username is zhangsan, and the password is 123456	User login failed, incorrect username or password
Duplicate username on registration	Username already existed as zhangsan	The username already existed, and registration had failed
Registration successful	Unregistered username	Registration successful, user can use the account to sign up

My Resources Module

The screenshots of the test results for this module are displayed in Figures 4 to 6 in the appendix, with the testing information summarized in table 4.2. As indicated by the table, the functionality of the My Resources module operates fully normally and meets expectations.

Mock Recharge Module

The test results for this module are shown in Figures 7 to 10 in the appendix, with the test information summarized in table 4.3. According to the table, if a single recharge amount is less than 500 euros, the account balance will increase by the amount recharged, but the membership status remains non-member. If the recharge amount is 500 euros or more, not only will the account balance increase by the amount recharged, but the membership status will also change to member, allowing all members to enjoy discounted prices on course and material purchases. The functionality of the mock recharge module is fully operational and meets expectations.

Purchasing Courses and Course Unlocking Module

The test results for this module are illustrated in Figures 11 to 12 in the appendix, with the testing information summarized in table 4.4. The functionality of the purchasing courses and course unlocking module is fully normal and meets expectations. Similar tests have also been conducted for the points exchange and course unlocking modules, and their functionality aligns with expectations as well. The only difference between the two modules is that the account balance is replaced by account points; therefore, the test process for the

Table 4.2: My Resources Module Testing

Testing Items	Testing Data	Testing Results
Publish material	Input cover, name, points, link and description of a new material	Click "confirm," and the frontend displays "save successful." The information of the newly published materials is rendered on the frontend page.
Search for material name	Enter "data" in the search box	Display all materials whose material name contains "data"
Edit	Click "Edit" and randomly modify one or more of the following information: cover, name, points, link, and description.	Click "confirm," and the frontend displays "save successful." The modified information will be rendered in real-time on the frontend page

other module is not reiterated here.

4.2.2 Testing of the Admin Module

For the administrator module, most of its submodules are designed, implemented, and function similarly to those in the regular user module; thus, the distinctive Points Area submodule was selected for testing. The testing of other submodules will not be reiterated.

Points Area Module

The screenshots of the test results for this module are shown in Figures 13 to 19 in the appendix, with the testing information summarized in table 4.5. As indicated by the table, the functionality of the Points Area module is fully operational and meets expectations. For the Online Courses module, the only difference from the Points Area module is the replacement of user points with user balances, hence the testing process for the Online Courses module will not be further detailed.

4.2.3 Interoperability Testing

Some module tests on this platform involve the interaction between the regular user module and the administrator module. Specifically, this means that modifications in one module are

Table 4.3: Mock Recharge Module Testing

Testing Items	Testing Data	Testing Results
Number of recharges less than 500 euros per transaction	Recharge 100 euros in one transaction	Display "Recharge Successful," membership status remains non-member, balance increases by 100 euros
Number of recharges greater than or equal to 500 euros per transaction	Recharge 500 euros in one transaction	Display "Recharge Successful," membership status changes to member, balance increases by 500 euros

synchronized to another module, or they affect the content displayed on another module. The linkage tests primarily involve three modules: the Recommendations Module, the Orders Module, and the Materials Review Module, and we will now present the test plans and results for these three modules.

Recommendation Module

This module is a part of the administrator module. On the administrator interface's Course Info, Points Area, and Document Review pages, an administrator can select to recommend or not recommend a course or resource. Once a course or resource is recommended, it is prominently displayed in a large image format on the regular user interface. The primary purpose of this functionality is to prominently recommend certain courses and resources to users, placing these in eye-catching advertising slots. It is important to note that the platform offers four types of courses or resources: video courses, points area, image & text courses, and online resources. Within each category, only one course or resource can be recommended, as there is only one advertising slot per category. If multiple courses or resources are recommended simultaneously within a category, an alert message will be displayed on the administrator interface.

The screenshots of the test results for this module are shown in Figures 20 to 22 in the appendix, with the test information summarized in table 4.6. As indicated by the table, the functionality of the recommendation module operates fully normally and meets expectations.

Table 4.4: Purchasing Courses and Course Unlocking Module Testing

Testing Items	Testing Data	Testing Results
Account balance is sufficient, attempt to purchase a course	Account balance is 600 euros, attempt to purchase a course valued at 100 euros	Purchase successful, course unlocked, account balance decreased by 100 euros
Account balance is insufficient, attempt to purchase a course	Account balance is 20 euros, attempt to purchase a course valued at 100 euros	Purchase failed, course not unlocked, account balance remains unchanged

Order Module

Standard users can use points or euros to purchase courses or resources on the user interface. There are three types of courses or resources available for purchase: online courses, which require euros; points area courses, and resources, both of which require points for exchange. Whenever a user purchases any of these three types of resources, the order information, such as purchase time, buyer, and course name, is recorded on the administrator interface for the administrator's review. Now, taking the purchase of online courses as an example, I will test the functionality of the orders module to see if the order information is immediately synchronized to the administrator interface after a user makes a purchase on the user interface.

The screenshots of the test results for this module are displayed in Figures 23 to 24 in the appendix, with the test information summarized in table 4.7 above. As indicated by the table, the functionality of the orders module is fully operational and meets expectations. The synchronization of order information when users purchase points area courses or resources is identical to that for online courses, thus further explanation is unnecessary.

Document Review Module

When a user uploads materials in the My Profile module, the relevant information of that material is immediately synchronized to the administrator interface's document review module. If the administrator has not reviewed it, the review status defaults to pending. If the administrator has reviewed it, the review status is updated to approved or not approved. Only materials with an approved status are displayed in the user interface's massive resources module, available for other users to browse and download. Next, we will test the

document review module.

The screenshots of the test results for this module are shown in Figures 25 to 26 in the appendix, with the test information summarized in table 4.8. As indicated by the table, the functionality of the document review module operates fully normally and meets expectations.

Table 4.5: Points Area Module testing

Testing Items	Testing Data	Testing Results
Add new courses	Input cover, name, type, rec, points, video, link, and description	Click "confirm," and the frontend displays "save successful." The information of the newly published courses is rendered on the frontend page.
Search for material name	Enter "build" in the search box	Display all courses whose course name contains "build"
Edit	Click "Edit" and randomly modify one or more of the following information: cover, name, type, rec, points, video, link, and description.	Click "confirm," and the frontend displays "save successful." The modified information will be rendered in real-time on the frontend page
Filter courses based on whether they are recommended	Enter "build" in the search box. In the "recommended or not" filter box, select "yes" and "no" respectively	When "yes" is selected, the search results return empty because none of the courses with "build" in their name are recommended. When "no" is selected, the search results return three courses, which are exactly all the courses that contain "build" in their name
Batch delete	Randomly select three courses and click "batch delete"	The three selected courses are deleted with one click
Click to view	Randomly select a course and click "click to view"	The administrator can view the entire content of the course

Table 4.6: Recommendation Module Testing

Testing Items	Testing Data	Testing Results
In each category of courses or resources, only one course or resource can be recommended	Recommend "CS7DS4 Data Visualization" in the Video Courses section	Display "CS7DS4 Data Visualization" prominently with a large image in the Video Courses section of the user interface
In each category of courses or resources, multiple courses or resources are recommended simultaneously	Recommend both "CS7DS4 Data Visualization" and "Full Stack Netflix Clone" in the Video Courses section	Unable to recommend "Full Stack Netflix Clone," an alert message on the administrator interface states: "Recommendation already exists, please remove the current one before recommending another"

Table 4.7: Order Module Testing

Testing Items	Testing Data	Testing Results
After a user purchases an online course on the user interface, will the order information be immediately synchronized to the administrator interface?	User zhangsan1 purchased "CS7CS4 Machine Learning" from the online courses module	The order information is immediately synchronized to the administrator interface, and the specific order can be identified by observing the Order Time and Ordering User, confirming that user zhangsan1 purchased "CS7CS4 Machine Learning"

Table 4.8: Document Review Module Testing

Testing Items	Testing Data	Testing Results
After a user uploads materials, an administrator reviews them	If the administrator has not yet reviewed or is in the process of reviewing, the review status is "pending."	In the massive resources module, the materials are not displayed
After a user uploads materials, an administrator reviews them	If the administrator approves the review, the status is "approved"	In the massive resources module, the materials are displayed
After a user uploads materials, an administrator reviews them	If the administrator does not approve, the status is "not approved"	In the massive resources module, the materials are not displayed

5 Conclusion

This paper presents the development and implementation of a fully open-source course management platform. The platform utilizes Spring Boot for backend operations and Vue for the frontend, achieving complete separation between the front and back ends. Comprehensive system tests have been conducted, with results confirming that all modules function as expected. The project introduces several innovative features:

1. Complex Multidimensional Categorization Design: Courses are categorized into two types—text and image, and video—further divided into paid courses and open courses, with different pricing for members and non-members. The user interface supports viewing from multiple dimensions.

2. Dual Role Design: Two roles are defined, administrator and user, with users further classified into regular and member users. Members enjoy discounts provided by the platform.

3. Integrated Rich Text Editor: The editor supports format adjustments, image uploads, and code block rendering within its interface.

4. Support for Online Video Streaming and Downloading: This functionality enables real-time media access and offline availability.

5. User Check-in Functionality: Each user can check in once per day, earning 10 points upon successful check-in. The backend logic for this feature is relatively complex.

6. Restricted Access to Course Content: Only introductory content such as course descriptions is displayed before payment or point redemption. Full course content, including videos and materials, is made accessible only after payment or redemption.

7. Usage of ECharts for Data Analysis: This tool allows administrators to analyze platform resource data across different modules effectively.

6 Future Work

Although the functionality of this course management platform is already well-developed and addresses most of the pain points existing in current course management platforms, there are areas where enhancements could further improve deployment ease, performance, and big data handling.

Ease of Deployment: Utilizing Docker and Kubernetes for software packaging, deployment, and management to ensure software availability and scalability. These tools provide automated deployments, scalable service management, and consistent environments, significantly simplifying the deployment process and reducing issues caused by configuration differences across environments.

Performance Enhancement: Employing the in-memory database Redis to store frequently accessed data that does not require persistence, such as tokens, to enhance the system's queries per second (QPS). As a high-performance key-value store, Redis offers rapid data read/write capabilities suitable for handling a large volume of read and write requests, thus improving the overall system response speed and processing capacity [20].

Big Data Handling: Using object storage services, such as Amazon S3 or Google Cloud Storage, to store large volumes of video files. Considering the use of distributed file systems like HDFS or some NoSQL databases for storage solutions, these technologies are well-suited for processing and storing large-scale datasets, supporting data high availability and flexibility.

Future developments of this course management platform should build upon the completed work and further enhance these three areas. With additional refinements, I believe this platform can move towards commercialization, providing extensive services for course management and online learning to a broad user base.

Bibliography

[1] A. Yuen, R. Fox, A. Sun, and L. Deng, "Course management systems in higher education: Understanding student experiences," *Interactive Technology and Smart Education*, vol. 6, no. 3, pp. 189–205, 2009.

[2] E. N. Ekwonwune and D. C. Edebatu, "Design and implementation of an online course management system," *Journal of Software Engineering and Applications*, vol. 12, no. 2, pp. 21–33, 2019.

[3] A. Alameen and B. Dhupia, "Implementing adaptive e-learning conceptual model: A survey and comparison with open source lms," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 14, no. 21, pp. 28–45, November 2019. [Online]. Available: https://www.learntechlib.org/p/217206

[4] H. e Fatima, "Open source lms: Top platforms for 2024," https://arbisoft.com/blogs/open-source-lms-top-platforms-for-2024, 2024, [Accessed 16-08-2024].

[5] A. Adedoyin, F. Enebe, R. Oyekunle, and N. Balogun, "Design and implementation of an online teaching and learning management system," *FUDMA Journal of Sciences*, vol. 7, no. 1, pp. 148–155, 2023.

[6] V. Tanzu, "Spring boot," https://spring.io/projects/spring-boot, 2024, accessed: 2024-08-16.

[7] "Mybatis," https://mybatis.org/mybatis-3/, 2024, accessed: 2024-08-16.

[8] baomidou, "Mybatis plus," https://mybatis.plus/en/, 2024, accessed: 2024-08-16.

[9] E. You, "Introduction to vue.js," https://vuejs.org/guide/introduction.html, 2024, accessed: 2024-08-16.

[10] Meta, "React (javascript library) - wikipedia," https://en.wikipedia.org/wiki/React_(JavaScript_library), 2024, accessed: 2024-08-16.

[11] ElemeFE, "Element - a desktop ui toolkit for web," https://element.eleme.io/#/en-US, 2024, accessed: 2024-08-16.

[12] M. AB, "Mysql - wikipedia," https://en.wikipedia.org/wiki/MySQL, 2024, accessed: 2024-08-16.

[13] Jetbrains, "Jetbrains idea 2024," https://www.jetbrains.com/idea/, 2024, accessed: 2024-08-16.

[14] P. C. Ltd, "Navicat premium 17," https://en.wikipedia.org/wiki/Navicat, 2024, accessed: 2024-08-16.

[15] Oracle, "Java8," https://en.wikipedia.org/wiki/Java_version_history#Java_8, 2024, accessed: 2024-08-16.

[16] O. Foundation, "Node.js 16," https://nodejs.org/en, 2024, accessed: 2024-08-16.

[17] wangeditor team, "wangeditor4," https://www.wangeditor.com/v4-en/, 2024, accessed: 2024-08-16.

[18] Y. Yi and Y. Liu, "Design and implementation of course review system," in *Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering*, 2022, pp. 137–142.

[19] Y. Peng, N. Liu, Y. Li, and Z. Shao, "Design and implementation of the online course registration system at tsinghua university," in *2012 International Conference on Systems and Informatics (ICSAI2012)*. IEEE, 2012, pp. 1179–1182.

[20] Y. Liu and M. Shabaz, "Design and research of computer network micro-course management system based on jsp technology," *International Journal of System Assurance Engineering and Management*, vol. 13, no. Suppl 1, pp. 203–211, 2022.

A1 Appendix

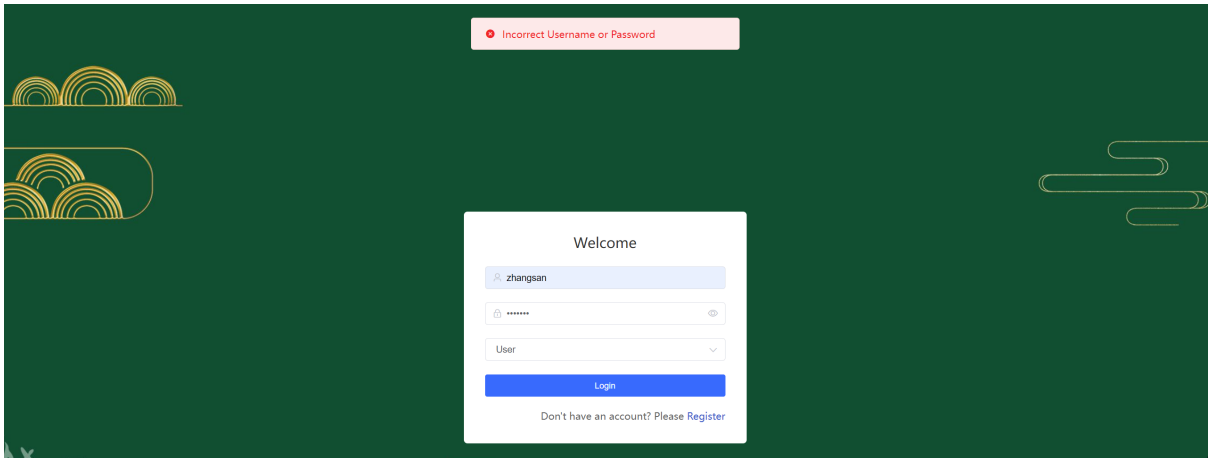

Figure A1.1: Login failed

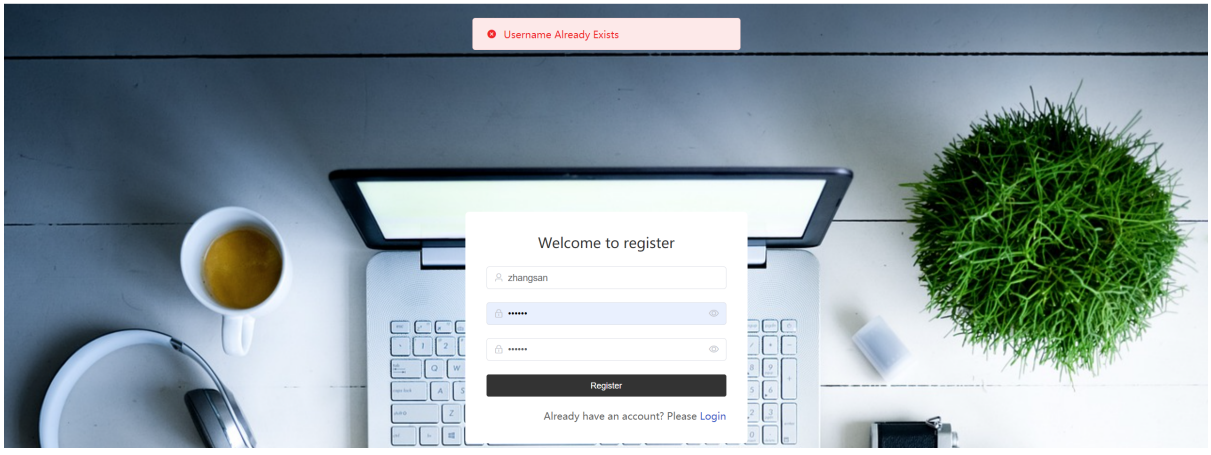

Figure A1.2: Sign-up Failed

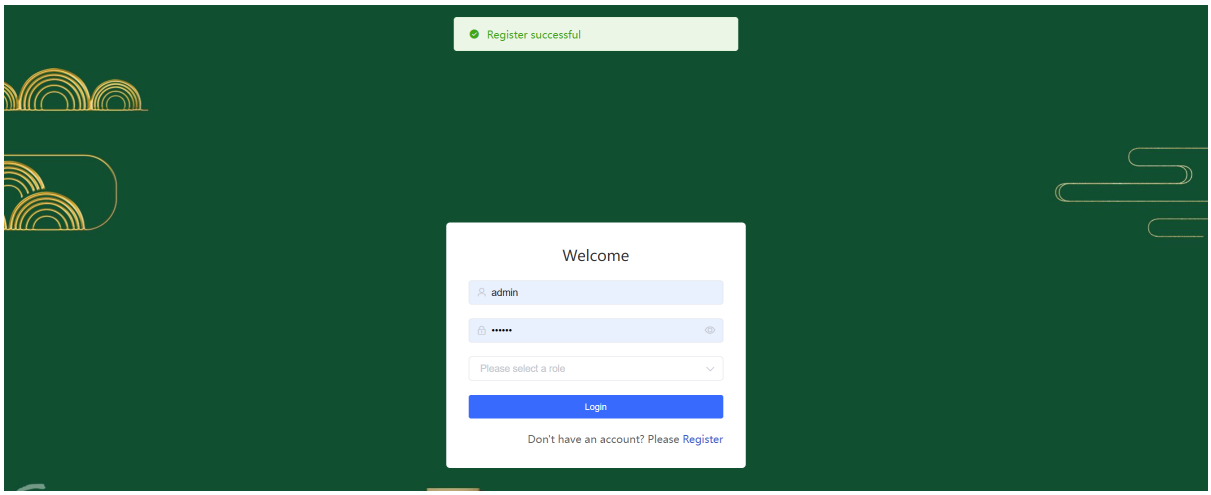

Figure A1.3: Sign-up successful

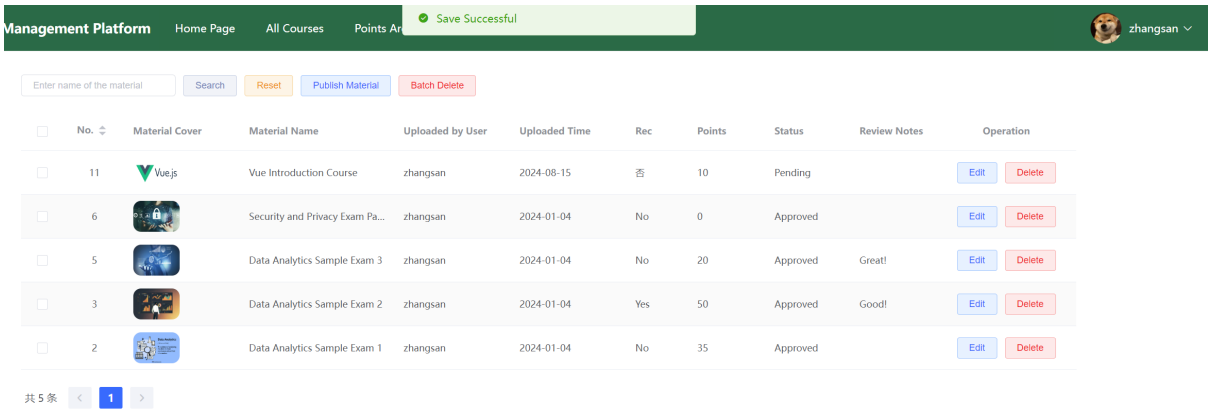

Figure A1.4: publish material

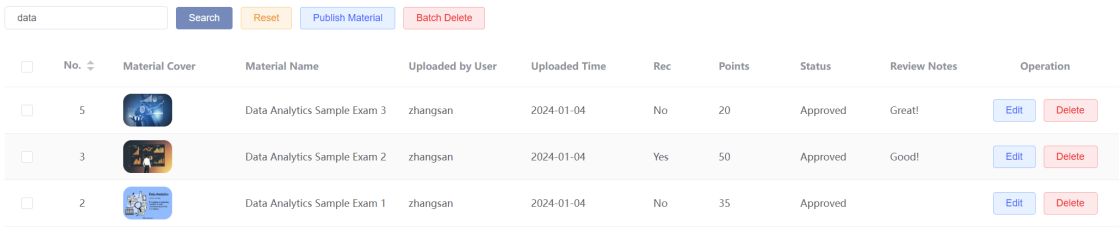

Figure A1.5: Search for Material Name

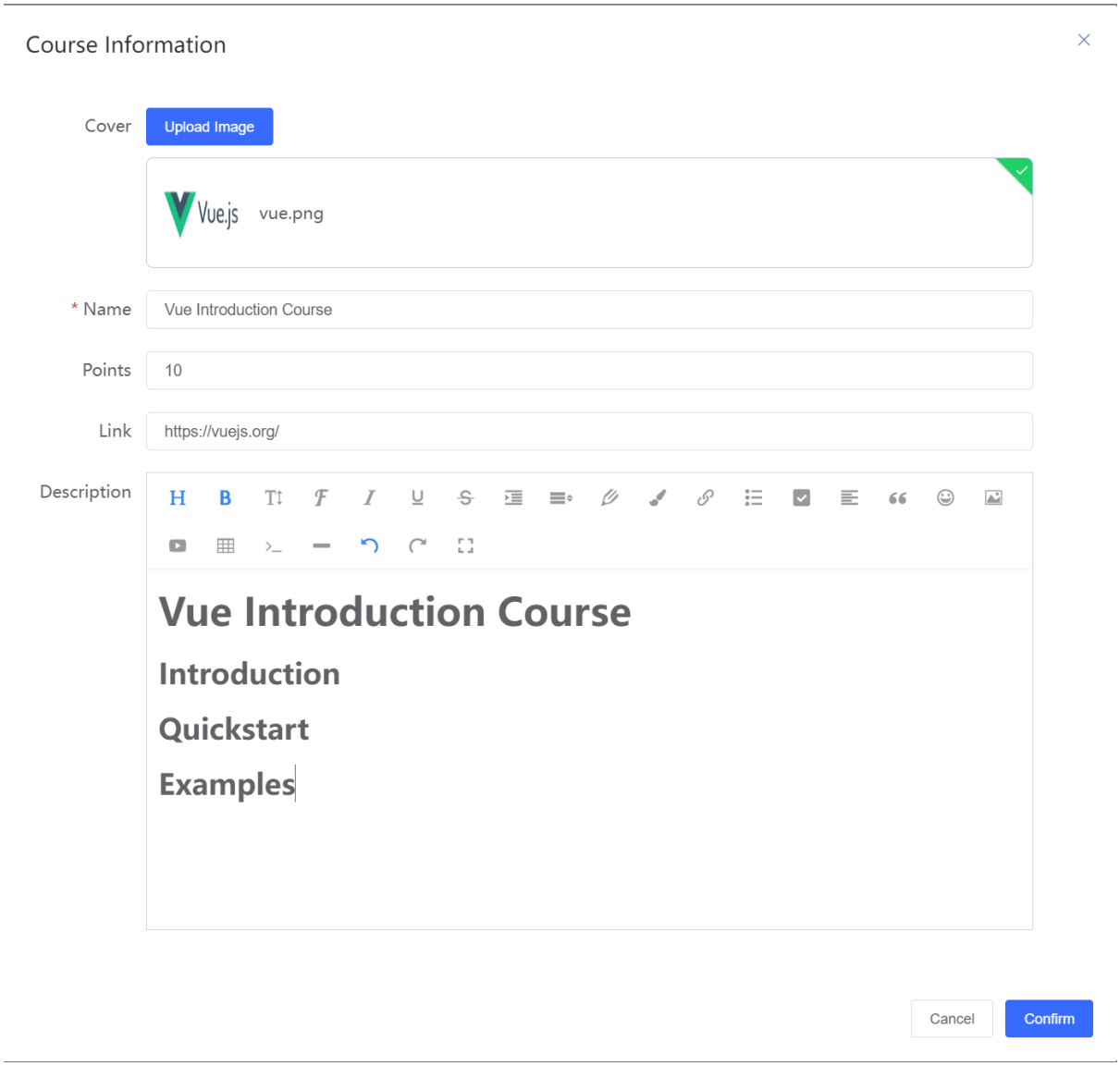

Figure A1.6: Edit

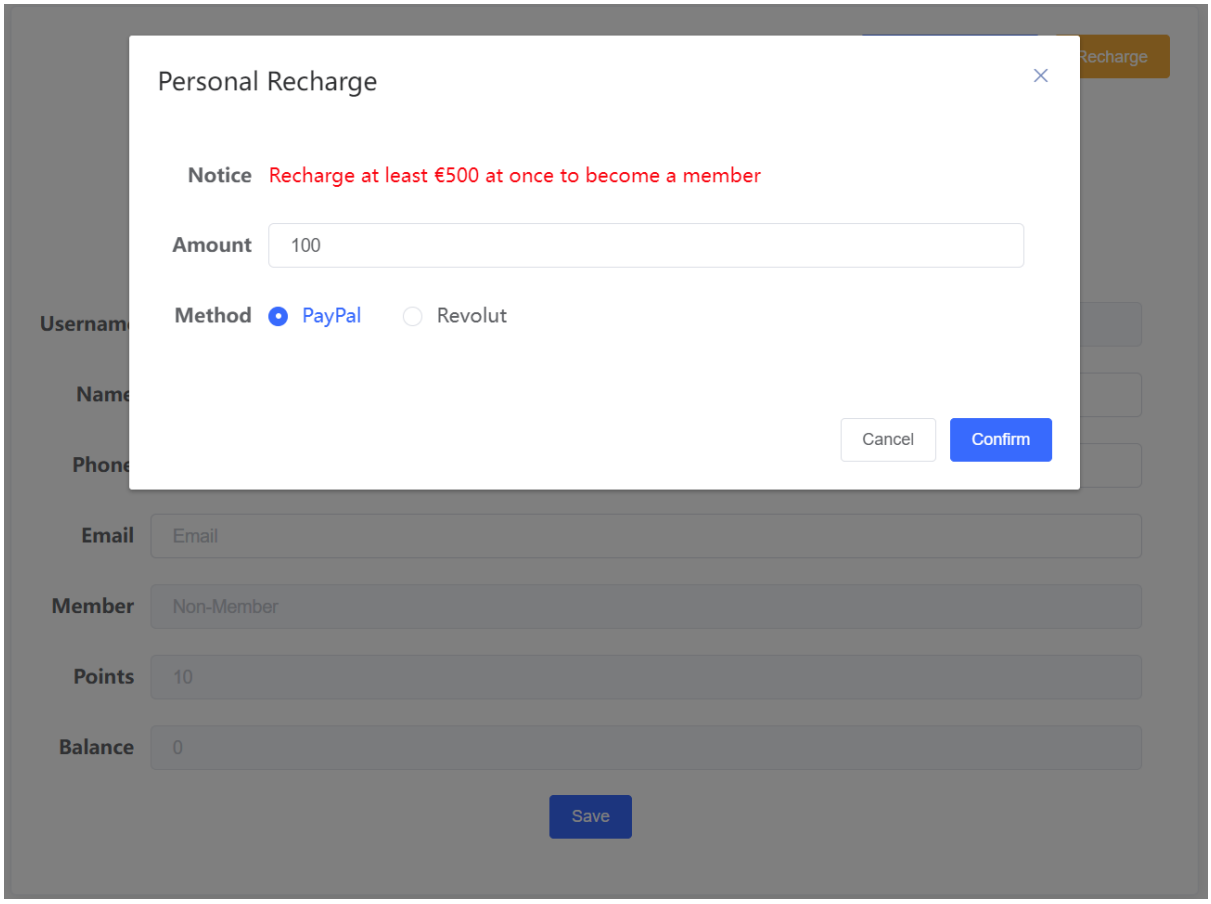

Figure A1.7: Number of recharges less than 500 euros per transaction-before recharge

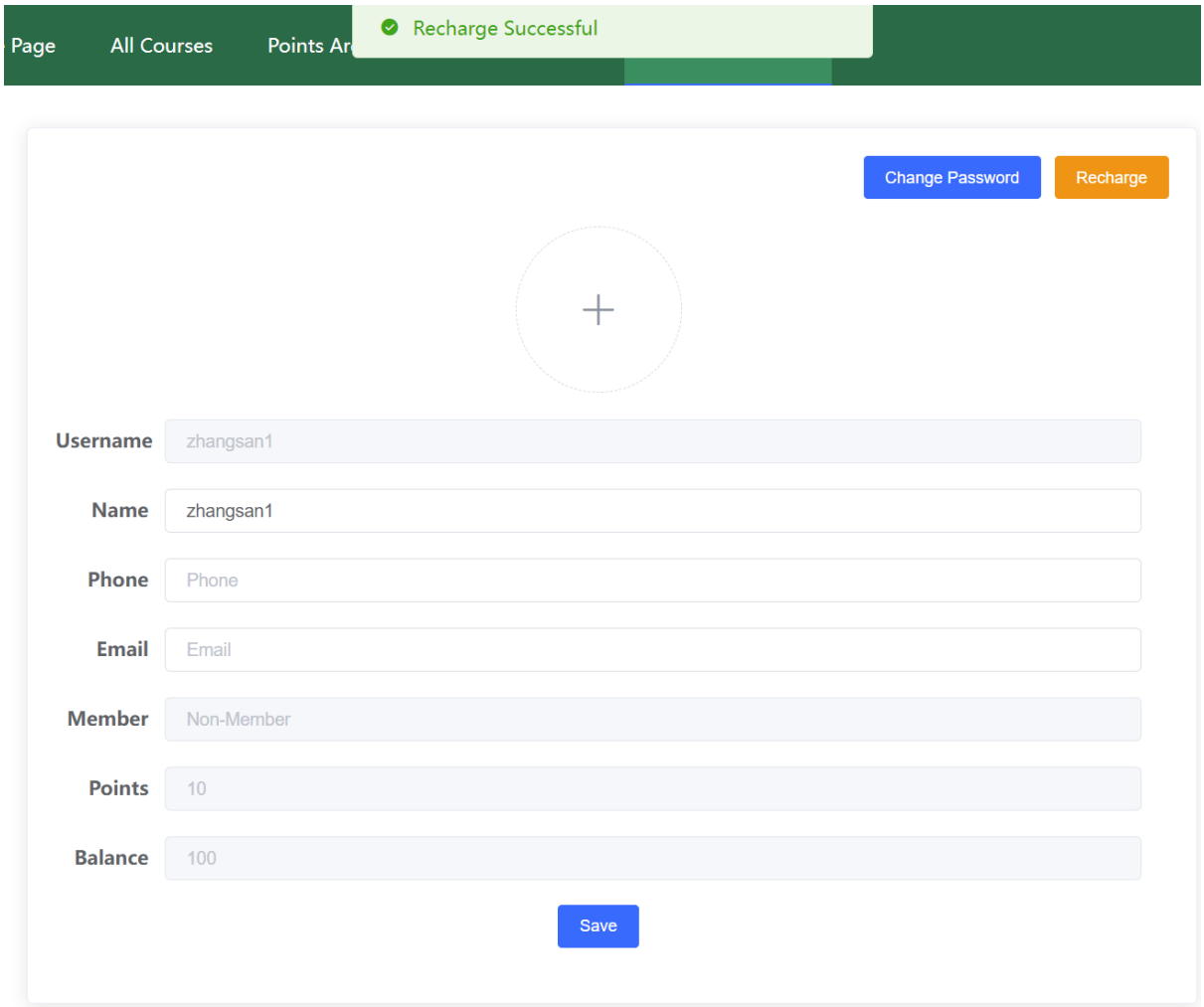

Figure A1.8: Number of recharges less than 500 euros per transaction-after recharge

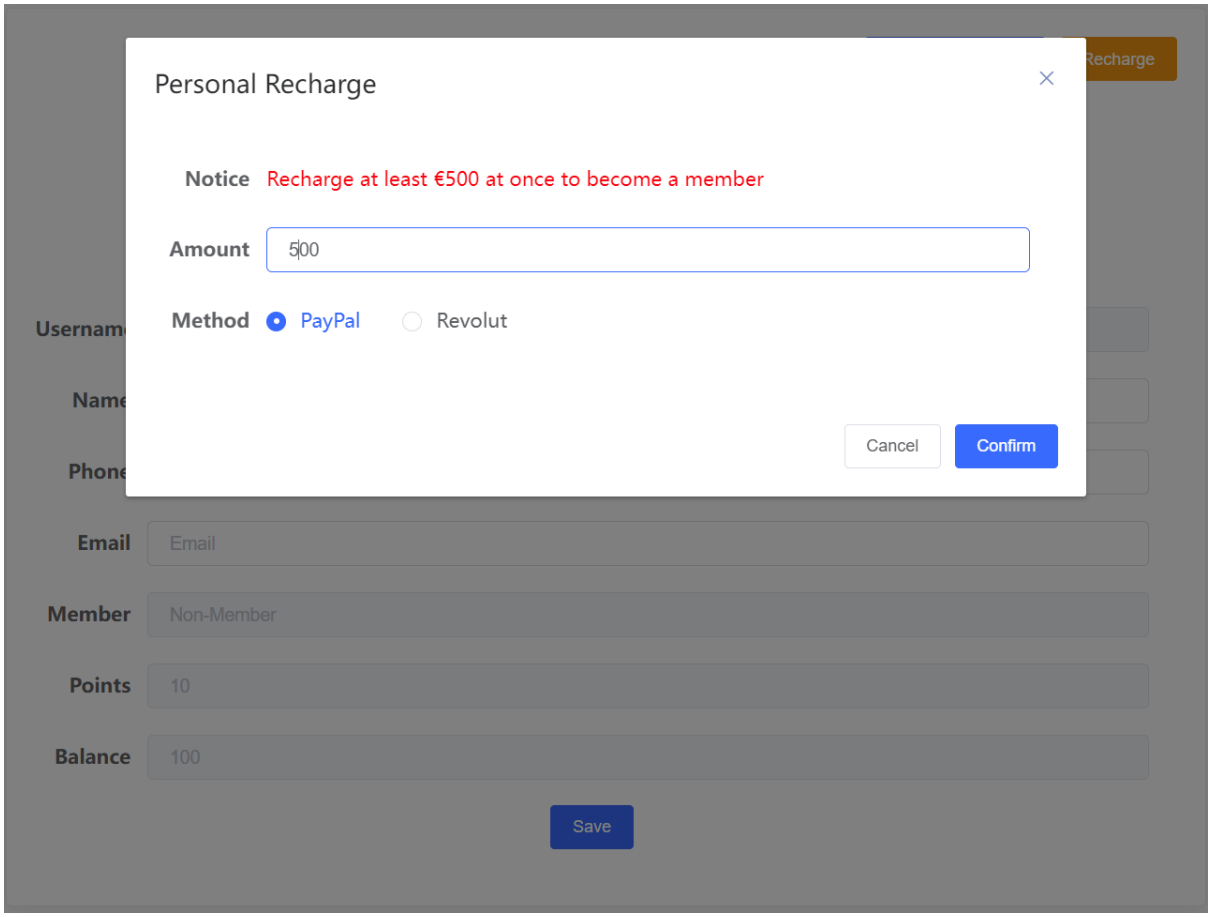

Figure A1.9: Number of recharges greater than or equal to 500 euros per transaction-before recharge

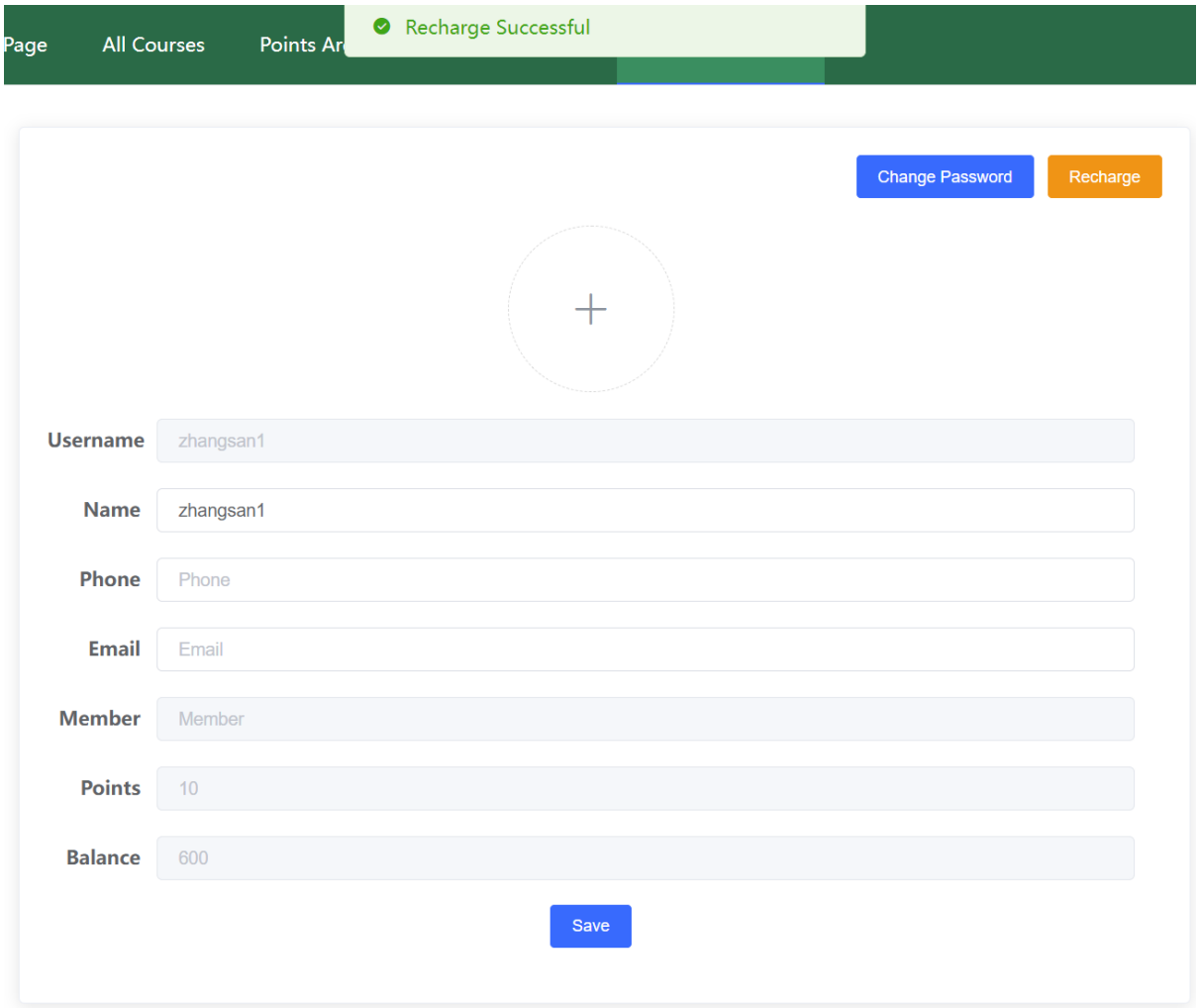

Figure A1.10: Number of recharges greater than or equal to 500 euros per transaction-after recharge

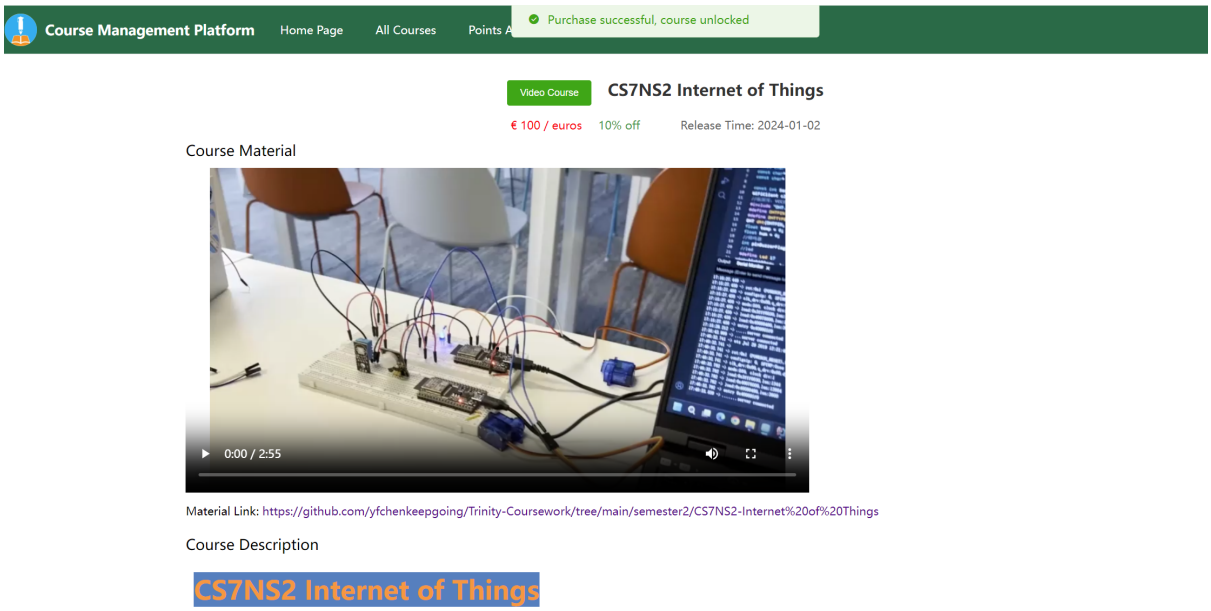

Figure A1.11: Account balance is sufficient, course purchase successful

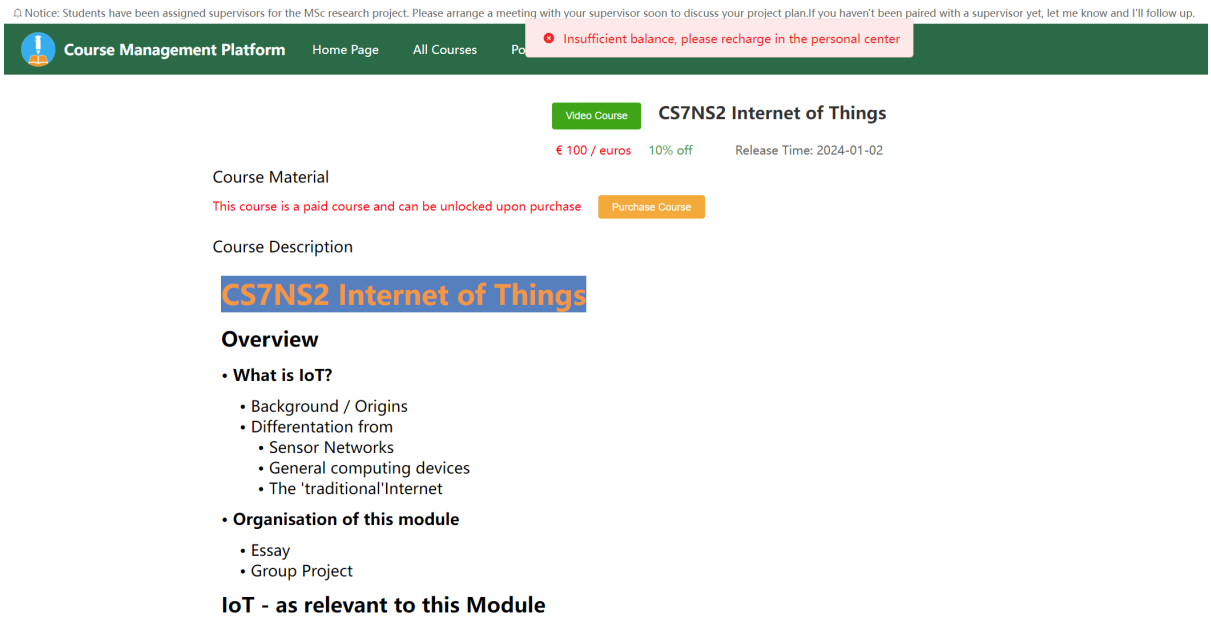

Figure A1.12: Account balance is insufficient, course purchase failed

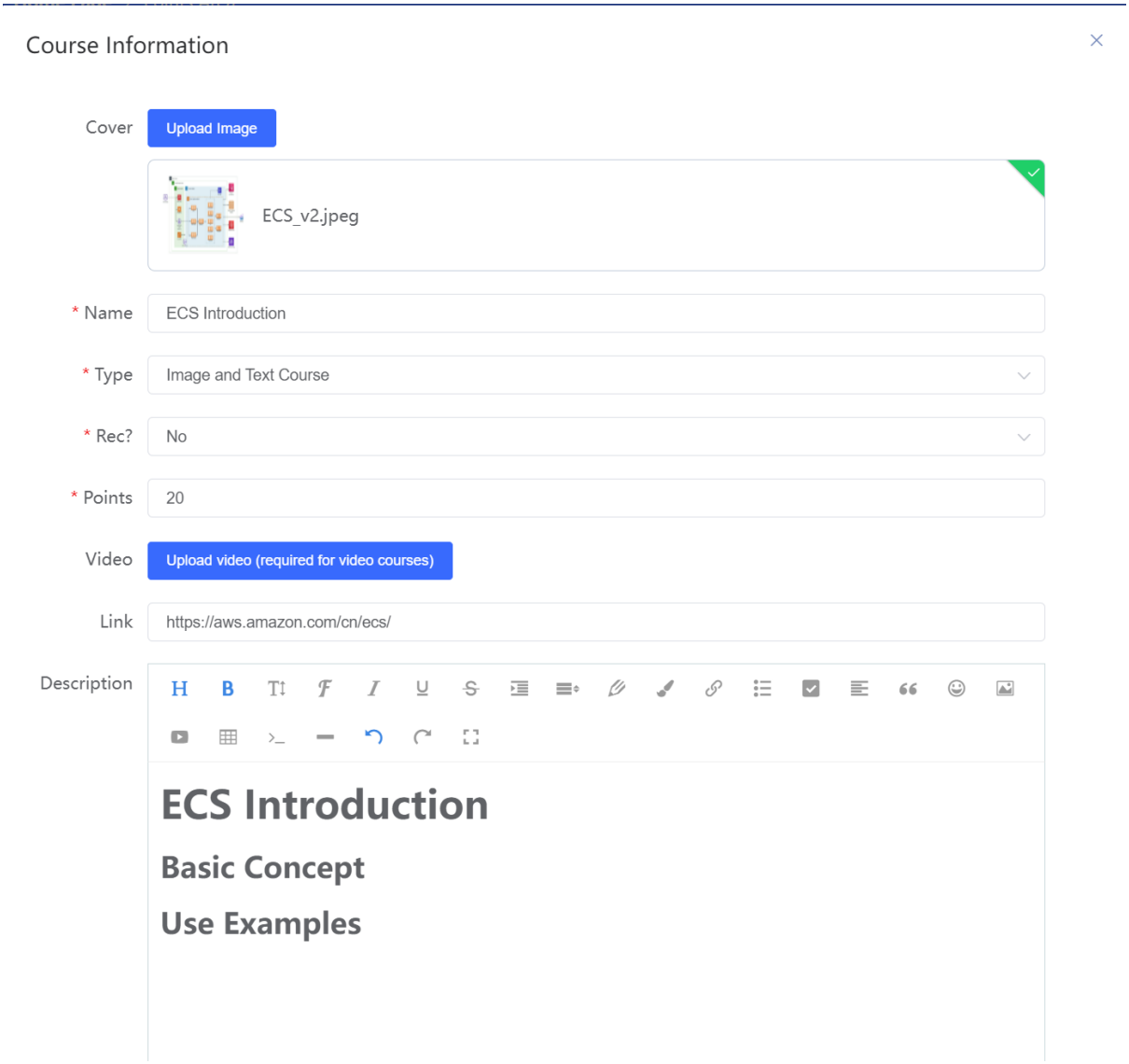

Figure A1.13: Add new courses function

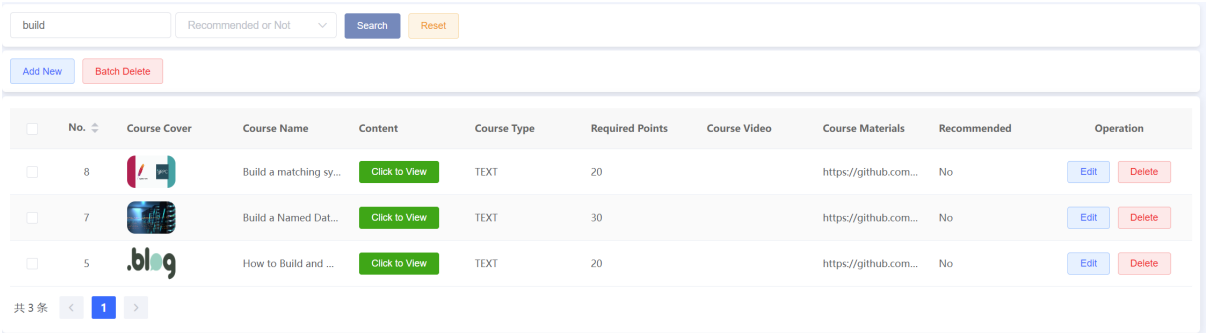

Figure A1.14: Search for material name function

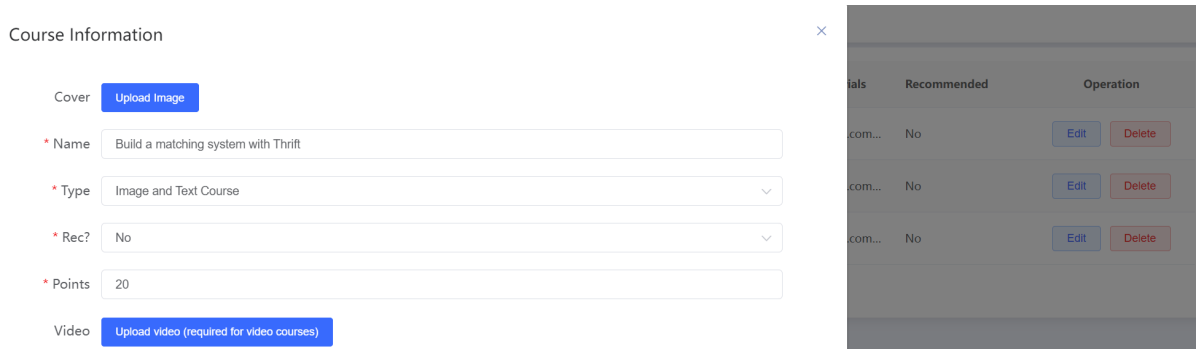

Figure A1.15: Edit function

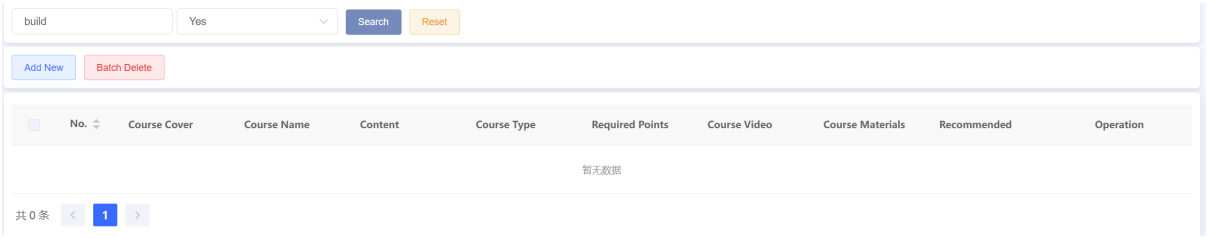

Figure A1.16: Filter courses based on whether they are recommended-recommended

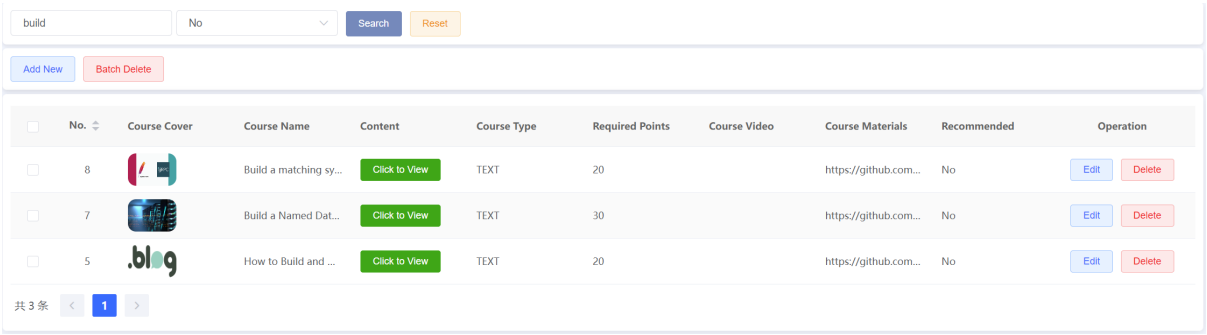

Figure A1.17: Filter courses based on whether they are recommended-not recommended

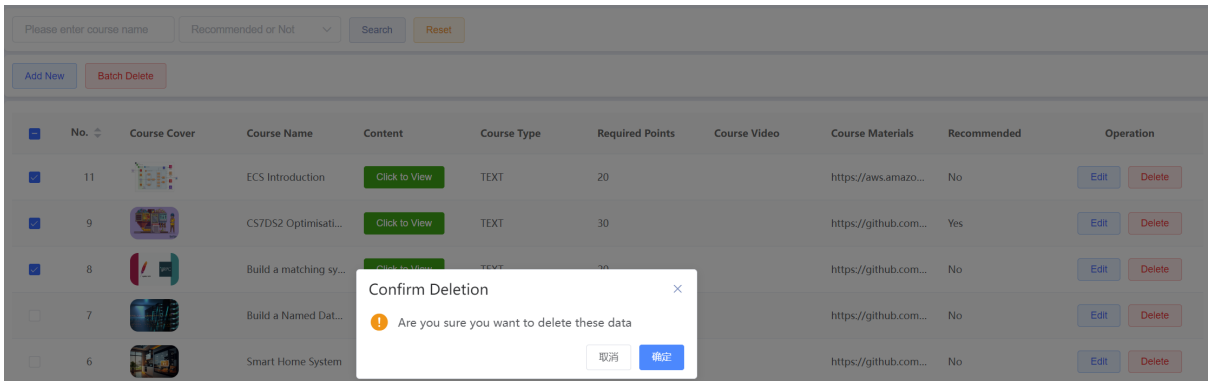

Figure A1.18: Batch delete function

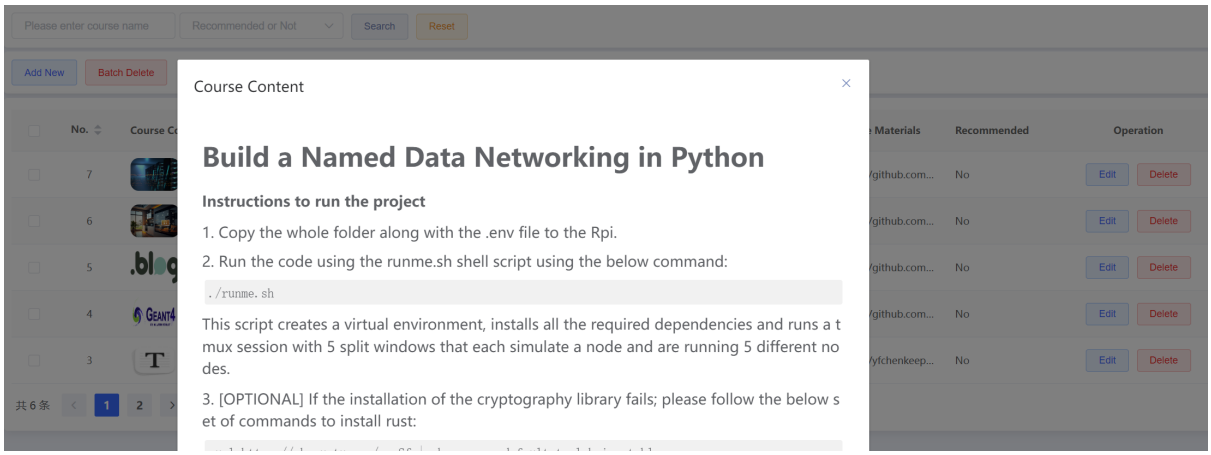

Figure A1.19: Click to view function

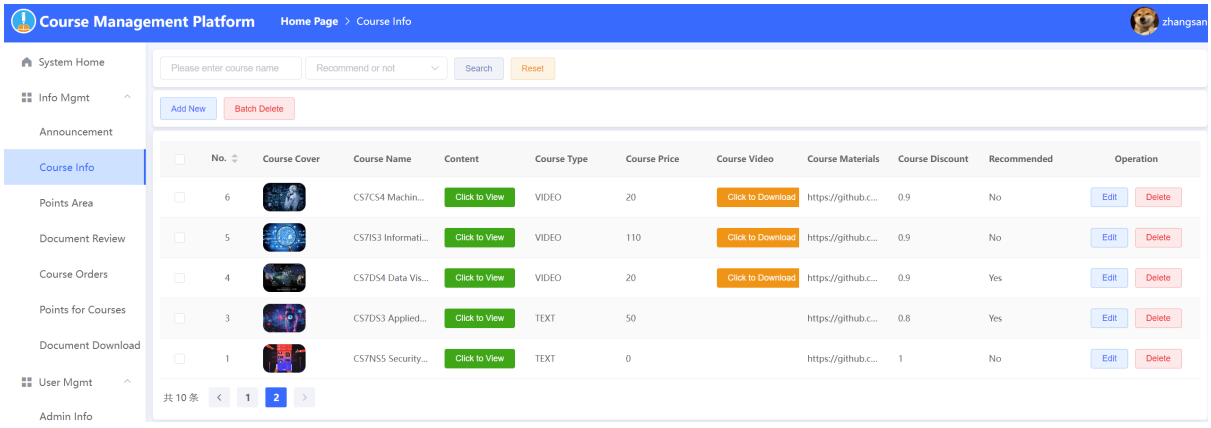

Figure A1.20: Administrator interface: Only "CS7DS4 Data Visualization" is recommended

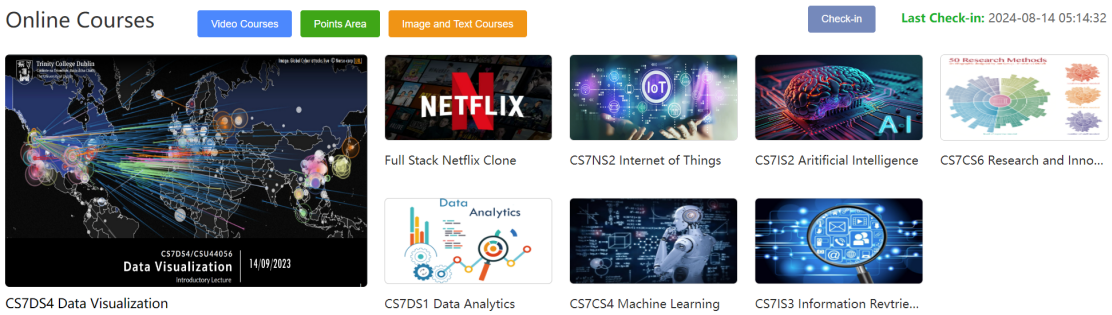

Figure A1.21: User interface: Only "CS7DS4 Data Visualization" is recommended

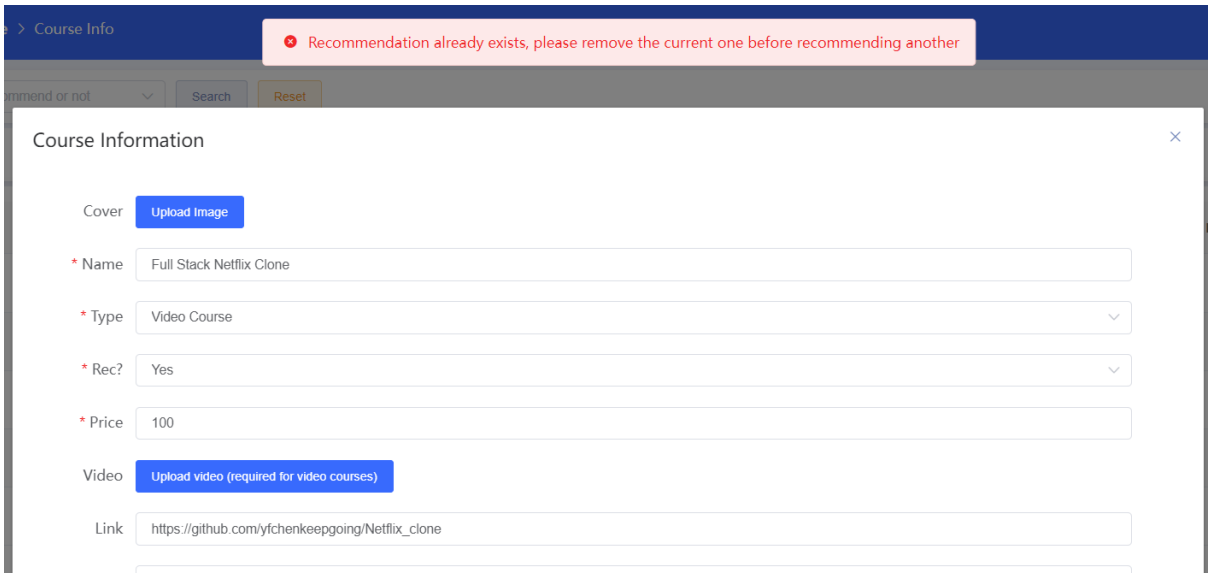

Figure A1.22: Administrator interface: "CS7DS4 Data Visualization" and "Full Stack Netflix Clone" are recommended simultaneously

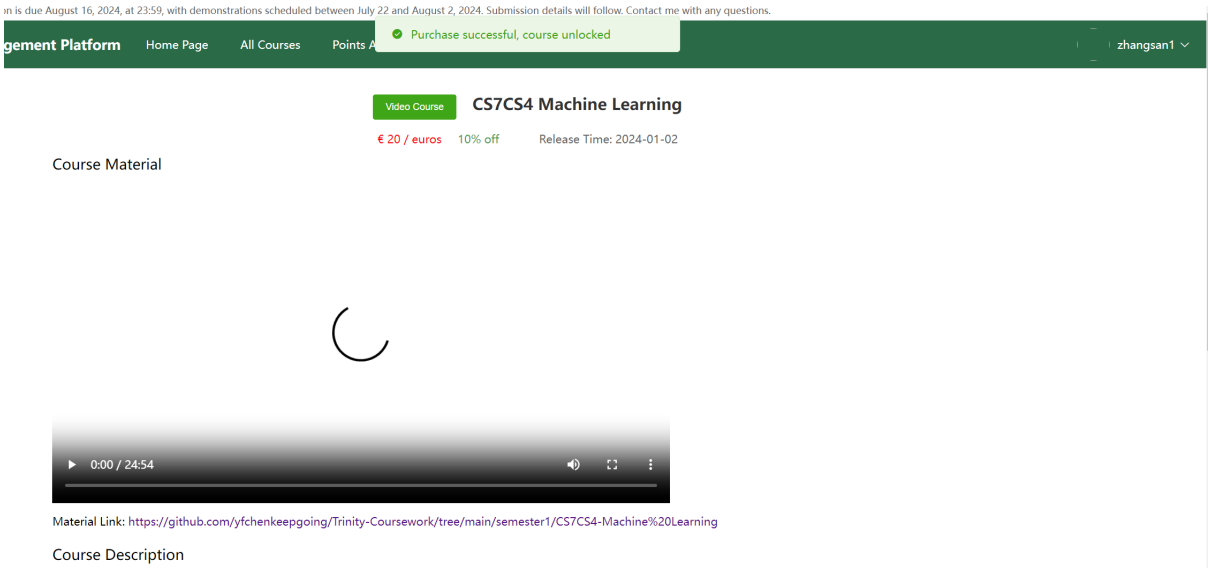

Figure A1.23: User zhangsan1 purchases "CS7CS4 Machine Learning."

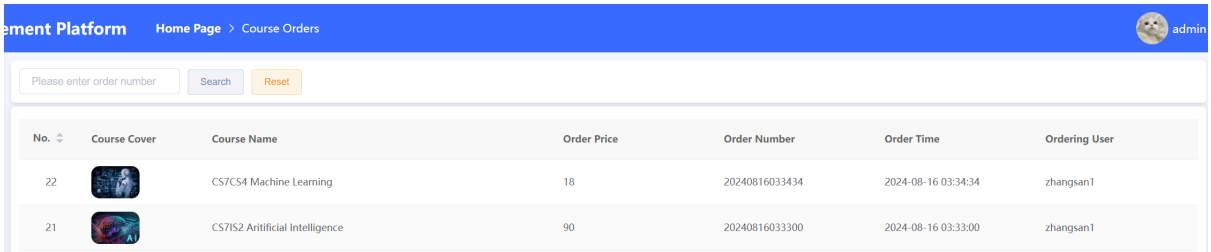

Figure A1.24: Order information is updated on the administrator interface

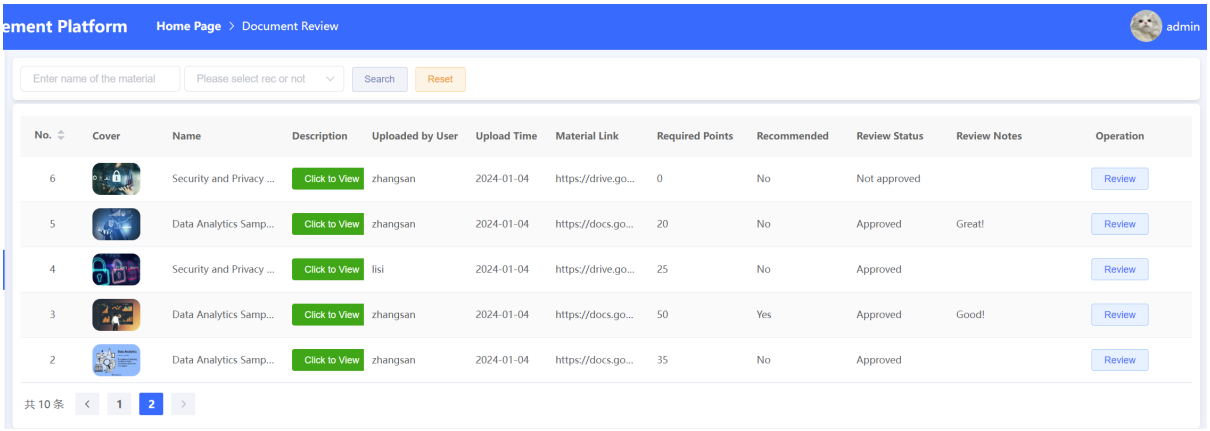

Figure A1.25: Administrator document review interface

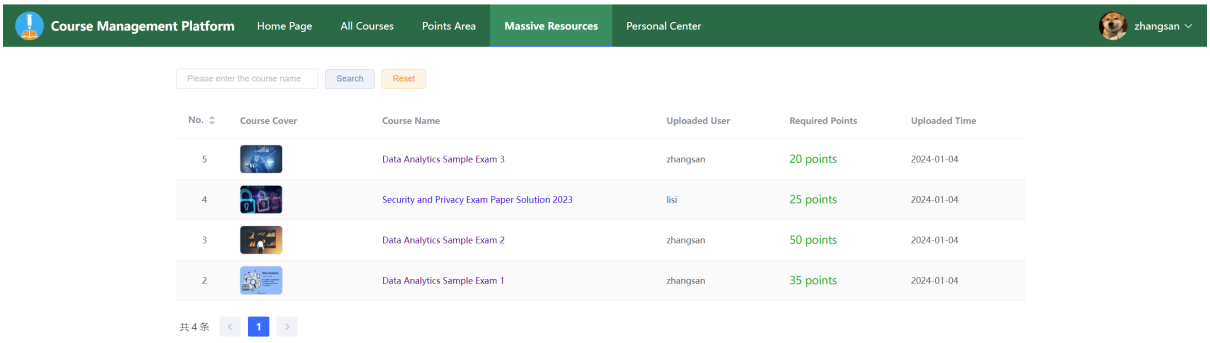

Figure A1.26: User massive resources interface