Software Design Document

for

<SMOOTEA Learning Access and Class Enrollment System>

Version 1.0

Prepared by
< NUR SYAFIKA ALYA BINTI MOHD ZAMRI AZHAR, 2023823848>

<5 AUGUST 2025>

Table Of Contents

1.0 Introduction

1.1 Purpose

As Smootea Academy grows rapidly and more and more people require the efficient learning management online, the architecture design of Smootea Learning Access & Class Enrollment System (SLACES) is the subject of this Software Design Description (SDD) document. SLACES was created to overcome the shortcomings of the previous manual enrollment and content delivery system of the Smootea Academy that was largely based on the use of static websites and third-party messaging applications like Facebook and WhatsApp. These approaches tended to cause delays, manual verification and absence of tracking of progress, which led to operational inefficiencies on the part of the administrators and students.

With this knowledge of the challenges, SLACES will offer a centralized, dynamic, and secure location to manage the classes, enroll and access the materials, and simplify the operations of the academy and the learners. This SDD specifies the architectural design of the system, design rationale, software packages and frameworks needed and the interplays of different system components. The paper will be used to establish clear and broad-based demands of developing, maintaining, and future development of SLACES, so as to provide a strong foundation of scalable and professional learning management at Smootea Academy. Finally, this SDD will be used to launch a technically competent and effective solution, which will facilitate the educational goals and further development of Smootea Academy.

1.2 Scope

SMOOTEA Learning Access & Class Enrollment System is a web-based system designed to be used by SMOOTEA Academy in order to make the process of class enrollment and access to learning materials as simple as possible. The system has two main users, namely, the students who have bought online classes in the academy and the administrators who control the information about the classes and the access of the students.

The system will be an internal system by SMOOTEA Academy via a web browser where students will be able to register and access their respective classes. After registration, the students will be asked to input a unique invoice number that they received during the purchase. The system will check the code with the records that are in the database. When the information is a match, the student will

be allowed to access the corresponding class information without any manual authorization by the administrator.

Under this system, the management of all classes are centralised and this leads to less administrative burden and delivery of content to the students at a faster rate. The system will cover user registration and authentication, invoice code validation, role-based access control (student and admin), administration of class and invoice records on the admin side, progress tracking and student dashboards to access the enrolled classes and digital materials. Nonetheless, integration with payment gateways, external social media APIs, automated certificate generation, and live online classes are not part of this project.

This project was developed using the Waterfall methodology, following a structured and sequential process from requirements gathering and analysis, through system design and implementation and deployment.

1.3 References

Book Glinz, M., Loenhoud, H. van, Staal, S., & Bühne, S. (n.d.). Handbook for the Cpre Foundation Level according to the Ireb Standard (First Release, Vol. 1.0.0).

John W. Satzinger, Robert B. Jackson, Stephen D. Burd. System Analysis and Design in a Changing World (7th edition) .

Website Bandakkanavar, R. (2023, May 8). Software Requirements Specification document with example.

Krazytech. *https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database*

1.4 Document Structure

The Smootea Learning Access and Class Enrollment System (SLACES) is a standalone and independent software solution that is created specifically to the Smootea Academy. In contrast to a mere add-on or upgrade to the existing systems, SLACES brings an entirely new approach to the management of class enrollment and access to the educational materials. In the past, Smootea Academy used manual methods and third-party applications like static websites, Facebook, and WhatsApp to register classes, confirm payment, and deliver content. The new system automates and streamlines these processes, which demonstrates the desire of Smootea Academy to be efficient in operation and modern approaches to education.

SLACES is composed of two primary user interfaces, namely the admin dashboard and the student dashboard. These elements are tightly interconnected, forming a whole system that makes it possible to manage all classes, enroll students, and access learning materials without any difficulties. The design facilitates easy exchange of information between administrators and students to enhance effective communication and cooperation during the learning process. Through its centralization, SLACES removes the manual effort that used to be spent on the process of verification and access control and serves as a considerable step toward efficient, scalable, and user-centric educational management. The standalone character of the system highlights the fact that it is a fundamental tool in the enhancement of Smootea Academy objectives of better operations and student satisfaction.

1.5 Definitions, Acronyms and Abbreviations

Convention	**Description**
Font Size	The header will include a font size of 16. The sub header is going to have a font size of 14. Font size 12 will be used for writing written material.
Font Type	The font used for the header, sub header, and written content is Times New Roman.
Bold	It will be used to highlight headers and sub headers, as well as focusing on some contents titles.
SRS	Stands for System Requirement Specification

DB	Stands for database
SSD	Stands for system sequence diagram
CRUD	Create, Update and Delete

1.6 System Overview

Functionality

The Smootea Learning Access & Class Enrollment System (SLACES) is a system that is intended to simplify and automate the process of online class enrollment and access in Smootea Academy which is an educational branch of the Smootea brand. SLACES eliminates the manual, disjointed procedures that were conducted using fixed websites and messaging applications such as Facebook and WhatsApp. Key functionalities of SLACES are:

1. Class Enrollment and Management
   - The students will be able to register and enroll in the classes through a security unique invoice code.
   - Administrators can control the offering of classes, enrollments and regulate access to learning materials.

2. Delivery of Learning Material
   - Admins are able to upload, update and structure educational material (videos, PDFs, links) per each class.
   - Course content can be dynamically viewed and interacted with by students and progress tracked.

3. Dashboard analytics and progress tracking
   - Students and admins are able to track the participation and completion of classes and access analytics on course activity and engagement.

4. Communication and Support to the Users
   - Incorporated capabilities that allow students to reach out to admins through WhatsApp in case of support and resolution of issues.

Context

SLACES came about due to the high rate of expansion of Smootea Academy and the inefficiency of its previous manual systems. In the past, enrolment and access to content was dependent on many third-party platforms which caused delays, errors and administrative bottlenecks. The new system will:

- Make the operations more efficient to both students and administrators.
- Decrease the time and effort required to enroll and access materials.
- Enhance accuracy and reliability in student progress and access permission management.
- Improve communication and coordination in the academy.

Design

SLACES has two primary user interfaces that are specific to the functions in the academy:

1. Admin Interface

Allows controlling classes, learning resources, enrollments, and student accounts.

Offers participation tracking tools, invoice management and administrative tools.

2. Student Interface

- Enables students to enroll, sign in, view materials and track their progress.

It has support features like direct contact with admins to help.

Classes of users and their features

Admins:

- Have high privileges, which are in charge of all system resources, the content of classes and access to users.

Students:

- Have restricted rights, which are centered on registering to classes and getting the designated learning resources.

Operating Environment:

- It is a web-based system; it needs a reliable internet connection.
- It is created to work with modern web browsers (e.g. Chrome, Edge) on Windows 10 or later.

- It is designated with hardware and software requirements to perform smoothly.

Background Information

As Smootea Academy continued to grow, its legacy manual operations were no longer adequate to support its operations. These challenges are overcome by the transfer to an automated centralized platform with SLACES, which offers a unified and unified system of enrolling a class, delivering materials, and monitoring progress. This enhancement helps in more effective operation, effective resource management, and an expandable base of further developments.

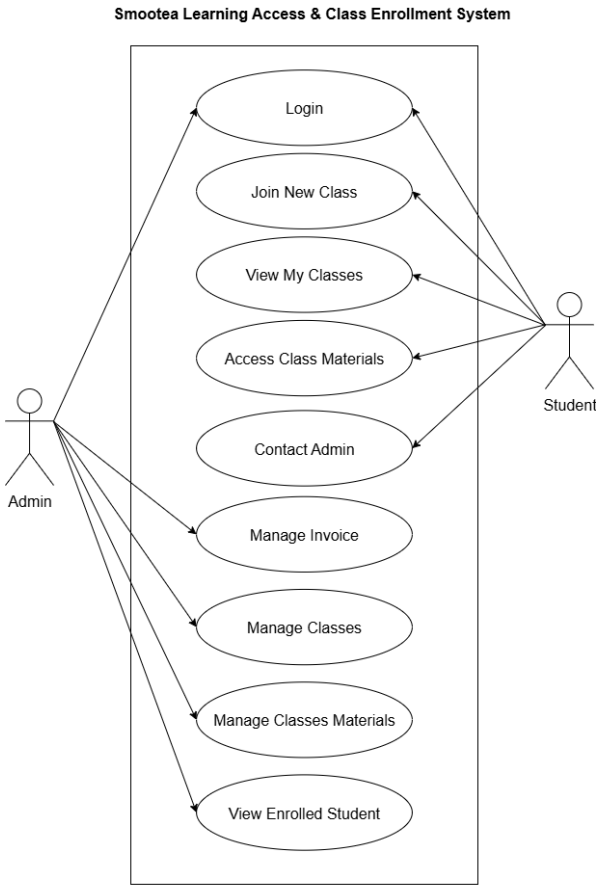

Figure 1.0 Use Case Diagram

1. UC-B01 Login
• User logs into the system.

2. UC-S01 Join New Class
• Student submits invoice number to enroll in class.

3. UC-S02 View My Classes
• Student views list of joined classes and its progress.

4. UC-S03 Access Class Materials
• Student searches and views class content.

5. UC-S04 Contact Admin
• Student clicks WhatsApp button to ask for help.

6. UC-A01 Manage Classes
• Admin creates and updates class info including title, description and thumbnail.

7. UC-A02 Manage Class Materials
• Admin creates, adds, edits, and deletes class materials.

8. UC-A03 Manage Invoices
• Admin adds, updates, or deletes invoice records.

9. UC-A04 View Enrolled Students
• Admin views student list, joined class and their progress

2.0 System Architecture

2.1 Architectural Description

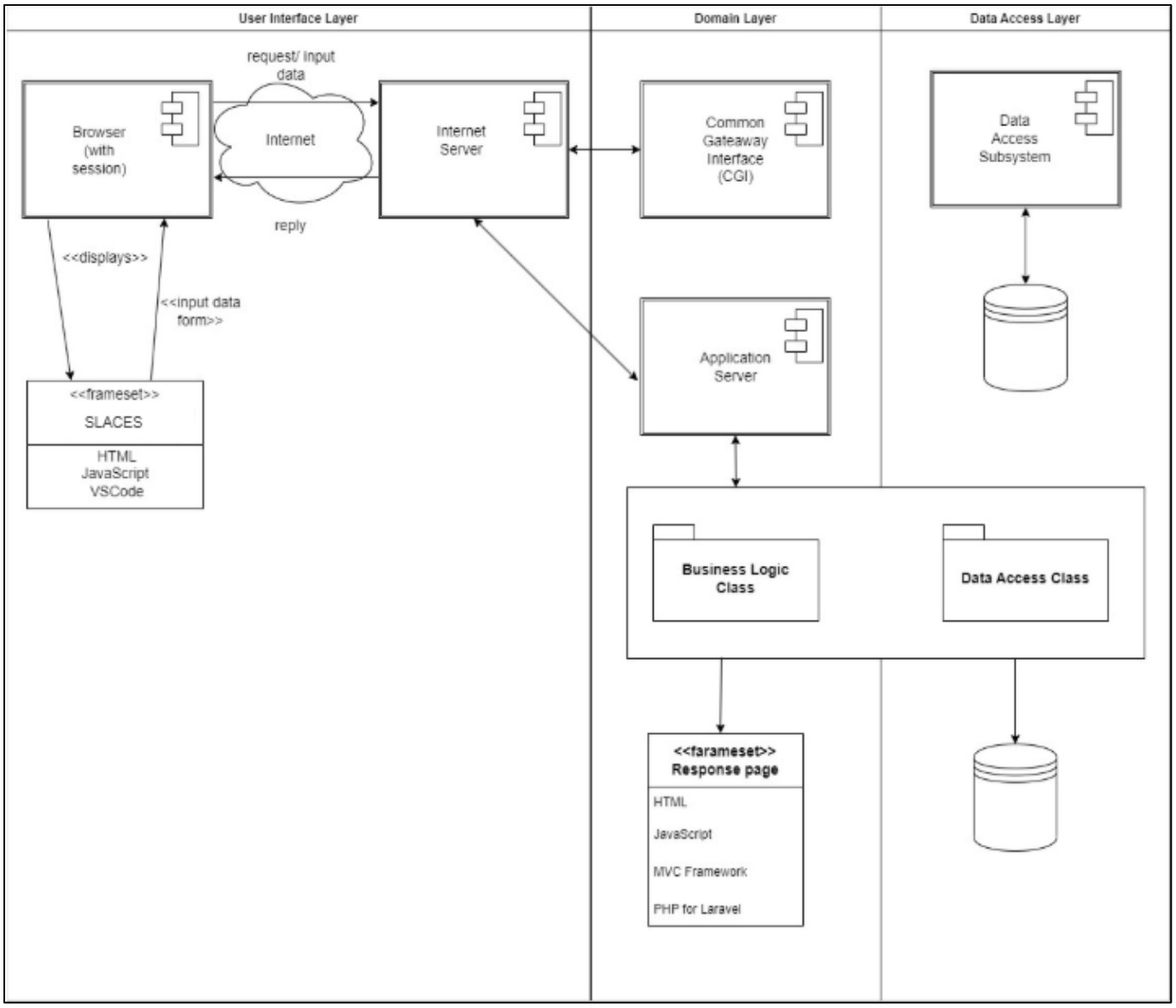

Figure 2.0

The Smootea Learning Access & Class Enrollment System (SLACES) has been designed based on a three-layer architecture model, which improves modularity, maintainability and scalability. The User Interface Layer is the first layer and includes the browser interface and the web components constructed using HTML, JavaScript, and Laravel Blade templates that are usually created using Visual Studio Code. This layer is in charge of all the user interactions, where it receives an input of data like login credentials, invoice codes, or material requests, and outputs the dashboards, class lists, and progress tracking. The Internet Server is the point of entry into the application and receives user requests through the Internet.

Domain Layer is the heart of SLACES and it comprises business logic and business processes. It contains such elements as the Common Gateway Interface (CGI) and the Application Server that process the requests of the User Interface Layer. The Business Logic Class is in charge of business rules, e.g. validating enrollments, class creation, student progress tracking, and access rights control, whereas the Data Access Class is in charge of mediation with the underlying database. This decoupling makes the business processes more secure and efficient without risking sensitive logic to the interface.

Lastly, there is the Data Access Layer which is focused on data management. It is comprised of the Data Access Subsystem and the database (MySQL) which safely stores all the data in the system such as user accounts, classes, invoices, enrollments and learning materials. This layer isolates the real database activities in the business logic so that the Domain Layer can conduct data transactions without having direct contact with the database structure. Caching can be used to enhance performance of the data which is frequently accessed.

All in all, this three-layer architecture makes SLACES robust, secure, and flexible. The system is easier to update and expand with each layer having clear responsibilities, and the system maintains data integrity and helps Smootea Academy to operate efficiently in terms of education.

2.2 Design Rationale

In case of the Smootea Learning Access & Class Enrollment System (SLACES), three-layered architecture model was selected to achieve maximum modularity, maintainability, scalability and to minimize the system complexity. This separation of the system into layers, namely User Interface, Domain and Data Access, allows each layer to be developed, maintained and updated separately, which reduces the possibility of unexpected effects in other components of the system. The modular approach is particularly useful in the case of SLACES, which deals with highly important processes like user registration, class enrollment, role-based access, and progress monitoring of a dynamic online learning environment.

Debugging, testing, and further development is easier because of the separation of concerns which is part of this architecture. As an example, the user interface (which is implemented using HTML, JavaScript, Blade and CSS) can be updated without affecting the business logic or the database schema. Likewise, data storage modifications or database optimization (MySQL and caching techniques) do not need frontend or business rule changes. This versatility helps in the long-term development of SLACES, where new features and improvements can be added with limited interference.

A trade-off of the 3-layer model is that communication and coordination among layers is more complex. Although this has the potential to add some overhead and possible latency (particularly in data-intensive workloads), the maintainability and scalability advantages are worth the tradeoff in SLACES, since it is a centralized solution to educational management. Careful caching and efficient access patterns further optimize performance, and take advantage of the strengths of the architecture.

Other architectures, including deployment diagrams or network diagrams, were also discussed but not chosen. Deployment diagrams are concerned with hardware and physical deployment which are not sufficient in providing the logical structure and data flow needed in SLACES. Network diagrams focus on physical connectivity and infrastructure, and do not have the detail required to explain application logic and data processing. The three-tier architecture is most suitable in the business logic, data management and user interaction, which are in line with the objectives of SLACES to provide a robust, scalable and user-friendly learning management platform.

2.3 Decomposition Description

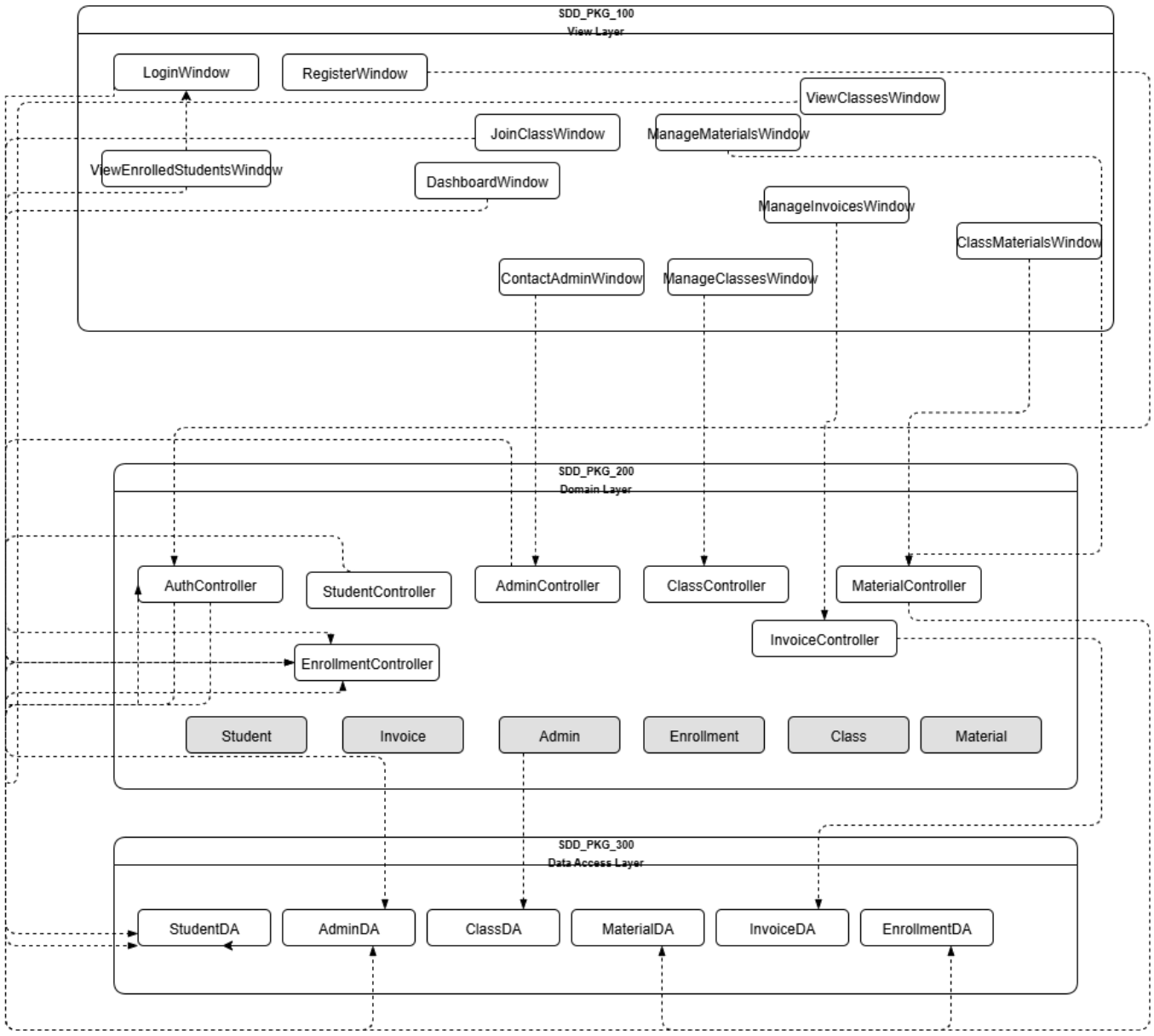

Figure 2.1 Package Diagram

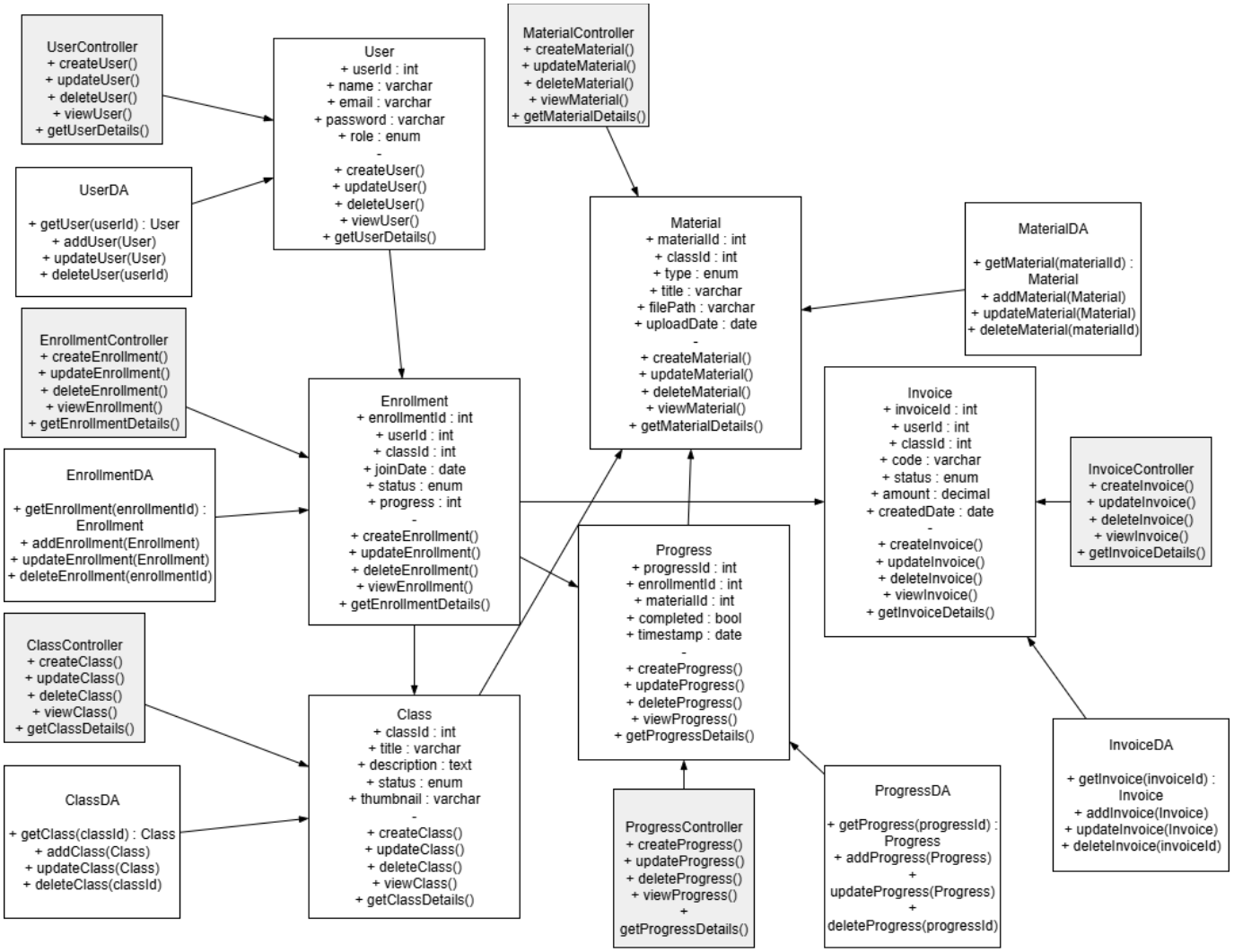

Figure 2.2 Detailed Design Class Diagram

3.0 Data Design

3.1 Database Description

The Smootea Learning Access & Class Enrollment System (SLACES) utilizes MySQL version 8.0 as its database management system to securely store and manage all core data. The database schema is designed to support the platform's key educational and management features, with well-defined tables for each major entity and their relationships.

The users table stores information about every user, including user IDs, names, emails, passwords, roles (admin, student), profile images, and timestamps for account creation and updates. This table supports authentication, authorization and personalization throughout the system.

The classes table manages all available learning modules, recording unique class IDs, titles, descriptions, status (active, inactive, upcoming), thumbnails, pricing details, start and end dates and timestamps. This enables dynamic class listings and management.

The enrollments table tracks each user's participation in classes, including enrollment IDs, user IDs, class IDs, status (active, pending), enrollment dates, completion percentages, and audit timestamps. This allows for monitoring progress and managing course access.

The materials table contains all learning resources associated with each class, such as videos, PDFs, links, and documents. It records material IDs, titles, descriptions, associated class IDs, resource type, file paths/URLs, thumbnail images, activity status, sorting order, and timestamps for creation and updates.

The invoices table manages payment and billing records for class enrollments. It stores invoice IDs, invoice numbers, user IDs, class IDs, payment amounts, status (used, unused), invoice dates, associated emails, and audit timestamps.

The progress table records granular user progress within specific class modules, including progress IDs, user IDs, class IDs, module IDs, completion status, completion timestamps, and audit timestamps. This supports detailed tracking of user learning and achievements.

The logs table maintains an audit trail of admin actions such as creating, editing, or deleting users, classes, invoices, and materials, with fields for log ID, admin ID, action type, description, and timestamps.

Each table in the SLACES database is defined with precise data types and lengths to ensure integrity, scalability, and performance. The schema supports secure, reliable, and efficient operation of all educational management and access functionalities in the SLACES platform.

3.3 Data Dictionary

Table: users

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	ID of the user	bigint	auto_increment	No	-	Yes	Yes
name	Name of the user	varchar(255)	-	No	-	Yes	No
email	Email of the user	varchar(255)	-	No	-	Yes	Yes
role	Role of the user	varchar(255)	-	No	guest	Yes	No
email_verified_at	Email verified date	timestamp	-	Yes	null	No	No
password	Password of the user	varchar(255)	-	No	-	Yes	No
avatar_url	Avatar image url	varchar(255)	-	Yes	null	No	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: classes

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	Class ID	bigint	auto_increment	No	-	Yes	Yes
title	Class title	varchar(255)	-	Yes	null	No	No

name	Class name	varchar(255)	default ''	No	''	Yes	No
description	Class description	text	-	No	-	Yes	No
thumbnail	Class thumbnail path	varchar(255)	-	Yes	null	No	No
price	Price	decimal(10,2)	default '0.00'	No	0.00	Yes	No
status	Status	enum	default 'active'	No	active	Yes	No
start_date	Start date	date	-	Yes	null	No	No
end_date	End date	date	-	Yes	null	No	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: enrollments

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	Enrollment ID	bigint	auto_increment	No	-	Yes	Yes
user_id	User ID (FK)	bigint	-	No	-	Yes	No
class_id	Class ID (FK)	bigint	-	No	-	Yes	No
status	Enrollment status	enum	default 'pending'	No	pending	Yes	No
enrollment_date	Enrollment date	date	-	No	-	Yes	No
completion_percentage	Completion %	int	default '0'	No	0	Yes	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: invoices

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	Invoice ID	bigint	auto_increme	No	-	Yes	Yes

			nt				
invoice_number	Invoice number	varchar(255)	-	No	-	Yes	Yes
user_id	User ID (FK)	bigint	-	Yes	null	No	No
class_id	Class ID (FK)	bigint	-	No	-	Yes	No
amount	Invoice amount	decimal(10, 2)	default '0.00'	No	0.00	Yes	No
status	Status	enum	default 'unused'	No	unused	Yes	No
invoice_date	Invoice date	date	-	Yes	null	No	No
email	Email	varchar(255)	-	Yes	null	No	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: materials

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	Material ID	bigint	auto_increment	No	-	Yes	Yes
title	Resource title	varchar(255)	-	No	-	Yes	No
description	Resource description	text	-	Yes	null	No	No
class_id	Class ID (FK)	bigint	-	No	-	Yes	No
type	Resource type	enum	-	No	-	Yes	No
file_path	Resource file path	varchar(255)	-	Yes	null	No	No
url	Resource URL	varchar(255)	-	Yes	null	No	No
thumbnail	Resource thumbnail path	varchar(255)	-	Yes	null	No	No
is_active	Is resource active?	tinyint(1)	default '1'	No	1	Yes	No
sort_order	Sort order	int	default '0'	No	0	Yes	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: progress

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique

id	Progress ID	bigint	auto_increment	No	-	Yes	Yes
user_id	User ID (FK)	bigint	-	No	-	Yes	No
class_id	Class ID (FK)	bigint	-	No	-	Yes	No
module_id	Module ID (FK)	bigint	-	No	-	Yes	No
completed	Completed status	tinyint(1)	default '0'	No	0	Yes	No
completed_at	Completion timestamp	timestamp	-	Yes	null	No	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: logs

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	Log ID	bigint	auto_increment	No	-	Yes	Yes
admin_id	Admin User ID (FK)	bigint	-	No	-	Yes	No
action	Action taken	varchar(255)	-	No	-	Yes	No
description	Action description	text	-	Yes	null	No	No
created_at	Creation timestamp	timestamp	-	Yes	null	No	No
updated_at	Update timestamp	timestamp	-	Yes	null	No	No

Table: sessions

Attribute Name	Description	Type	Additional Info	Nullable	Default Value	Mandatory	Unique
id	Session ID	varchar(255)	-	No	-	Yes	Yes
user_id	User ID (FK)	bigint	-	Yes	null	No	No
ip_address	IP Address	varchar(45)	-	Yes	null	No	No
user_agent	User Agent	text	-	Yes	null	No	No
payload	Payload	longtext	-	No	-	Yes	No
last_activity	Last Activity Timestamp	int	-	No	-	Yes	

4.0 Component Design

4.1 Package Identifier

Identifier	Package
SDD_PKG_100	View Layer
SDD_PKG_200	Domain Layer
SDD_PKG_300	Data Access Layer

4.1.2 Package Purpose

A Reference Back to The Requirements Specification

The creation and functions of each package are based on the requirements outlined in the requirements specification document, specifically the Software Requirements Specification version 1.4. This document, along with stakeholders' requirements, various websites for supplementary information and consultations with lecturers for advice regarding the project, ensures that each component is designed to meet the specific needs and performance standards necessary for the system's success.

Rationale For Creation of The Package:

Each package has been created to address particular functional and performance needs, ensuring that the system operates efficiently and effectively. Below are the rationales for each package:

- **View Layer (SDD_PKG_100):** This package is created to handle user interactions and ensure a seamless user experience. It provides the necessary interfaces for users to input data and interact with the system. The View Layer meets the requirements for user input handling, data display and updating input data.
- **Domain Layer (SDD_PKG_200):** The Domain Layer is responsible for processing business rules and logic. It abstracts complex business processes and ensures that the system adheres to the specified business rules. This package meets the requirements for business logic processing, data abstraction, and improving the readability of the View Layer.
- **Data Access Layer (SDD_PKG_300):** This package is essential for managing the system's data operations. It establishes connections to the database, maintains these connections and handles CRUD operations. The Data Access Layer meets the requirements for data source creation, database connectivity, data manipulation and user request handling.

4.1.3 Package Function

- *What does the package do?*

PACKAGE	PACKAGE FUNCTION
View Layer (SDD_PKG_100)	• To present data that users can read or create on specific windows • To modify the input data • To manage user interactions • To show specific forms for user data input and system retrieval • To upload data through designated windows
Domain Layer (SDD_PKG_200)	• To represent actual entities in the SLACES • To process business rules for use cases • To utilize straightforward business logic • To merge data and abstract logic from a complex database to enhance the readability of the ViewLayer
Data Access Layer (SDD_PKG_300)	• To establish a data source • To configure a database connection • To sustain data connectivity • To perform CRUD (Create, Read, Update, Delete) operations on data • To enable data search functionality • To handle user requests during sessions

Table 4.1 Package Functions

4.1.4 Package Dependencies

Every package might be dependent on some other packages in the system. These dependencies define the factors that connect one package to the other and the limitations placed on one package by another. The View Layer (SDD_PKG_100) uses the function of the Domain Layer (SDD_PKG_200) for data controlling and business logic calculations. It is required that it should always be together with the Domain Layer to work effectively. Much like the implementation of the Business Logic Layer, the Domain Layer (SDD_PKG_200) has a dependency on the Data Access Layer (SDD_PKG_300) during data access/mutation activities. When it comes to CRUD operations, and database connections, this responsibility lies on the Data Access Layer. SDD_PKG_300 Data Access Layer is the core component of the entire system because the layer will directly work with the database. Disruptions or challenges within this layer can impact the rest of the layers of the given supply chain.

4.1.5 Package Components

PACKAGES	CLASSSES
View Layer	RegisterWindow DashboardWindow JoinClassWindow ViewClassesWindow ClassMaterialsWindow ContactAdminWindow ManageClassesWindow ManageMaterialsWindow ManageInvoicesWindow ViewEnrolledStudentsWindow ViewProgressWindow ViewInvoiceWindow ViewMaterialWindow ViewLogWindow LogoutWindow

Domain Layer	ClassController MaterialController EnrollmentController InvoiceController ProgressController LogController User Class Material Enrollment Invoice Progress
Data Access Layer	UserDAO ClassDAO MaterialDAO EnrollmentDAO InvoiceDAO ProgressDAO LogDAO

4.1.5.1 Class Identifier

Packages	Classes
View Layer (SDD_PKG_100)	RegisterWindow (SDD_PKG_101) DashboardWindow (SDD_PKG_102) JoinClassWindow (SDD_PKG_103) ViewClassesWindow (SDD_PKG_104) ClassMaterialsWindow (SDD_PKG_105) ContactAdminWindow (SDD_PKG_106) ManageClassesWindow (SDD_PKG_107) ManageMaterialsWindow (SDD_PKG_108) ManageInvoicesWindow (SDD_PKG_109) ViewEnrolledStudentsWindow (SDD_PKG_110) ViewProgressWindow (SDD_PKG_111) ViewInvoiceWindow (SDD_PKG_112) ViewMaterialWindow (SDD_PKG_113) ViewLogWindow (SDD_PKG_114) LogoutWindow (SDD_PKG_115)

Domain Layer (SDD_PKG_200)	ClassController (SDD_PKG_201) MaterialController (SDD_PKG_202) EnrollmentController (SDD_PKG_203) InvoiceController (SDD_PKG_204) ProgressController (SDD_PKG_205) LogController (SDD_PKG_206) User (SDD_PKG_207) Class (SDD_PKG_208) Material (SDD_PKG_209) Enrollment (SDD_PKG_210) Invoice (SDD_PKG_211) Progress (SDD_PKG_212)
Data Access Layer (SDD_PKG_300)	UserDAO (SDD_PKG_301) ClassDAO (SDD_PKG_302) MaterialDAO (SDD_PKG_303) EnrollmentDAO (SDD_PKG_304) InvoiceDAO (SDD_PKG_305) ProgressDAO (SDD_PKG_306) LogDAO (SDD_PKG_307)

4.1.5.2 Class Purpose

View Layer

Class	**Purpose**
RegisterWindow (SDD_PKG_101)	Provides user registration interface, collects required details, validates input, and creates new user accounts.
DashboardWindow (SDD_PKG_102)	Displays user dashboard with summary information, navigation to main features, and quick access to recent activities.
JoinClassWindow (SDD_PKG_103)	Allows users to join available classes, view class info, and submit join requests.
ViewClassesWindow	Shows all available classes with details for browsing,

(SDD_PKG_104)	searching, and filtering.
ClassMaterialsWindow (SDD_PKG_105)	Displays learning materials for a selected class, allowing users to view, download, or interact with resources.
ContactAdminWindow (SDD_PKG_106)	Enables users to contact administrators for support, feedback, or inquiries via forms or messaging.
ManageClassesWindow (SDD_PKG_107)	Provides admin interface for creating, updating, and managing classes and their settings.
ManageMaterialsWindow (SDD_PKG_108)	Allows admins to upload, edit, organize, and delete class materials.
ManageInvoicesWindow (SDD_PKG_109)	Enables management of invoices for classes, including viewing, generating, and updating payment records.
ViewEnrolledStudentsWindow (SDD_PKG_110)	Displays list of students enrolled in a class, with options to view profiles and track progress.
ViewProgressWindow (SDD_PKG_111)	Shows student progress data for a class, module, or material, highlighting completion rates and activity.
ViewInvoiceWindow (SDD_PKG_112)	Lets users and admins view details of invoices, payment status, and related actions.
ViewMaterialWindow (SDD_PKG_113)	Displays details of a single material, including type, description, and access/download options.
ViewLogWindow (SDD_PKG_114)	Shows logs of system or user actions, providing audit trails for admin review.
LogoutWindow (SDD_PKG_115)	Handles user logout process, clears sessions, and redirects to login or home screen.

Domain Layer

Class (Code)	**Purpose**
ClassController (SDD_PKG_201)	Coordinates creation, update, deletion, and retrieval of class data and manages business logic for classes.
MaterialController (SDD_PKG_202)	Manages CRUD operations and logic for class materials.
EnrollmentController (SDD_PKG_203)	Facilitates user enrollment, status updates, and progress tracking in classes.
InvoiceController (SDD_PKG_204)	Handles invoice creation, validation, payment status updates, and retrieval.

ProgressController (SDD_PKG_205)	Manages progress tracking for users across classes and modules.
LogController (SDD_PKG_206)	Processes system and admin activity logs, provides access to logs, and manages log records.
User (SDD_PKG_207)	Represents a system user with identification, authentication, and profile details.
Class (SDD_PKG_208)	Models a learning class with attributes such as title, description, price, and schedule.
Material (SDD_PKG_209)	Represents a learning resource associated with a class, including files, links, or documents.
Enrollment (SDD_PKG_210)	Tracks a user's enrollment in a class, including status and progress.
Invoice (SDD_PKG_211)	Represents billing and payment details for class enrollments.
Progress (SDD_PKG_212)	Captures user progress for class modules and resources.

Data Access Layer

Class (Code)	**Purpose**
UserDAO (SDD_PKG_301)	Handles database operations for users, including CRUD actions and queries.
ClassDAO (SDD_PKG_302)	Manages class-related database transactions, such as creation, updates, and retrievals.
MaterialDAO (SDD_PKG_303)	Executes data access functions for class materials.
EnrollmentDAO (SDD_PKG_304)	Facilitates enrollment data operations, linking users and classes in the database.
InvoiceDAO (SDD_PKG_305)	Processes invoice data, including creation, updates, and queries.
ProgressDAO (SDD_PKG_306)	Handles database operations for progress tracking and updates.
LogDAO (SDD_PKG_307)	Manages log entries in the database, enabling retrieval and storage of system activity records.

4.1.5.3 Class function

- *What does the class do?*

View Layer

Class	Pseudocode
RegisterWindow (SDD_PKG_101)	BEGIN 1. Display RegisterWindow END
DashboardWindow (SDD_PKG_102)	BEGIN 1. Display DashboardWindow END
JoinClassWindow (SDD_PKG_103)	BEGIN 1. Display JoinClassWindow END
ViewClassesWindow (SDD_PKG_104)	BEGIN 1. Display ViewClassesWindow END
ClassMaterialsWindow (SDD_PKG_105)	BEGIN 1. Display ClassMaterialsWindow END
ContactAdminWindow (SDD_PKG_106)	BEGIN 1. Display ContactAdminWindow END
ManageClassesWindow (SDD_PKG_107)	BEGIN 1. Display ManageClassesWindow END
ManageMaterialsWindow (SDD_PKG_108)	BEGIN 1. Display ManageMaterialsWindow

	END
ManageInvoicesWindow (SDD_PKG_109)	BEGIN 1. Display ManageInvoicesWindow END
ViewEnrolledStudentsWindow (SDD_PKG_110)	BEGIN 1. Display ViewEnrolledStudentsWindoow END
ViewProgressWindow (SDD_PKG_111)	BEGIN 1. Display ViewProgressWindow END
ViewInvoiceWindow (SDD_PKG_112)	BEGIN 1. Display ViewInvoiceWindow END
ViewMaterialWindow (SDD_PKG_113)	BEGIN 1. Display ViewMaterialWindow END
ViewLogWindow (SDD_PKG_114)	BEGIN 1. Display ViewLogWindow END
LogoutWindow (SDD_PKG_115)	BEGIN 1. Display LogoutWindow END

Domain Layer

Class	**Pseudocode**	**Method(s)**
ClassController (SDD_PKG_201)	BEGIN 1. Retrieve class data from database.	create(), update(), delete()

	2. Display classes information. If method is create class: 1. Create new class object. 2. Store class data. 3. Set class status to active. If method is update class: 1. Retrieve class object. 2. Save updated class data. If method is delete class: 1. Remove class object. END	
MaterialController (SDD_PKG_202)	BEGIN 1. Retrieve materials for class. 2. Display materials. If method is add material: 1. Create material object. 2. Store material data. If method is update material: 1. Save updated material data.	add(), update(), delete()

	If method is delete material: 1. Remove material object. END	
EnrollmentController (SDD_PKG_203)	BEGIN 1. Retrieve enrollment info. 2. Display enrollment status. If method is enroll: 1. Create enrollment object. 2. Store enrollment data. If method is update enrollment: 1. Save updated enrollment status. If method is delete enrollment: 1. Remove enrollment object. END	enroll(), update(), delete()
InvoiceController (SDD_PKG_204)	BEGIN 1. Retrieve invoice info. 2. Display invoice details. If method is create invoice: 1. Create invoice object. 2. Store invoice	create(), update(), delete()

	data. If method is update invoice: 1. Save updated invoice status. If method is delete invoice: 1. Remove invoice object. END	
ProgressController (SDD_PKG_205)	BEGIN 1. Retrieve progress for user/class/module. 2. Display progress data. If method is update progress: 1. Save progress completion. END	view(), update()
LogController (SDD_PKG_206)	BEGIN 1. Retrieve log entries. 2. Display logs. If method is add log: 1. Create log object. 2. Store log data. If method is delete log: 1. Remove log object. END	add(), delete()
User	BEGIN	register(),

(SDD_PKG_207)	1. Store and manage user data. 2. Authenticate user credentials. 3. Update profile info. END	login(), updateProfile()
Class (SDD_PKG_208)	BEGIN 1. Store class attributes (title, description, etc). 2. Link class to materials, enrollments. END	setTitle(), setDescription()
Material (SDD_PKG_209)	BEGIN 1. Store material attributes (type, file, link, etc). 2. Link material to class. END	setType(), setFile()
Enrollment (SDD_PKG_210)	BEGIN 1. Store enrollment user/class relation. 2. Track enrollment status/progress. END	setStatus(), setProgress()
Invoice (SDD_PKG_211)	BEGIN 1. Store invoice details (amount, status, class). 2. Link invoice to user/class. END	setAmount(), setStatus()
Progress	BEGIN	setCompleted(),

(SDD_PKG_212)	1. Track user/class/module completion. 2. Manage progress timestamps. END	setTimestamp()

Data Access Layer

Class	**Pseudocode**	**Method(s)**
UserDAO (SDD_PKG_301)	BEGIN findUser(bigint(20) userID) 1. Create connection, statement, resultset variable 2. Establish connection with database 3. Execute SQL statement to get user details based on userID and assign it to resultset variable 4. Set id, name, email, role, etc. from resultset data to user object 5. Return user object 6. Catch exception 7. Display exception createUser(User user) 1. Create connection, prepared statement variable 2. Establish connection with	findUser(bigint(20) userID): User createUser(User user): String updateUser(User user): String deleteUser(bigint(20) userID): bigint(20)

	database 3. Execute SQL statement to insert new user details into user table 4. Return 'SUCCESS' 5. Catch exception 6. Display exception updateUser(User user) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to update user details in user table 4. Return 'UPDATE SUCCESS' 5. Catch exception 6. Return 'UPDATE FAILED' deleteUser(bigint(20) userID) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to delete	

	user based on userID from user table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE FAILED' END	
ClassDAO (SDD_PKG _302)	BEGIN findClass(bigint(20) classID) 1. Create connection, statement, resultset variable 2. Establish connection with database 3. Execute SQL statement to get class details based on classID and assign it to resultset variable 4. Set id, title, description, price, etc. from resultset data to class object 5. Return class object 6. Catch exception 7. Display exception createClass(Class class) 1. Create connection, prepared statement variable 2. Establish	findClass(bigint(20) classID): Class createClass(Class class): String updateClass(Class class): String deleteClass(bigint(20) classID): bigint(20)

	connection with database 3. Execute SQL statement to insert new class details into class table 4. Return 'SUCCESS' 5. Catch exception 6. Display exception updateClass(Class class) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to update class details in class table 4. Return 'UPDATE SUCCESS' 5. Catch exception 6. Return 'UPDATE FAILED' deleteClass(bigint(20) classID) 1. Create connection, prepared statement variable 2. Establish connection with database	

	3. Execute SQL statement to delete class based on classID from class table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE FAILED' END	
MaterialDAO (SDD_PKG_303)	BEGIN findMaterial(bigint(20) materialID) 1. Create connection, statement, resultset variable 2. Establish connection with database 3. Execute SQL statement to get material details based on materialID and assign it to resultset variable 4. Set id, title, type, file_path, etc. from resultset data to material object 5. Return material object 6. Catch exception 7. Display exception createMaterial(Materia	findMaterial(bigint(20) materialID): Material createMaterial(Material material): String updateMaterial(Material material): String deleteMaterial(bigint(20) materialID): bigint(20)

	l material) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to insert new material details into material table 4. Return 'SUCCESS' 5. Catch exception 6. Display exception updateMaterial(Materi al material) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to update material details in material table 4. Return 'UPDATE SUCCESS' 5. Catch exception 6. Return 'UPDATE FAILED' deleteMaterial(bigint(2 0) materialID) 1. Create connection,	

	prepared statement variable 2. Establish connection with database 3. Execute SQL statement to delete material based on materialID from material table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE FAILED' END	
Enrollment DAO (SDD_PKG _304)	BEGIN findEnrollment(bigint(20) enrollmentID) 1. Create connection, statement, resultset variable 2. Establish connection with database 3. Execute SQL statement to get enrollment details based on enrollmentID and assign it to resultset variable 4. Set id, user_id, class_id, status, etc. from resultset data to	findEnrollment(bigint(20) enrollmentID): Enrollment createEnrollment(Enro llment enrollment): String updateEnrollment(Enr ollment enrollment): String deleteEnrollment(bigin t(20) enrollmentID): bigint(20)

	enrollment object 5. Return enrollment object 6. Catch exception 7. Display exception createEnrollment(Enrollment enrollment) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to insert new enrollment details into enrollment table 4. Return 'SUCCESS' 5. Catch exception 6. Display exception updateEnrollment(Enrollment enrollment) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to update enrollment details in enrollment table 4. Return 'UPDATE SUCCESS'	

	5. Catch exception 6. Return 'UPDATE FAILED' deleteEnrollment(bigint(20) enrollmentID) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to delete enrollment based on enrollmentID from enrollment table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE FAILED' END	
InvoiceDAO (SDD_PKG_305)	BEGIN findInvoice(bigint(20) invoiceID) 1. Create connection, statement, resultset variable 2. Establish connection with database 3. Execute SQL statement to get invoice details based	findInvoice(bigint(20) invoiceID): Invoice createInvoice(Invoice invoice): String updateInvoice(Invoice invoice): String deleteInvoice(bigint(20) invoiceID): bigint(20)

	on invoiceID and assign it to resultset variable 4. Set id, invoice_number, user_id, amount, etc. from resultset data to invoice object 5. Return invoice object 6. Catch exception 7. Display exception createInvoice(Invoice invoice) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to insert new invoice details into invoice table 4. Return 'SUCCESS' 5. Catch exception 6. Display exception updateInvoice(Invoice invoice) 1. Create connection, prepared statement variable 2. Establish connection with	

	database 3. Execute SQL statement to update invoice details in invoice table 4. Return 'UPDATE SUCCESS' 5. Catch exception 6. Return 'UPDATE FAILED' deleteInvoice(bigint(20) invoiceID) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to delete invoice based on invoiceID from invoice table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE FAILED' END	
ProgressDAO (SDD_PKG_306)	BEGIN findProgress(bigint(20) progressID) 1. Create connection, statement, resultset	findProgress(bigint(20) progressID): Progress createProgress(Progress progress): String

	variable 2. Establish connection with database 3. Execute SQL statement to get progress details based on progressID and assign it to resultset variable 4. Set id, user_id, class_id, module_id, completed, etc. from resultset data to progress object 5. Return progress object 6. Catch exception 7. Display exception createProgress(Progress progress) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to insert new progress details into progress table 4. Return 'SUCCESS' 5. Catch exception 6. Display exception	updateProgress(Progress progress): String deleteProgress(bigint(20) progressID): bigint(20)

	updateProgress(Progre ss progress) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to update progress details in progress table 4. Return 'UPDATE SUCCESS' 5. Catch exception 6. Return 'UPDATE FAILED' deleteProgress(bigint(20) progressID) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to delete progress based on progressID from progress table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE	

	FAILED' END	
LogDAO (SDD_PKG _307)	BEGIN findLog(bigint(20) logID) 1. Create connection, statement, resultset variable 2. Establish connection with database 3. Execute SQL statement to get log details based on logID and assign it to resultset variable 4. Set id, admin_id, action, description, etc. from resultset data to log object 5. Return log object 6. Catch exception 7. Display exception createLog(Log log) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to insert new log details into logs table	findLog(bigint(20) logID): Log createLog(Log log): String updateLog(Log log): String deleteLog(bigint(20) logID): bigint(20)

	4. Return 'SUCCESS' 5. Catch exception 6. Display exception updateLog(Log log) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to update log details in logs table 4. Return 'UPDATE SUCCESS' 5. Catch exception 6. Return 'UPDATE FAILED' deleteLog(bigint(20) logID) 1. Create connection, prepared statement variable 2. Establish connection with database 3. Execute SQL statement to delete log based on logID from logs table 4. Return 'DELETE SUCCESS' 5. Catch exception 6. Return 'DELETE	

	FAILED' END	

4.1.5.4 Class subordinates

Parent Table	Child Table	Foreign Key
users	enrollments	user_id
users	invoices	user_id
users	logs	admin_id
classes	enrollments	class_id
classes	invoices	class_id
classes	materials	class_id
users	progress	user_id
classes	progress	class_id
materials	progress	module_id

4.1.5.5 Class Dependencies

The Design Class Diagram illustrates the interdependence between classes, regardless of whether they are in the same package. This diagram displays the relationships between classes, showing how classes in the view layer depend on classes in the domain and how classes in the domain depend on classes in the data access layer.

4.1.5.6 Interfaces

The control and data flow from one class to another can be seen in the Figure 2.2 Detailed Design Class Diagram, Decomposition Description 2.3.

4.1.5.7 Data

All data that is being used can be referred to in the Data Dictionary in Section 3, Data Dictionary 3.3.

5.0 Human Interface Design (Screens)

5.1 Screen Images

The Human Interface Design of the Smootea Learning Access & Class Enrollment System is a visual of the main functions of the proposed system. It offers stakeholders a preview of the user interface and feature flow of the system early enough, where they can evaluate and validate it before it is implemented. The prototype is based on core modules which include student registration, joining classes, accessing class content and admin controls like invoice management and material management.

Landing Page

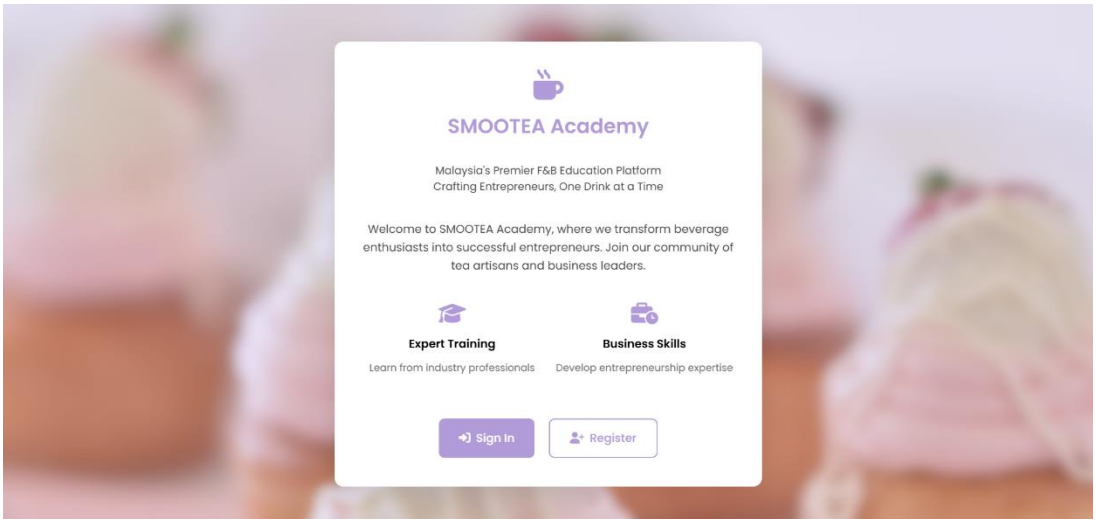

Figure 5.1.1 Landing Page

The landing page introduces users to SLACES, featuring options to register or log in.

Register Page

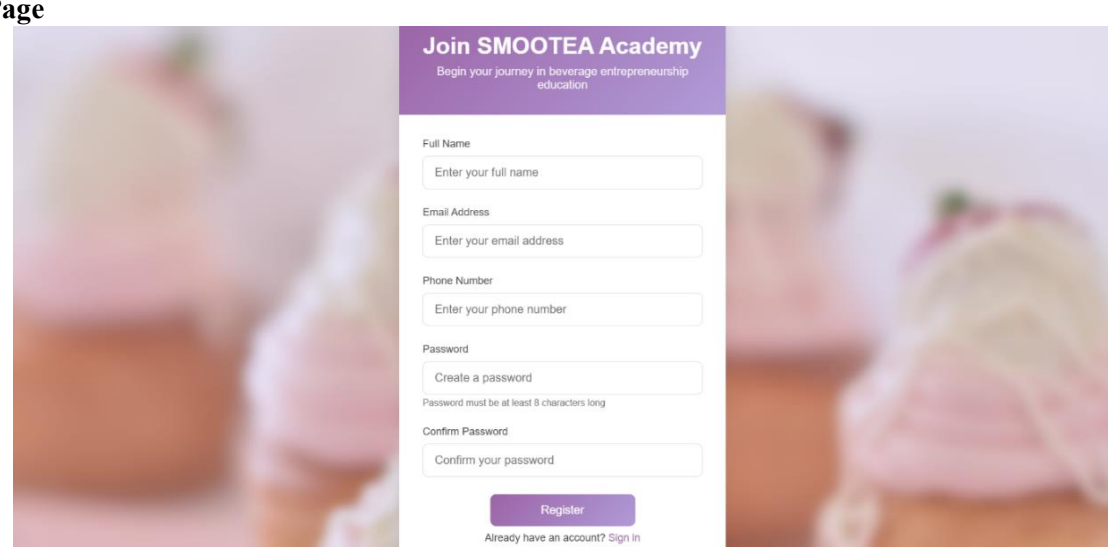

Figure 5.1.2 Register Page

Registration form for new users, requiring name, email, and password. The system validates inputs and ensures email uniqueness before account creation.

Login Page

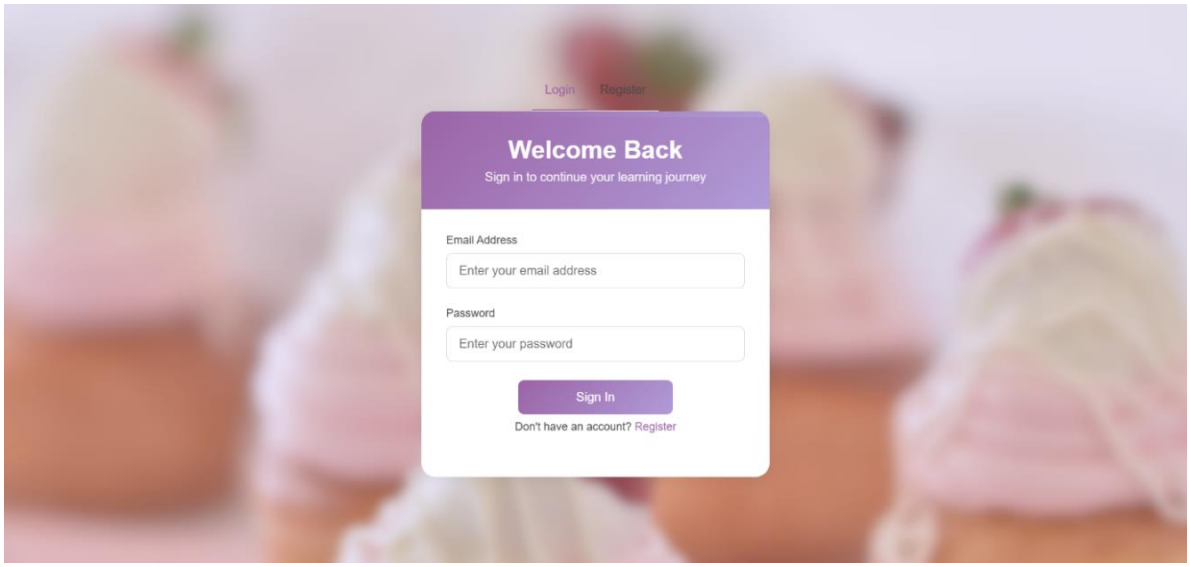

Figure 5.1.3 Login Page

Login interface for students and admins. Role-based redirection directs users to their respective dashboards upon successful authentication.

Student Dashboard

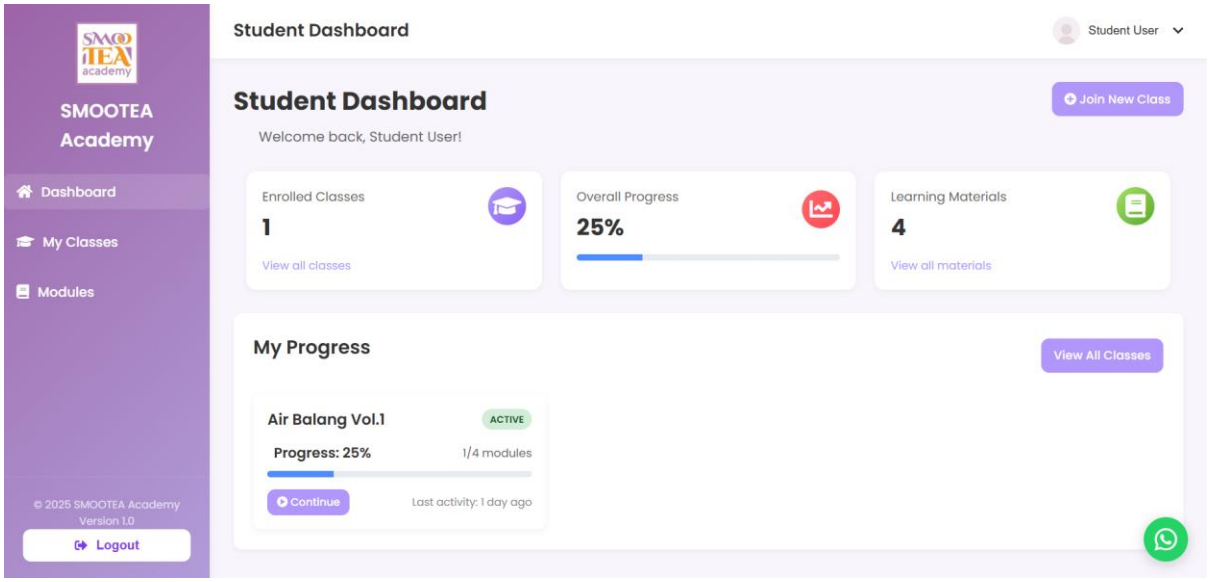

Figure 5.1.4 Student Dashboard

Personalized dashboard for students, displaying active classes, recent activity, and progress metrics. Includes quick access to learning resources and support

View My Class Page

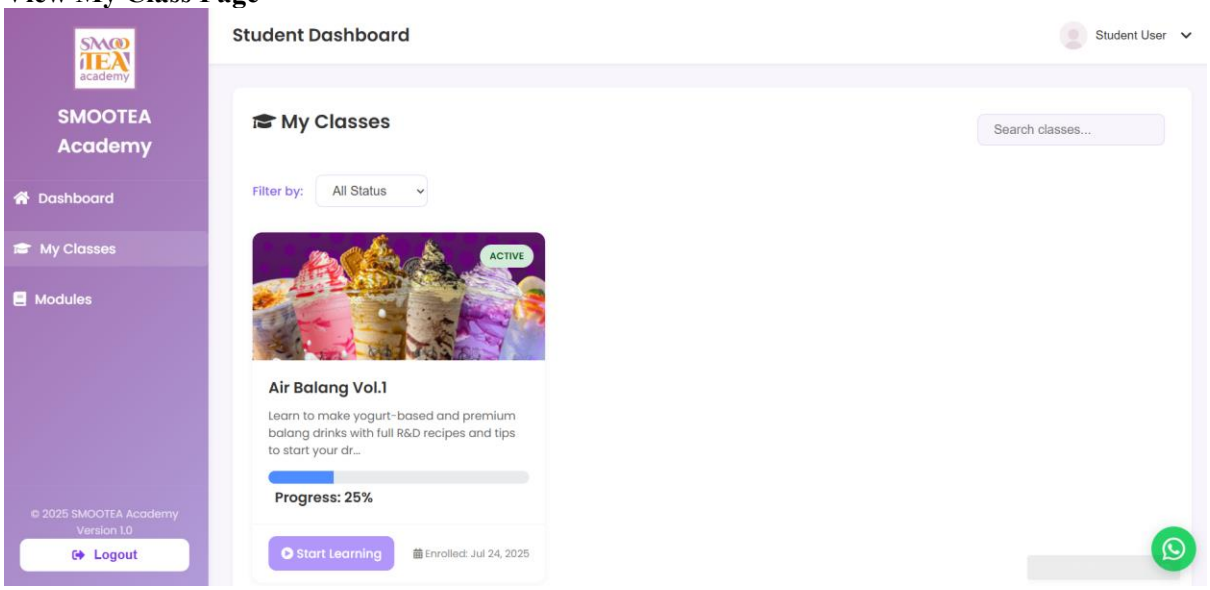

Figure 5.1.5 View My Classes Page

List of enrolled classes with titles, instructors, and access buttons. Students can navigate to specific class materials from this centralized view

Module Page

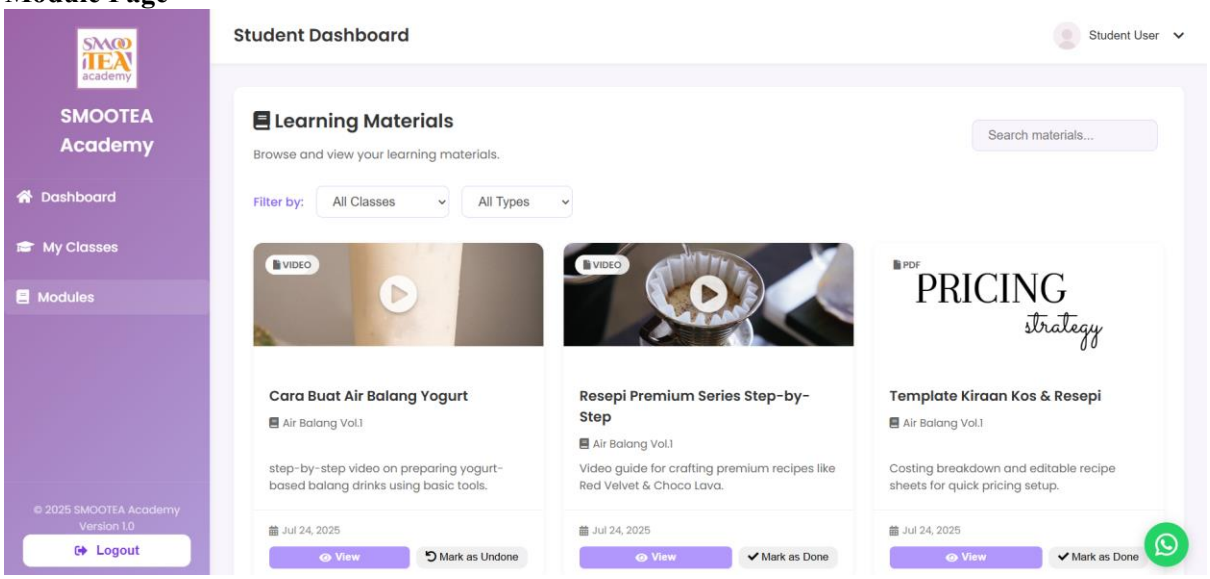

Figure 5.1.7 Module Page

Class materials hub, featuring downloadable/streamable content (PDFs, videos, links). Organized by module for structured learning.

Admin Dashboard

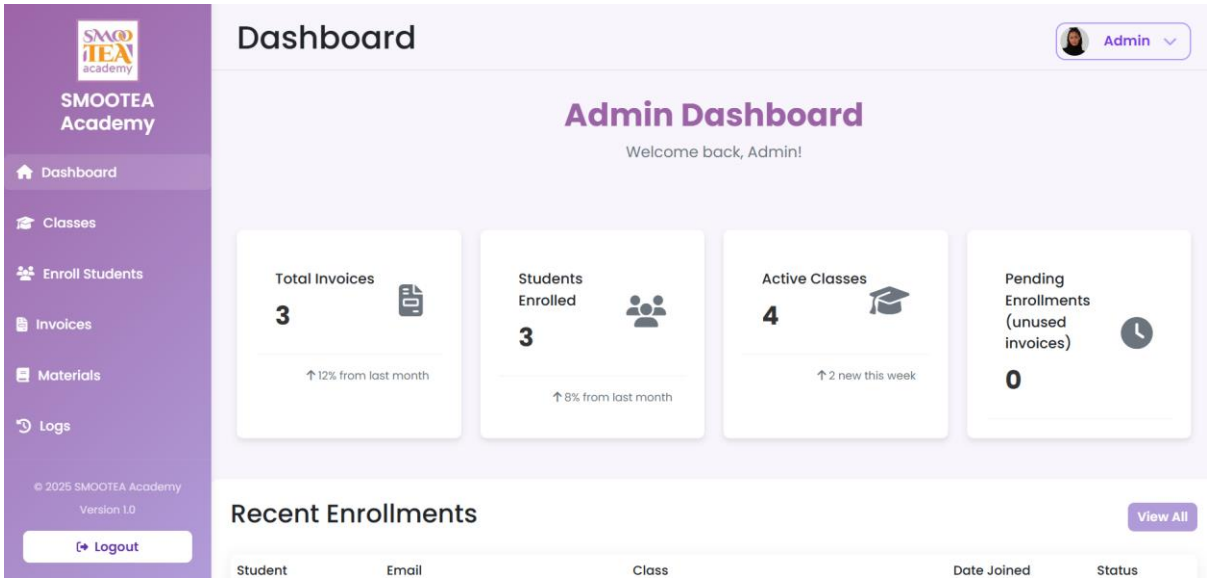

Figure 5.1.8 Admin Dashboard Page

Admin overview panel showing system metrics (total classes, students, invoices). Provides quick access to management tools and recent activity logs.

Manage Classes

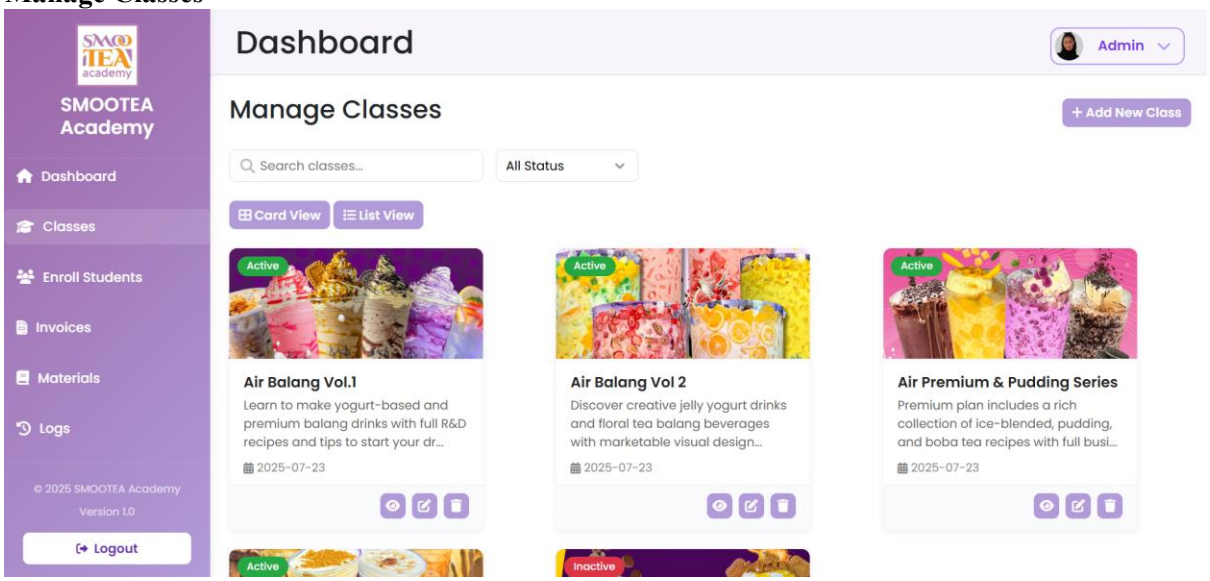

Figure 5.1.9 Admin Manage Classes

Class management interface for admins. Supports CRUD operations (Create, Read, Update, Delete) for class records, including titles and status assignments.

Admin Manage Classes – Add new Class

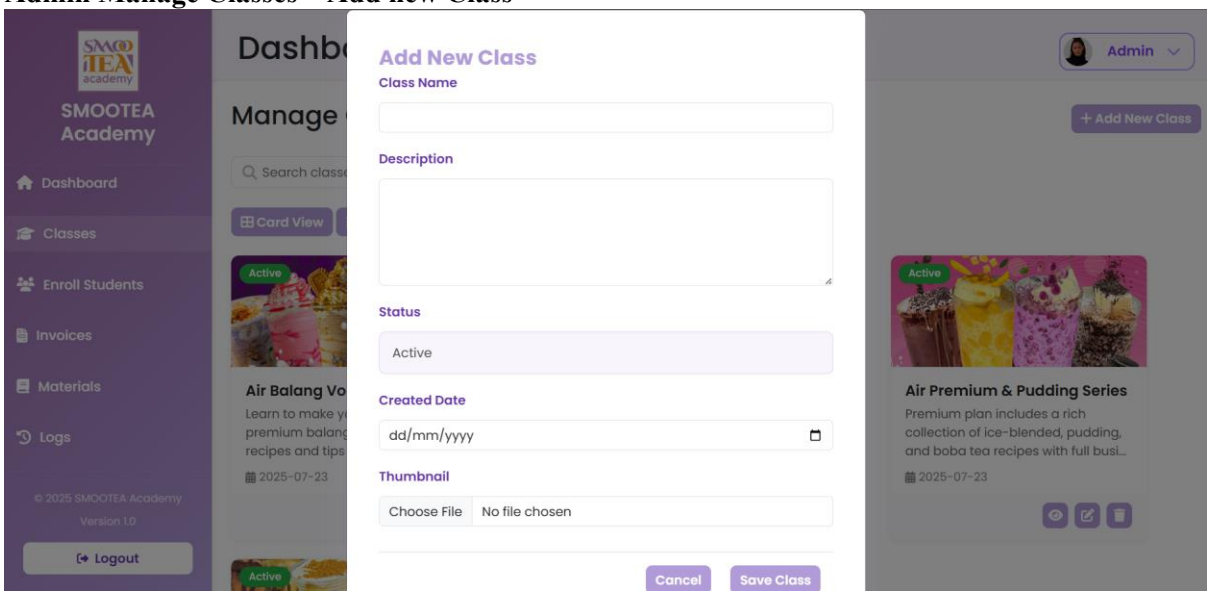

Figure 5.1.10 Admin Add New Class

Form to add new classes. Mandatory fields include title, description, and instructor. Ensures data consistency for future enrollments.

Admin Manage Classes – Edit Class

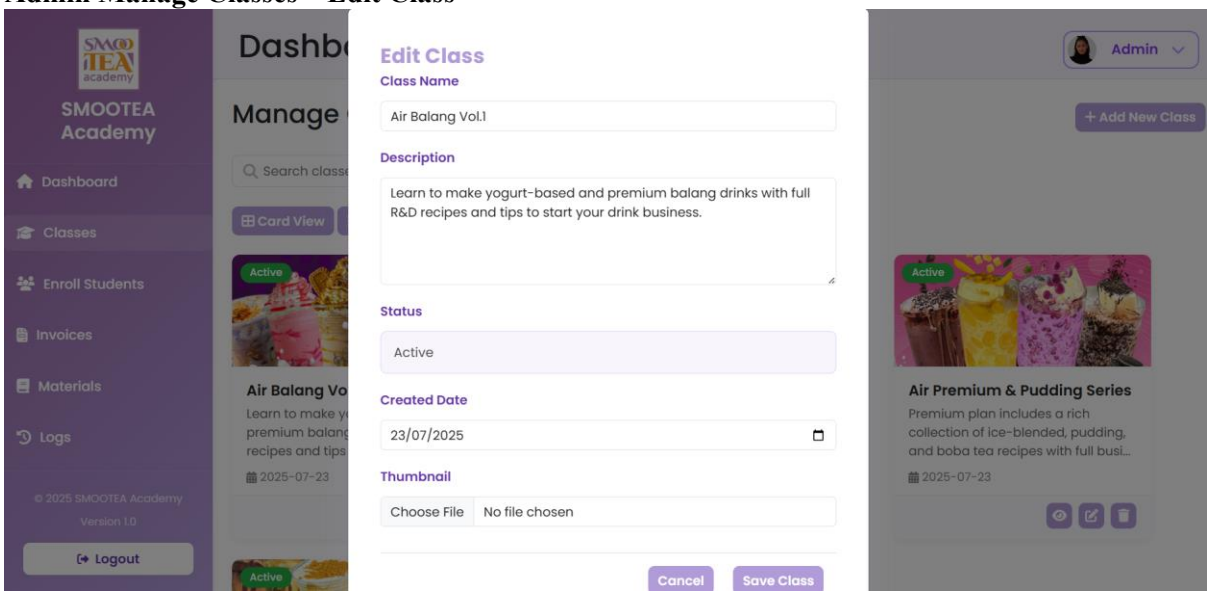

Figure 5.1.11 Admin Edit Classes

Class editing interface. Admins modify existing class details (e.g., title, description, status and thumbnail) with real-time database updates.

Admin Manage Classes – Delete Class

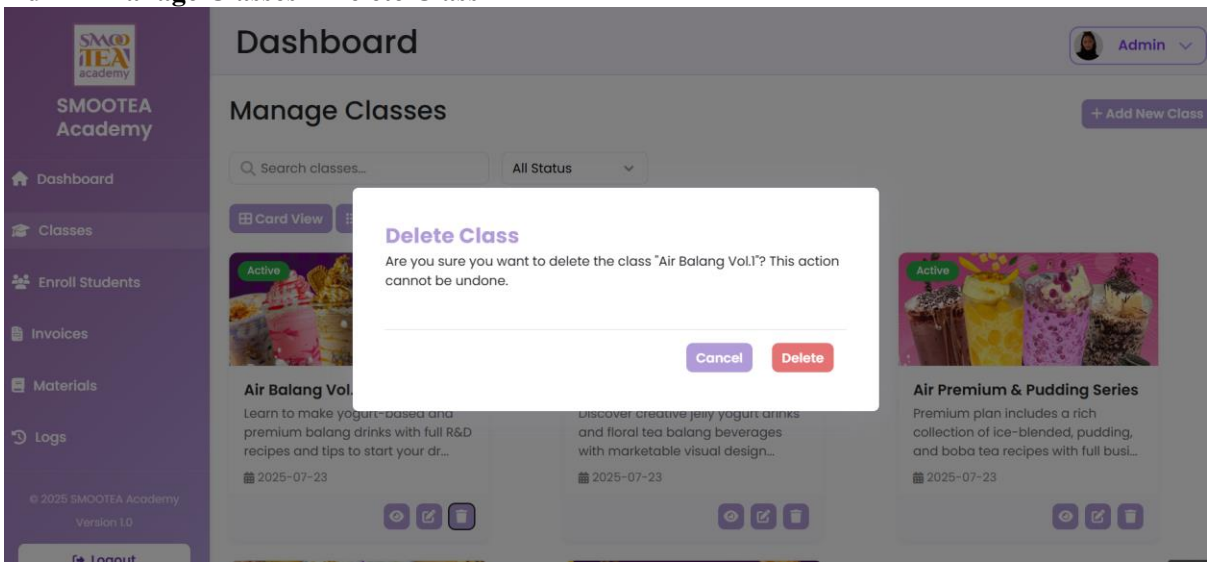

Figure 5.1.12 Admin Delete Classes

Confirmation dialog for class deletion. Prevents accidental data loss with a verification step.

Manage Class Materials

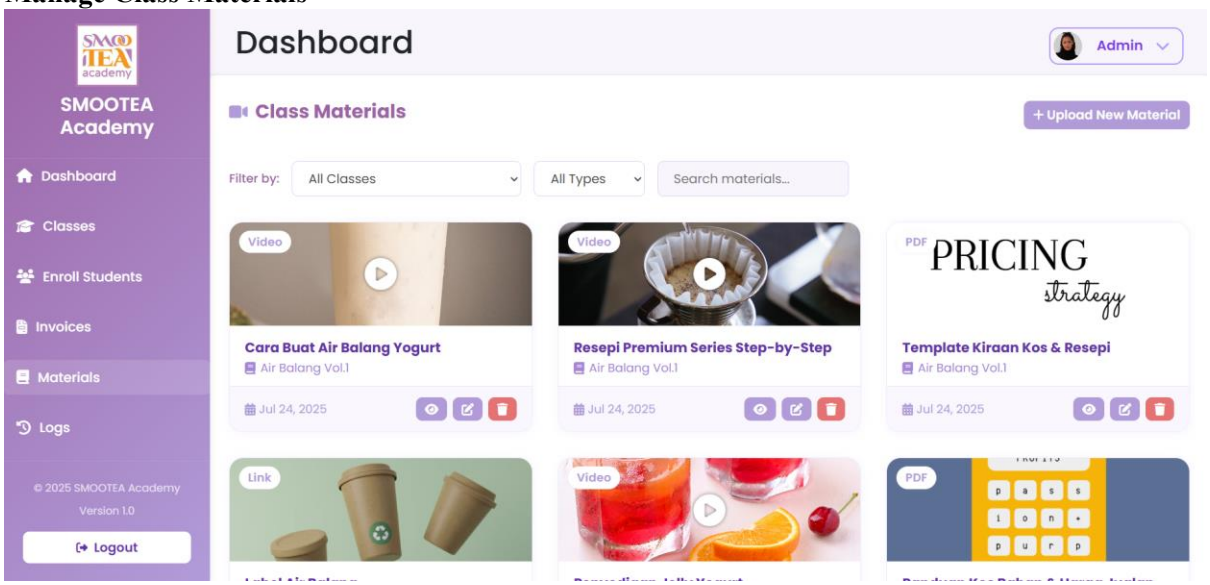

Figure 5.1.13 Admin Manage Class Materials

Material management panel. Admins upload, tag and organize resources (PDFs, videos) by class, with filtering options for efficiency.

Manage Class Materials – Upload New Material

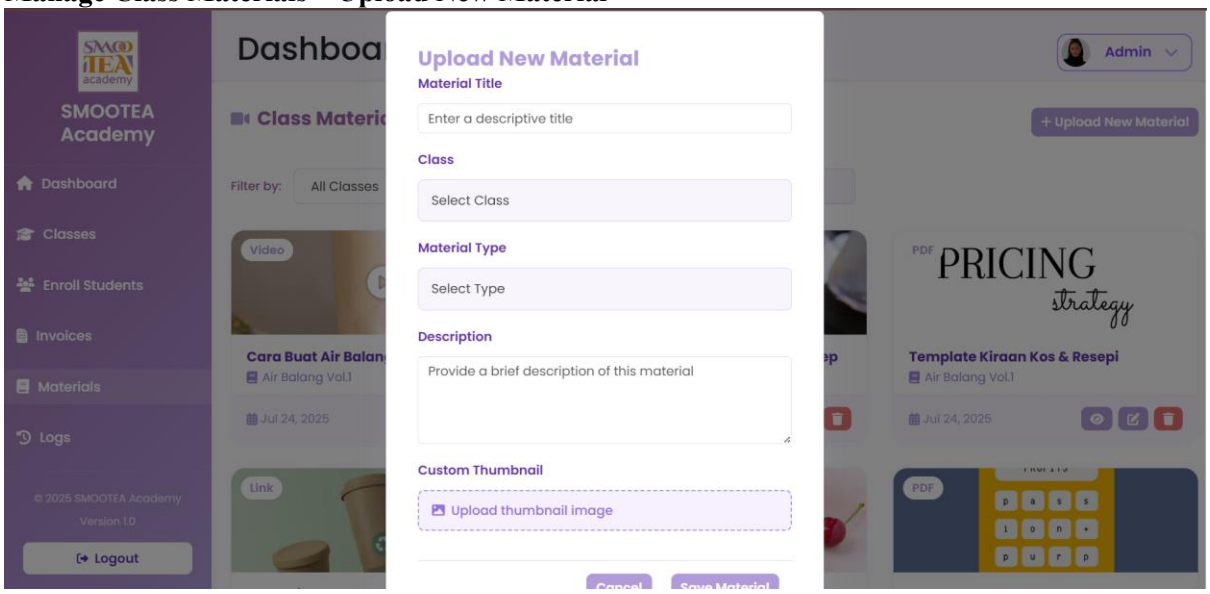

Figure 5.1.14 Admin Upload New Material

Form for uploading new learning materials. Supports files (PDF, video) or external links, with fields for title, type, and description.

Manage Class Materials – View Material

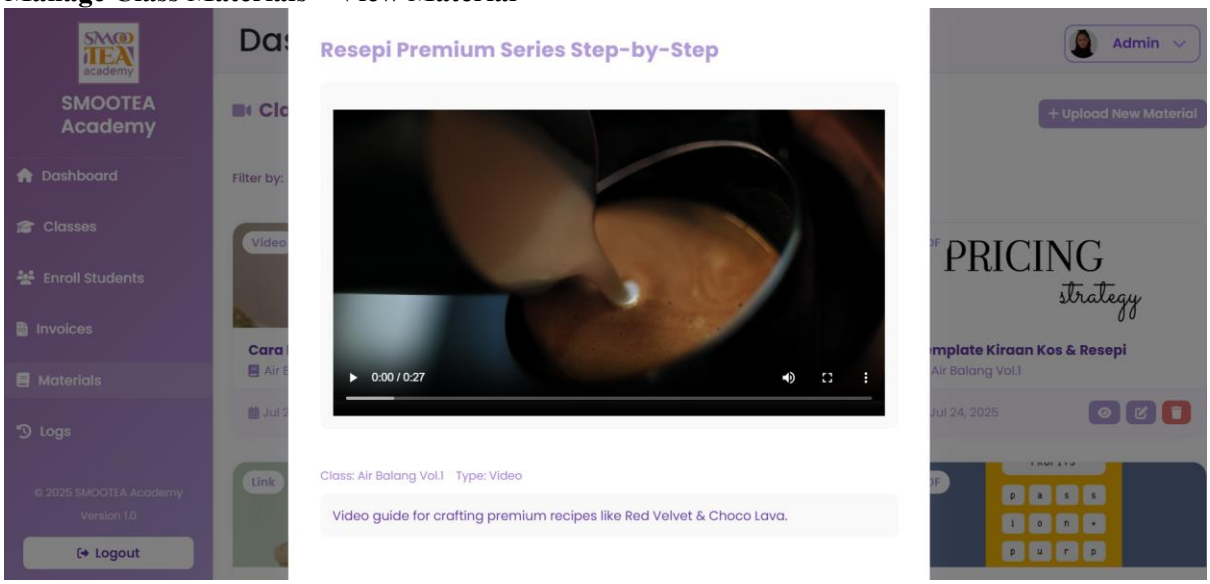

Figure 5.1.15 Admin View Material

Preview mode for uploaded materials. Admins verify content accuracy before publishing to students.

Manage Class Materials – Edit Material

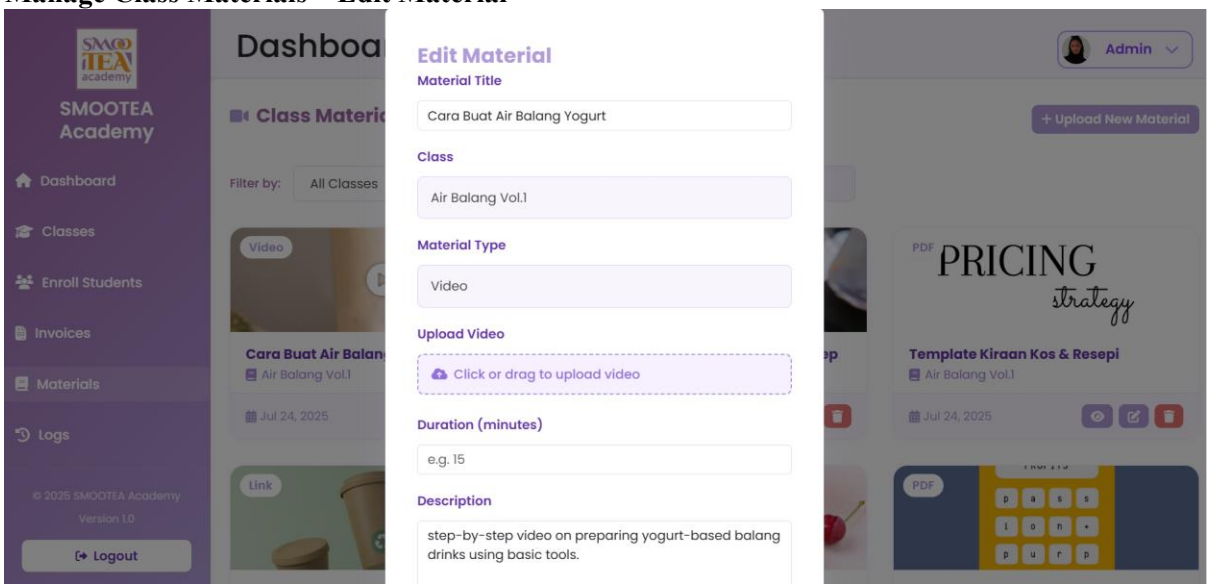

Figure 5.1.16 Admin Edit Material

Edit interface for existing materials. Admins update metadata (title, description) or replace files as needed.

Manage Class Materials – Delete Material

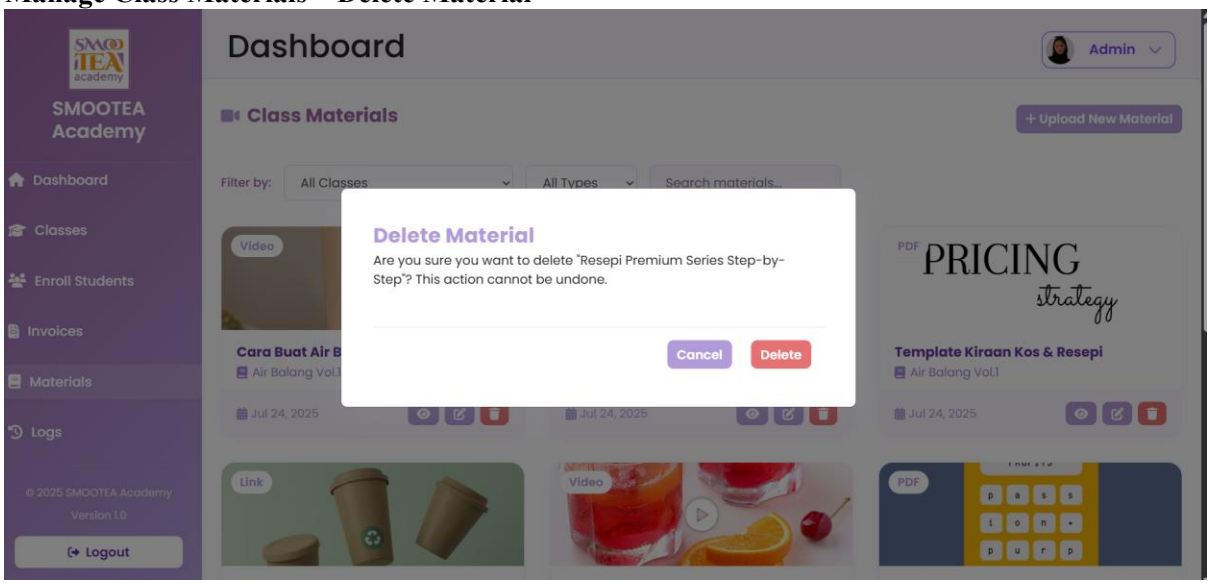

Figure 5.1.17 Admin Delete Material

Material deletion confirmation. Includes safeguards to prevent unintended removal of critical resources.

Admin Manage Invoices

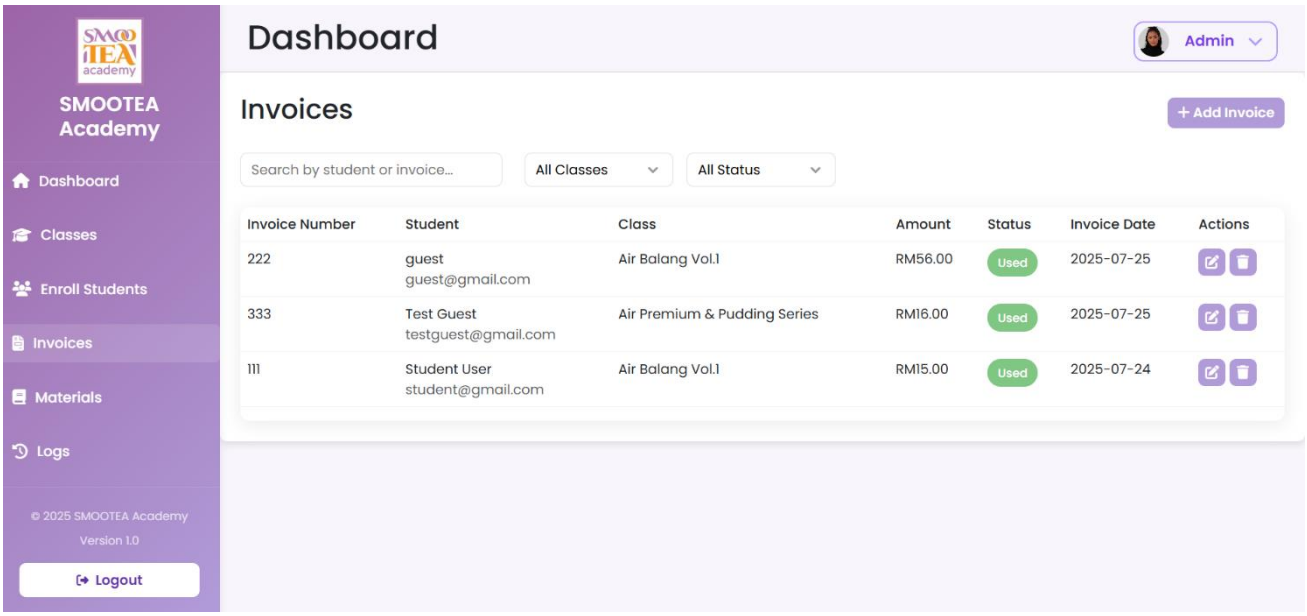

Figure 5.1.18 Admin Manage Invoices

Invoice management table. Admins add, edit or delete invoice records to validate student enrollments.

Admin View Enrolled Student

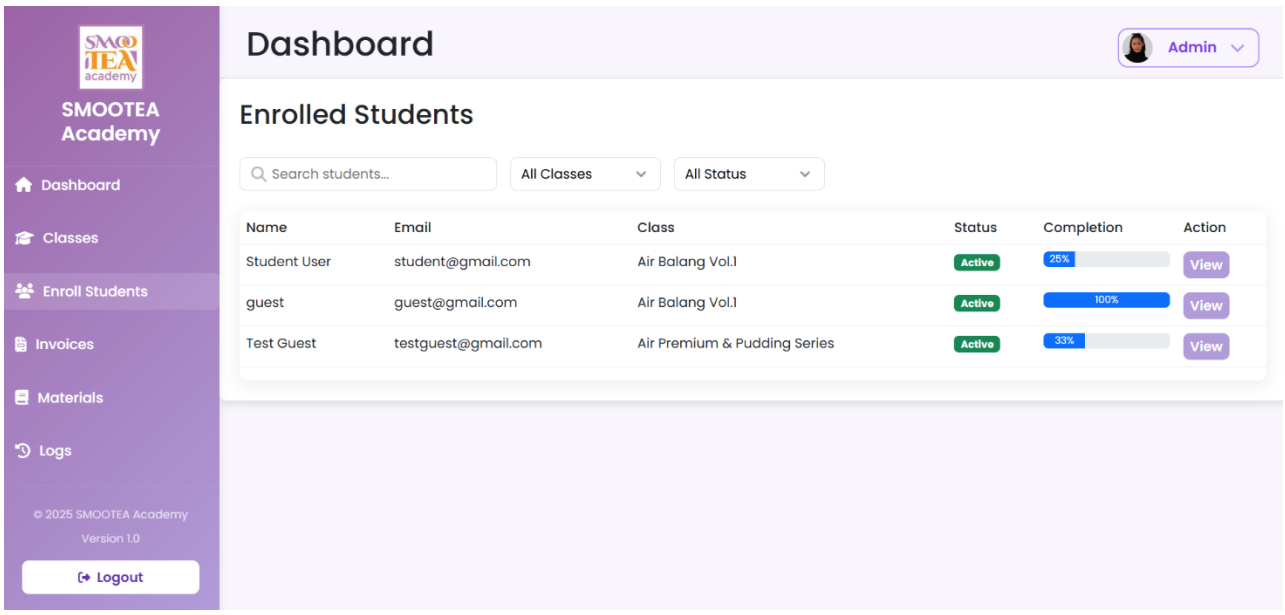

Figure 5.1.19 Admin View Enrolled Student

Enrollment oversight panel. Admins view enrolled student and their progress.

5.2 Overview of the User Interface

5.2.1 General Functionality of the System from the User's Perspective

System Overview:

The SLACES system automates the learning class enrollment and material access process, replacing manual methods. It allows students to register, join classes, access class materials, and track progress. Admin staff can manage classes, approve enrollments, upload materials, and view system logs and invoices.

User Interfaces:

- Student Interface:

  Enables students to register, join classes, view enrolled status, access materials, track progress, and view invoices.

- Admin Interface:

  Allows admin staff to manage classes, materials, student enrollments, invoices and system logs.

5.2.2 How Users Will Use the System to Complete Expected Features and Feedback

1. Login and Authentication:
   - Process: Users log in using their credentials.
   - Feedback: Users gain access to the system with correct credentials or receive an error message if credentials are incorrect.

2. Register and Join Class:
   - Process: Students register an account and browse available classes. They can join a class by submitting a request.
   - Feedback: Success message is displayed upon registration and successful class join; errors are shown if prerequisites are not met or data is invalid.

3. View Classes and Materials:
   - Process: Students navigate to 'View Classes' to see available courses, then access learning materials in 'Class Materials'.
   - Feedback: System displays available classes and materials. If no materials are available, an empty list is shown.

4. Track Enrollment and Progress:

- Process: Students view their enrollments and progress on dedicated pages.
- Feedback: System provides detailed progress charts and completion percentages. If progress is not available, a blank or zero progress is shown.

5. View Invoice:

- Process: Students view invoices for class enrollments and payments.
- Feedback: System displays invoice details, payment status, and downloadable receipts.

6. Manage Classes (Admin):

- Process: Admins can create, edit, or delete classes, with confirmation prompts for deletions ("Are you sure?").
- Feedback: Success message appears after addition, update, or deletion of classes. For deletions, confirmation is required.

7. Manage Materials (Admin):

- Process: Admins upload, edit, or remove class materials.
- Feedback: System confirms successful upload or removal of materials and displays updated list.

8. Manage Student Enrollment (Admin):

- Process: Admins approve or reject enrollment requests and view student progress.
- Feedback: Success message is shown when enrollment status is updated.

9. View and Manage Invoices (Admin):

- Process: Admins view, create, and update invoices for class enrollments.
- Feedback: Success message is displayed when invoice is processed or updated.

10. View System Logs (Admin):

- Process: Admins can review system logs for actions and events.
- Feedback: Logs are presented in detail; if no logs exist, an empty list is shown.

11. Logout:

- Process: Users can securely log out from the system.

- Feedback: User session is ended, and user is redirected to the login page.

5.3 Screen Objects and Actions

1. Admin Pages

Manage Classes Page

- Add New Class Button: Opens a popup form to enter class details and create a new class.

- Edit Button (on class card): Opens a popup to edit selected class details.

- Delete Button (on class card): Opens a confirmation dialog to delete the selected class.

- Class Cards: Display class title, ID, price, status, and thumbnail for easy browsing.

Manage Materials Page

- Add Material Button: Opens a form to upload new learning material or resource.

- Edit Button (on material entry): Opens a popup to modify material details.

- Delete Button (on material entry): Opens a confirmation dialog before removal.

- Material List/Table: Shows material title, type (video/pdf/link), class association, and upload date.

Manage Enrollments Page

- Enrollment List/Table: Displays enrollments with columns for Student Name, Class, Status, Date and progress.

- View Details Button: Shows enrollment information.

Manage Invoices Page

- Invoice Table: Shows invoice number, student, class, amount, status, and date.

- View Details Button: Displays invoice information.

- Update Status Button: Changes invoice status (used/unused).

View Logs Page

- Log Table: Displays admin actions with columns for Action, Description, Date, and Admin Name.

View Progress Page

- Progress Table: Shows student progress for each class/module with percentage completion and

last activity date.

2. Student Pages

Register Page

- Registration Form: Collects name, email, password, and other details.

- Submit Button: Creates a new user account.

Login Page

- Login Form: Allows users to enter email and password for authentication.

Dashboard Page

- Quick Links: Access to joining classes, viewing enrolled classes, and materials.

- Recent Activity: Displays recent enrollments, class activity, and progress.

Join Class Page

- Class List: Displays available classes with title, description, price, and join button.

- Join Button: Registers the student to a selected class.

View Enrolled Classes Page

- Enrolled Classes Table: Shows all classes the student has joined, with enrollment status and date.

View Class Materials Page

- Materials Table/List: Lists resources for a selected class, with download/access buttons.

View Progress Page

- Progress Chart/Table: Displays completion percentages for each module or material in a class.

Contact Admin Button

- Contact Form: Allows students to send questions or feedback to the admin.

Logout Button

- Ends the session and redirects to the login page.

5.4 Report Formats

<Not Applicable>

6.0 Traceability Requirements Matrix

SRS Requirement	SDD_PKG_101	SDD_PKG_102	SDD_PKG_103	SDD_PKG_104	SDD_PKG_105	SDD_PKG_106	SDD_PKG_107	SDD_PKG_108	SDD_PKG_109	SDD_PKG_110	SDD_PKG_111	SDD_PKG_112	SDD_PKG_113	SDD_PKG_114	SDD_PKG_115	Domain	Data Access
SRS_REQ_1000	✓	✓													✓	User, UserController	UserDAO
SRS_REQ_1001		✓	✓	✓												EnrollmentControlle	EnrollmentDAO
SRS_REQ_1002			✓	✓	✓											ClassController, EnrollmentControll er	ClassDAO, EnrollmentDAO
SRS_REQ_2000				✓	✓								✓			MaterialController, ProgressController	MaterialDAO, ProgressDAO
SRS_REQ_2001					✓						✓					ProgressController	ProgressDAO
SRS_REQ_3000						✓										LogController	LogDAO
SRS_REQ_4000							✓									ClassController	ClassDAO
SRS_REQ_4001								✓								MaterialController	MaterialDAO
SRS_REQ_4002									✓			✓				InvoiceController	InvoiceDAO
SRS_REQ_4003										✓						EnrollmentControlle	EnrollmentDAO
SRS_REQ_5000											✓					ProgressController	ProgressDAO
SRS_REQ_6000														✓	✓	LogController	LogDAO

7.0 Resources Estimates

Operating Environment

Hardware

Table 7.1 Hardware Description

Item	**Description**
Central Processing Unit (CPU)	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Graphic Processing Unit (GPU)	Integrated Intel UHD Graphics
Solid State Drive (SSD)	KINGSTON SNV2S1000G
Random Access Memory (RAM)	12GB DDR4 (HyperX)
Power Supply	Standard laptop AC adapter, Output: 19V DC, 2.37A, 45W; Input: 100~240V AC 50/60Hz universal
Keyboard	PC/AT Enhanced PS/2 Keyboard (101/102-Key)
Mouse	Logitech wired mouse

Operating System

Table 7.2 Operation System Description

Item	**Description**
Window 10 and above	Version 2202 and above

Software

Item	Description
Visual Studio Code	Version 1.90
MySQL/phpMyAdmin	MySQL database with phpMyAdmin GUI
Internet Browser	Microsoft Edge / Google Chrome
Apache HTTP Server / Laragon	Local development environment
Laravel Framework	PHP backend framework
StarUML / Draw.io	Design and diagramming tools
Microsoft Word	For documentation
WhatsApp Messenger	For communication
GitHub / Git	Version control
Antivirus/Firewall	McAfee

Table 7.3 Software Description

Design and Implementation Constraints

Types of Constraints	Constraints
Network	- The user access to SLACES may be compromised due to irregular internet connection and delay in enrollment or downloading of material.
	- The bandwidth, particularly when uploading files of large size (PDFs, videos), could be a problem in the effective sharing of material.
Software	- Data format inconsistencies (PDF, video) or integration with external APIs (e.g., WhatsApp) may require specialized handling or skillsets.
	- The accessibility can be affected by compatibility problems between various operating systems (Windows, MacOS, mobile browsers).
	- Potential recurring license costs for third-party tools (e.g., Laravel, design tools, antivirus).
Hardware	- Limited device processing power or memory, especially on older laptops or mobile devices, can affect performance and responsiveness.
	- Outdated or restricted input/output devices (e.g., keyboard, mouse, screen size) may reduce usability for some users.
	- Devices with short battery life or overheating may require additional power management or cooling solutions for extended use.
Tools and Database	- Complex queries or advanced reporting needs may demand further optimization or migration to more powerful database solutions.
	- Current database or local server configuration could have scalability constraints that will limit expansion or growth of users in the future.

Computer Resources Required for Operating the Software

Category	Minimum Requirements	Recommended Requirements
Server Hardware		
Processor	Intel Core i5-1035G1 or AMD Athlon Silver 3050U	Intel Core i7 or AMD Ryzen 7, 2.5 GHz or higher
Memory (RAM)	8 GB	16 GB or higher
Storage	250 GB SSD	1 TB SSD, RAID configuration for redundancy
Network	100 Mbps Ethernet	1 Gbps Ethernet or higher
Client Hardware		
Processor	Intel Core i3 or equivalent	Intel Core i5 or higher
Memory (RAM)	4 GB	8 GB or higher
Storage	100 GB HDD or SSD	256 GB SSD
Display	1366 x 768 resolution	1920 x 1080 resolution or higher
Network	10 Mbps internet connection	100 Mbps or higher for seamless operation
Server Software		
Operating System	Windows Server 2016 or Ubuntu 18.04 LTS	Windows Server 2019 / Ubuntu 22.04 LTS
Database Management	MySQL 5.7	MySQL 8.0
Web Server	Apache 2.4	Apache 2.4 or higher
Application Server	Apache 2.4	Apache 2.4 or higher
Client Software		
Operating System	Windows 10 or macOS Mojave	Windows 10 (latest updates), macOS Catalina or higher
Web Browser	Google Chrome 79, Firefox 72	Latest Chrome, Firefox, or Microsoft Edge
Other Software	PDF Reader (e.g., Adobe Acrobat Reader)	Office Suite (e.g., Microsoft Office 2016 or later)
Network Requirements		
Internal Network	100 Mbps LAN for internal communications	1 Gbps LAN for high-speed data transfer
Internet Connection	10 Mbps for basic operations	100 Mbps or higher for optimal performance, especially for remote access and cloud-based operations

8.0 Appendices

Activity Diagram

Business activity diagram

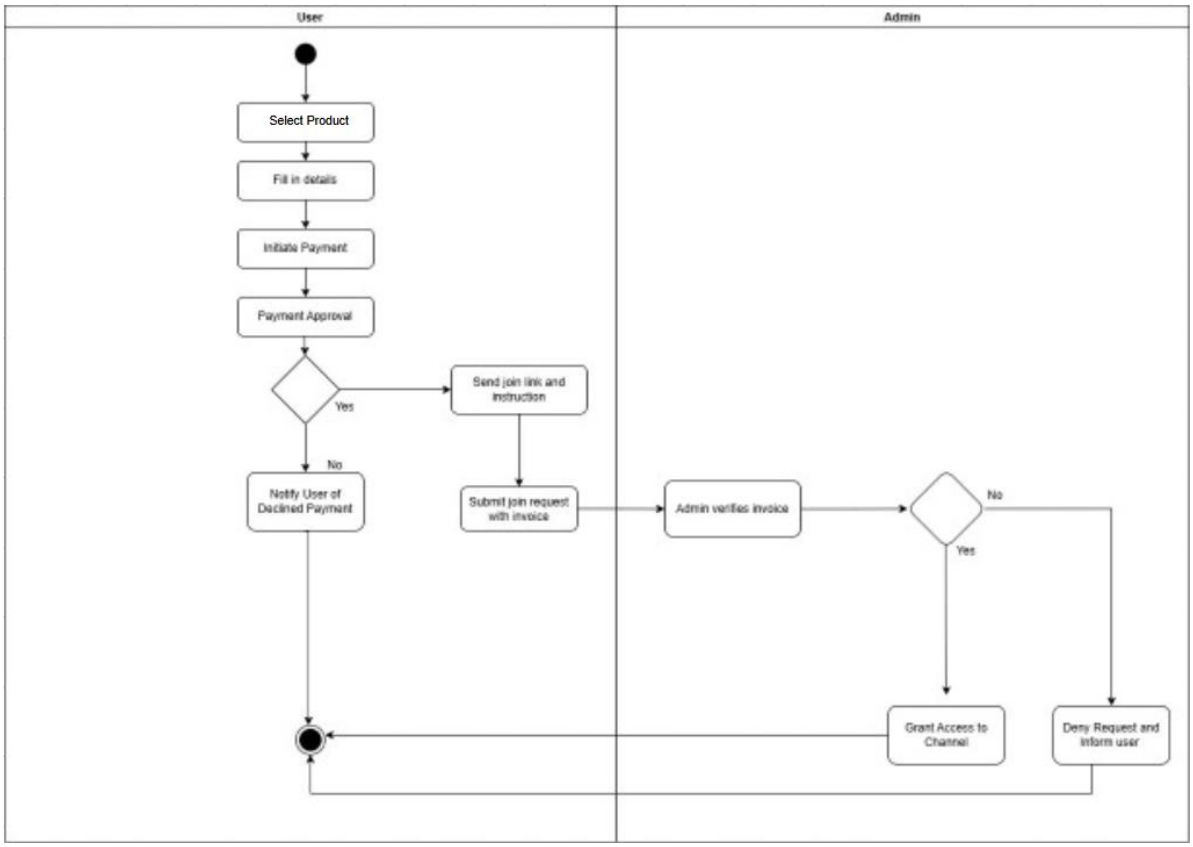

Figure B.1 Business Activity Diagram

Login

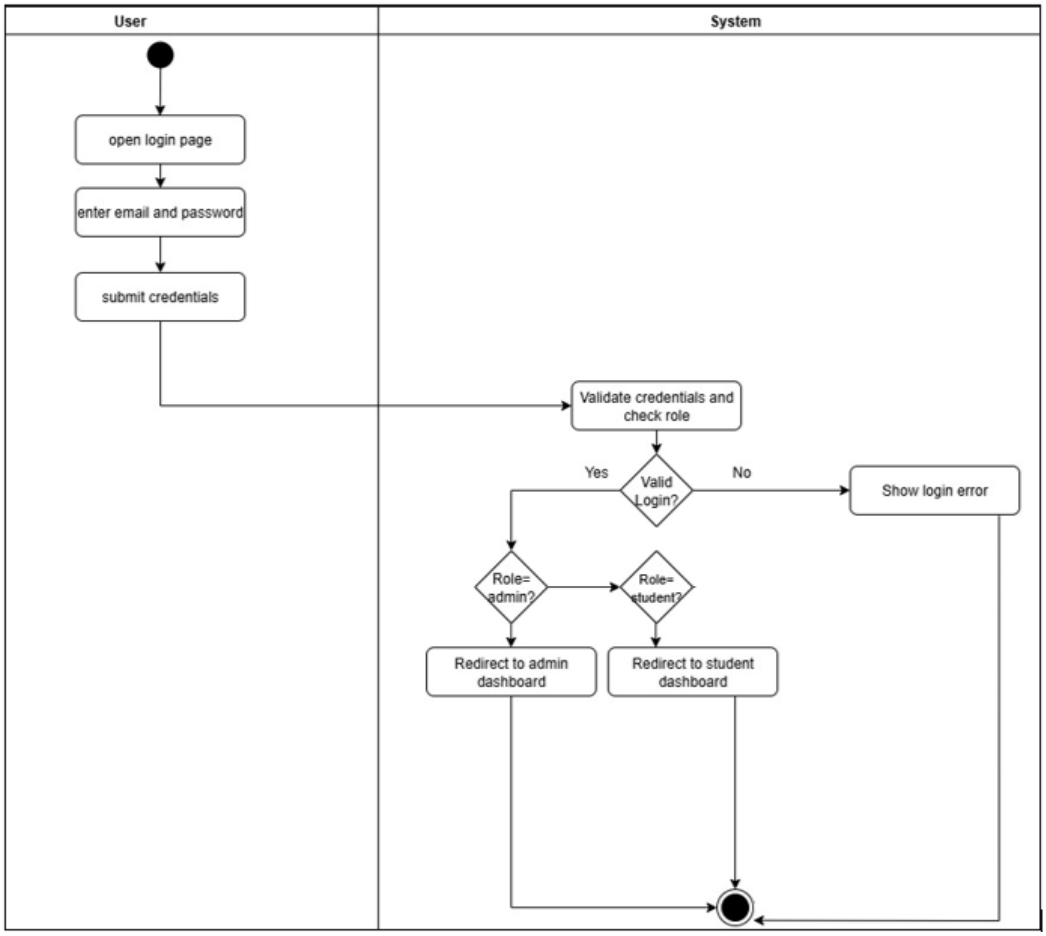

Figure B.2 Activity Diagram Login

Join New Class

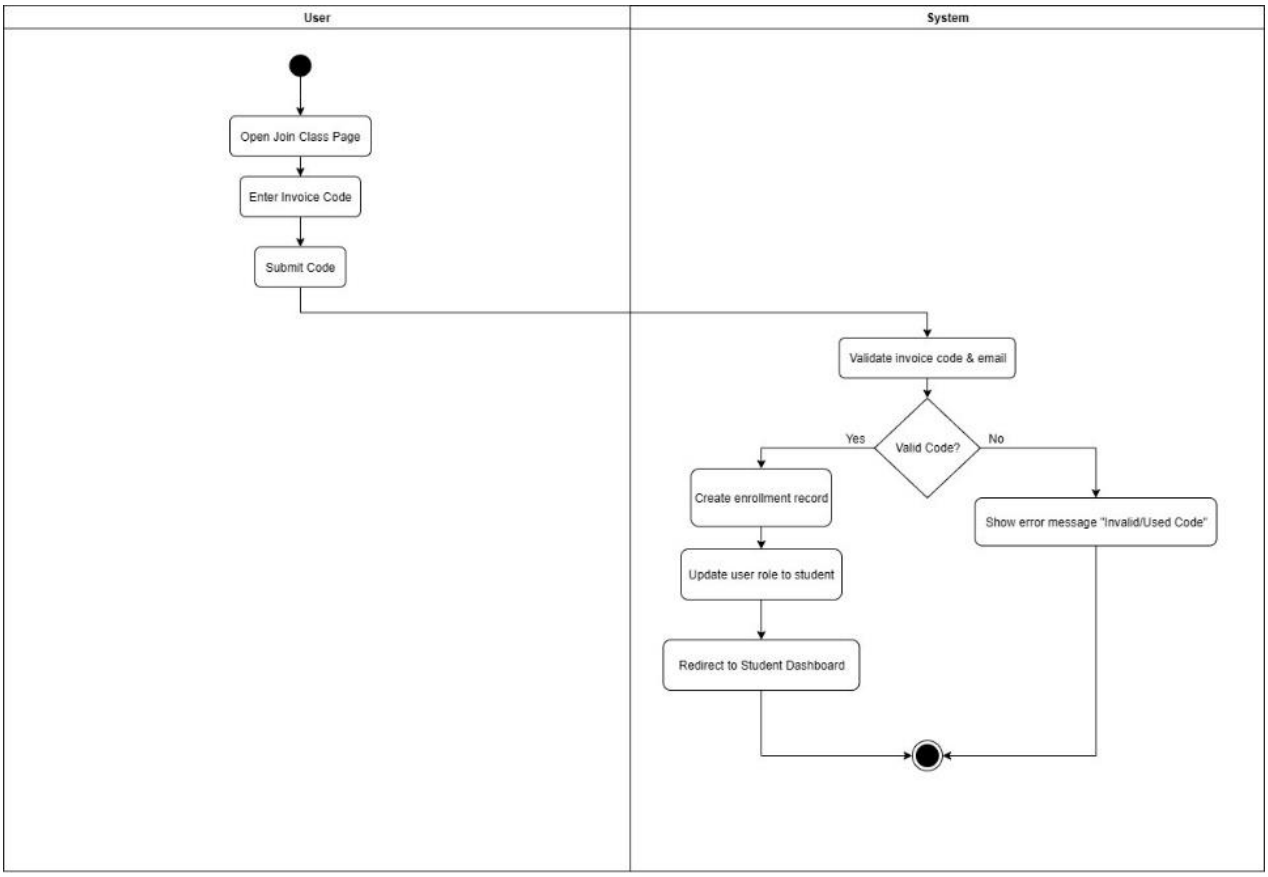

Figure B.3 Activity Diagram Join New Class

View My Classes

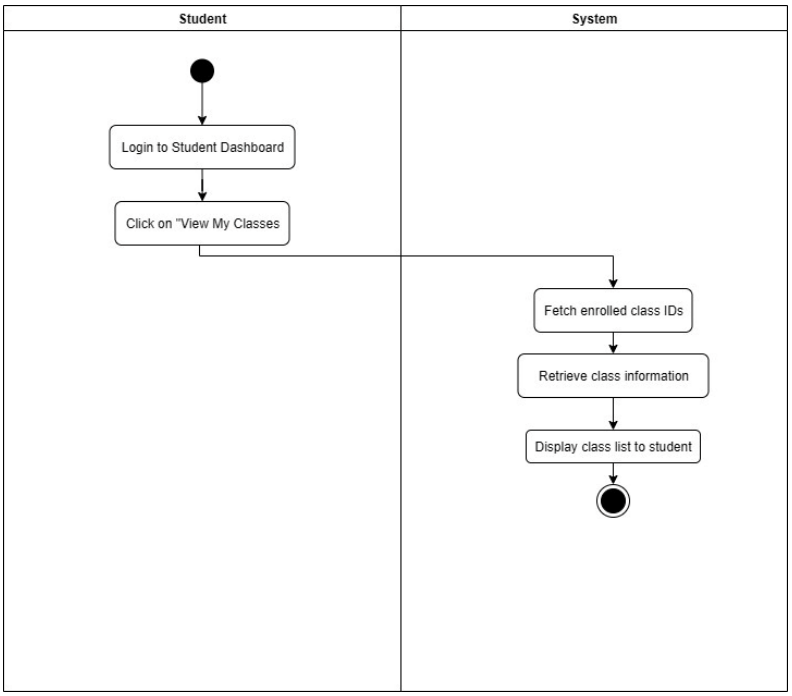

Figure B.4 Activity Diagram View My Classes

Access Class Materials

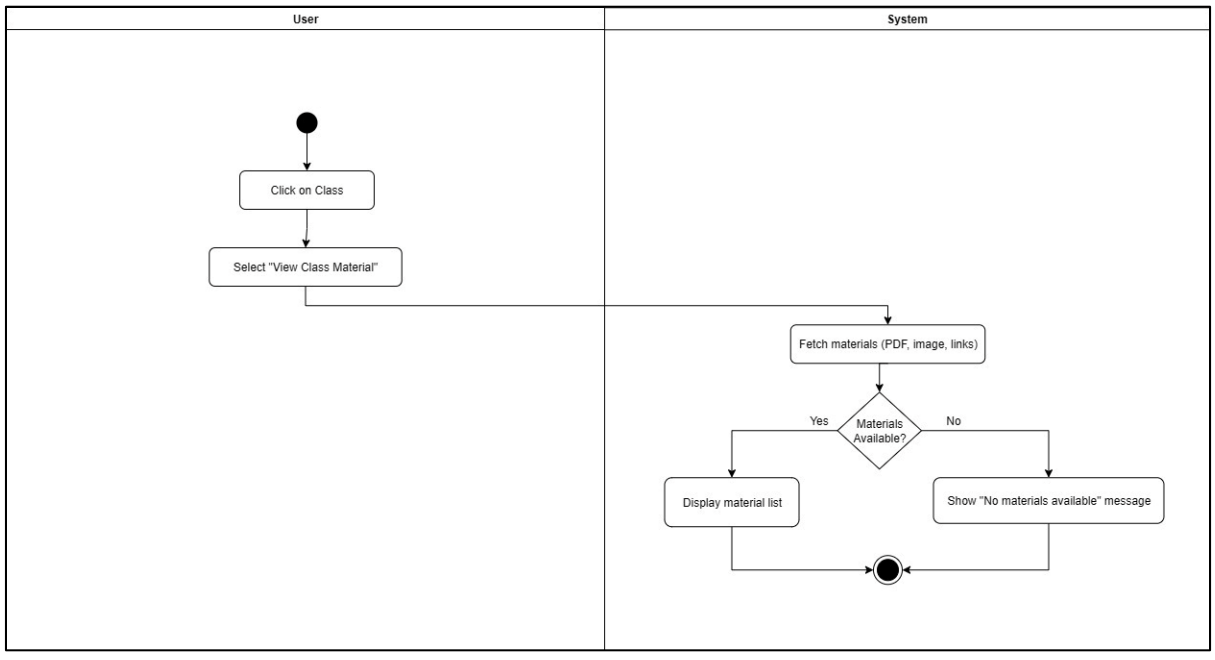

Figure B.5 Activity Diagram Access Class Materials

Contact Admin

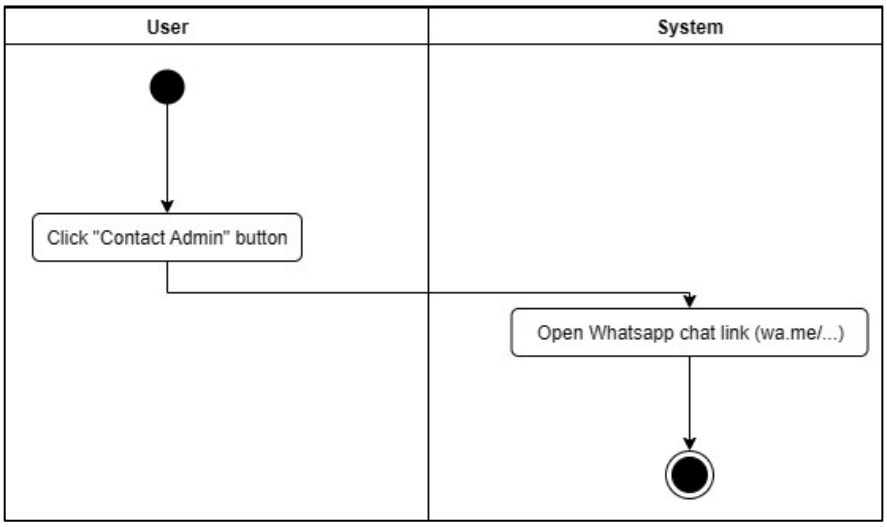

Figure B.6 Activity Diagram Access Contact Admin

Manage Classes

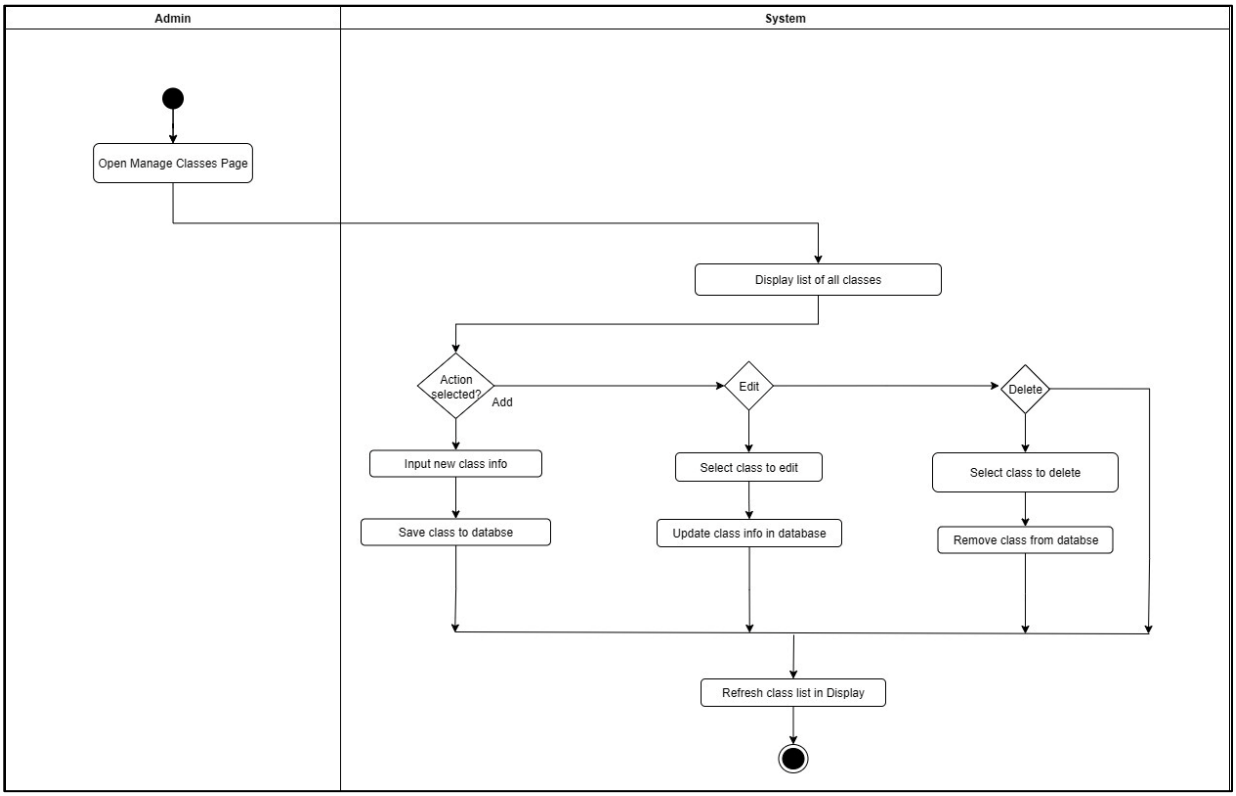

Figure B.7 Activity Diagram Manage Classes

Manage Class Materials

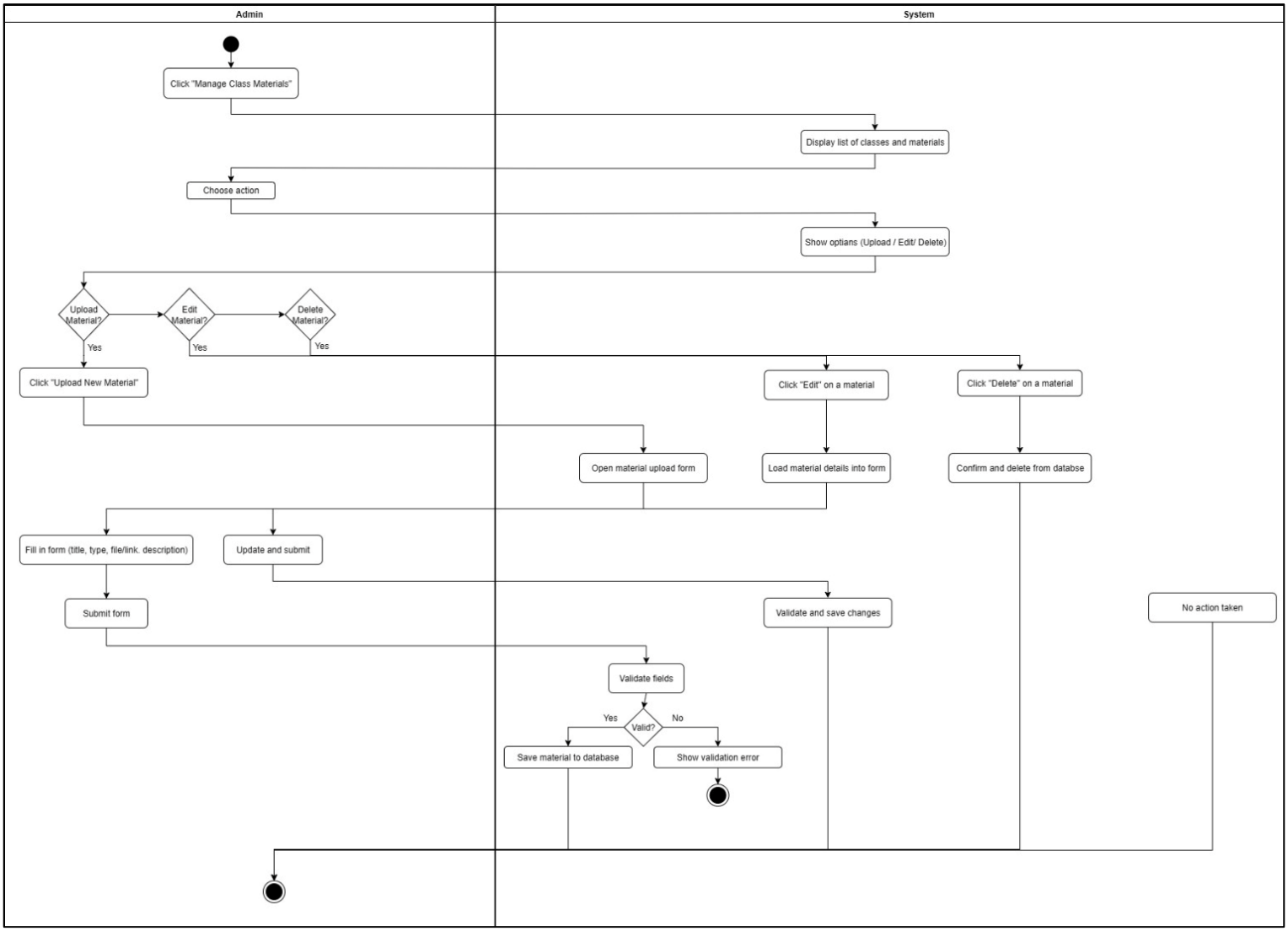

Figure B.8 Activity Diagram Manage Class Materials

Manage Invoices

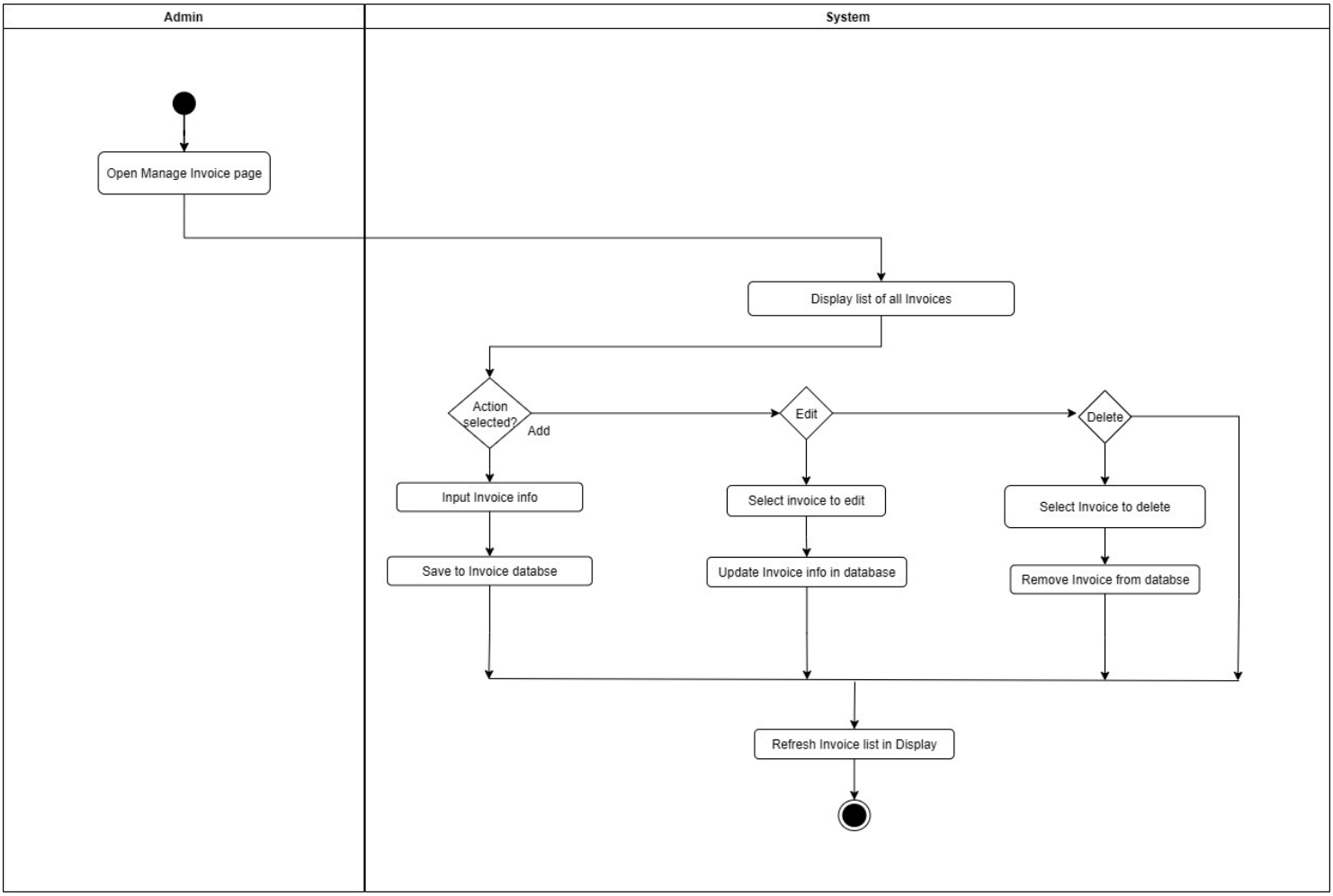

Figure B.9 Activity Diagram Manage Invoice

View Enrolled Students

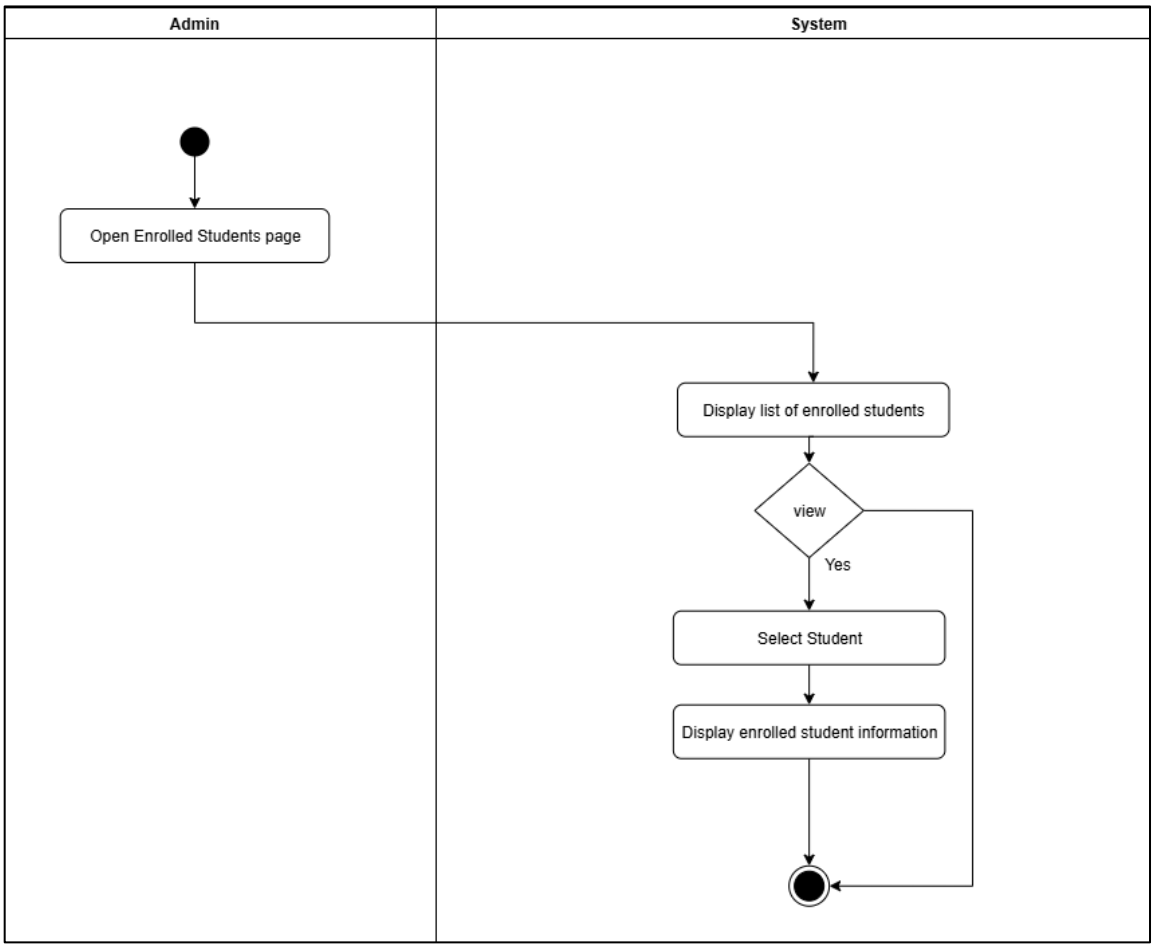

Figure B.10 Activity Diagram View Enrolled Students

Domain Class Diagram

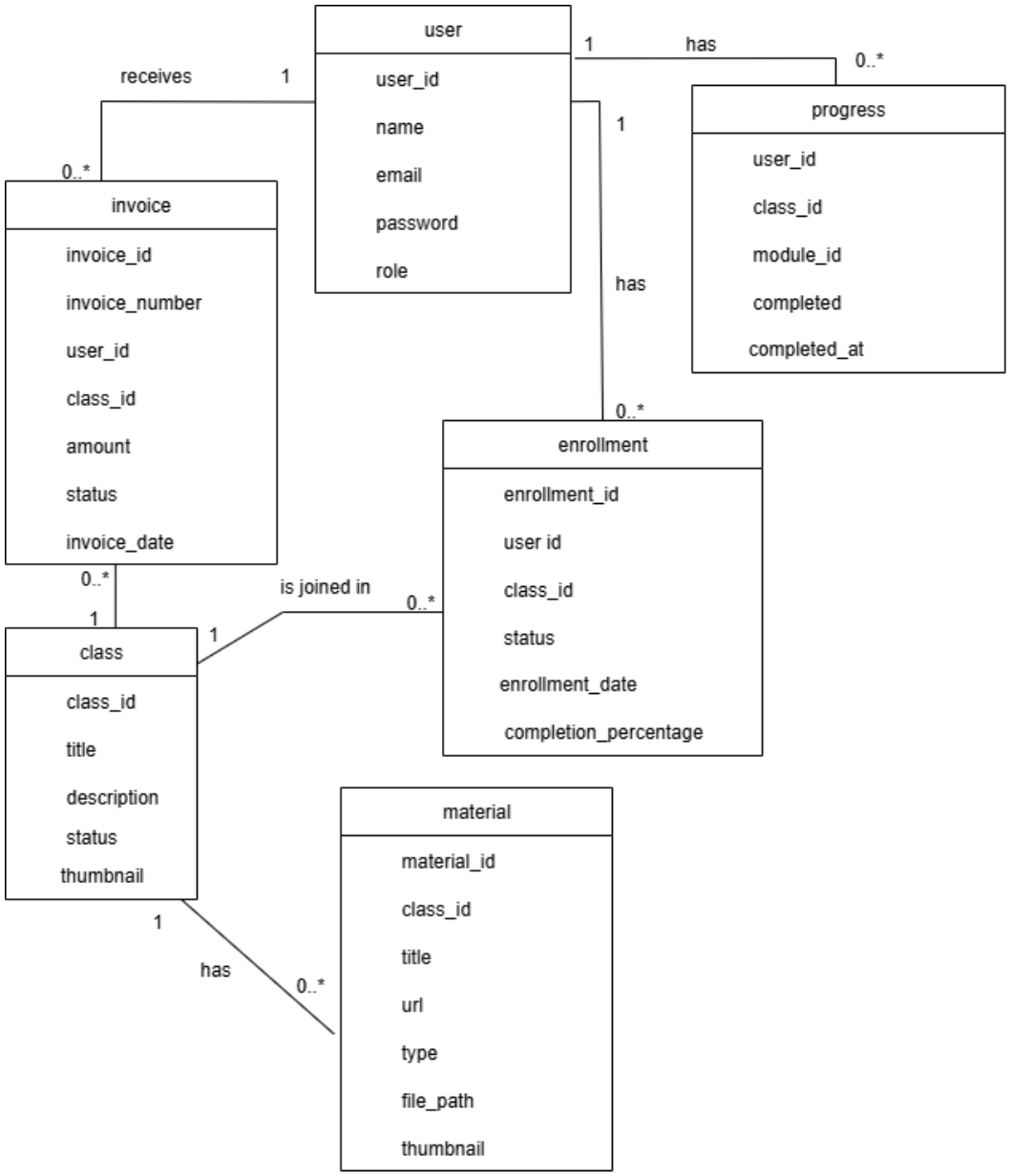

Figure B.11 Domain Class Diagram

Sequence Diagram

Login

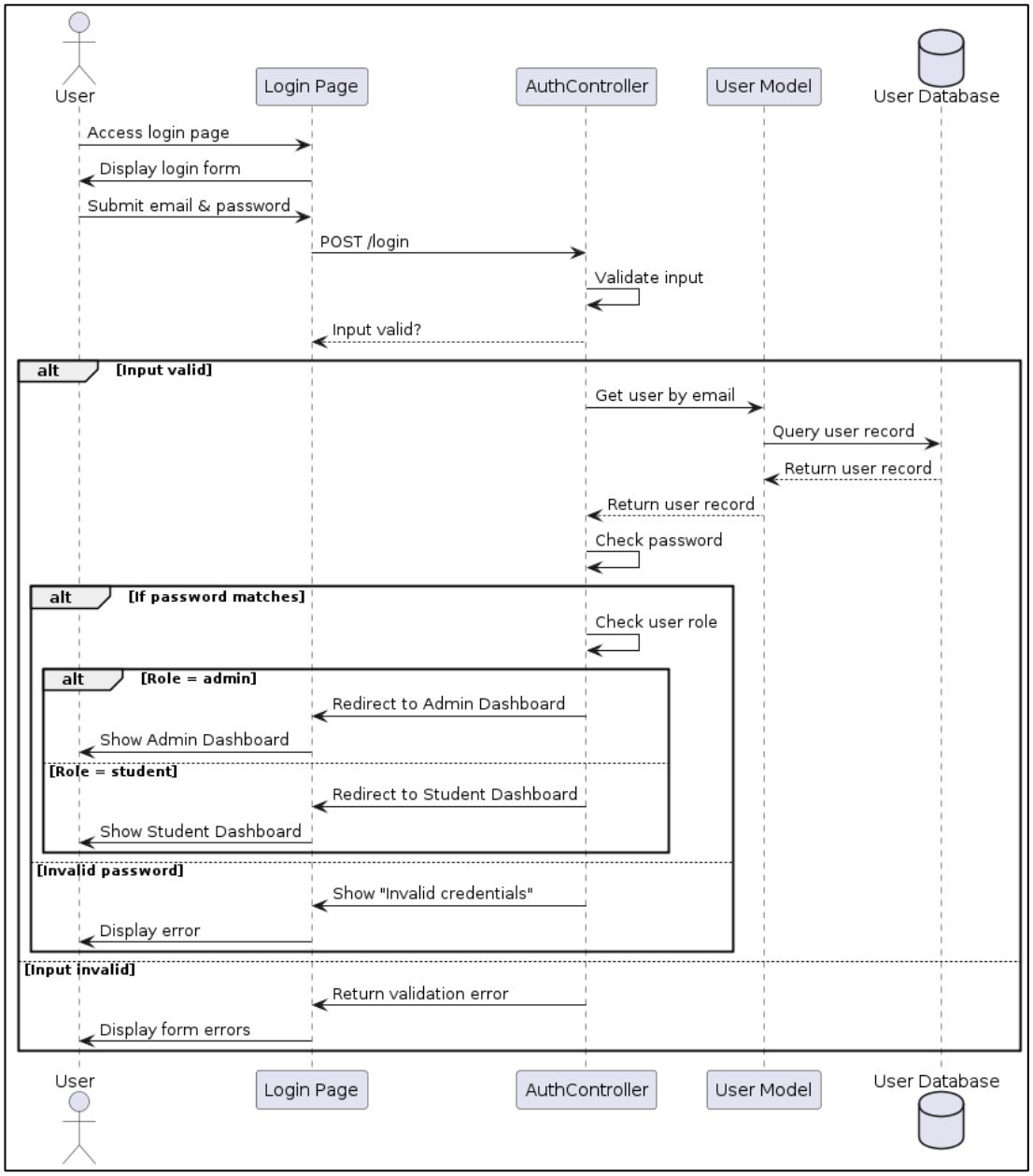

Figure B.12 Sequence Diagram Login

Join New Class

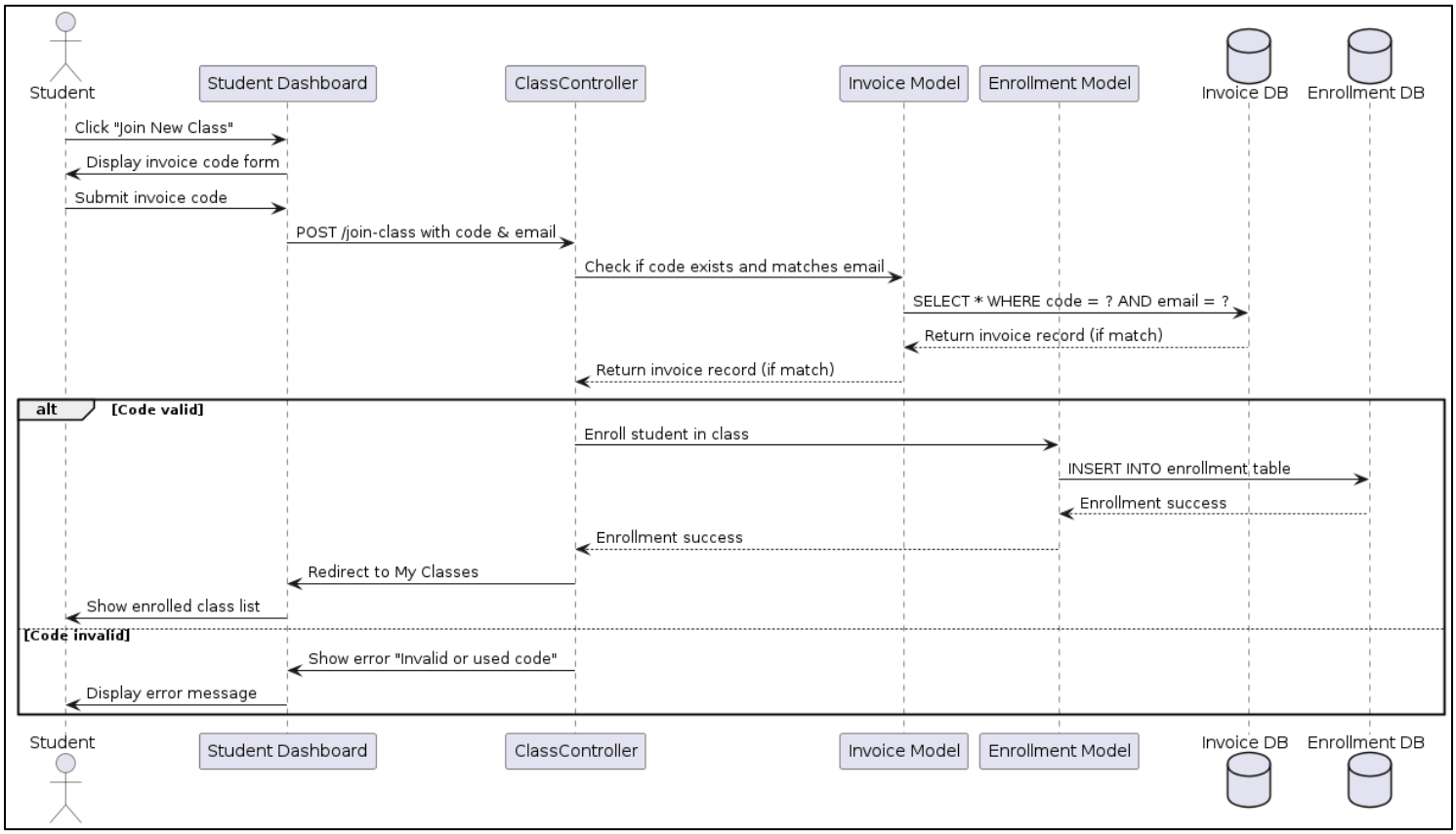

Figure B.13 Sequence Diagram Join New Class

View My Classes

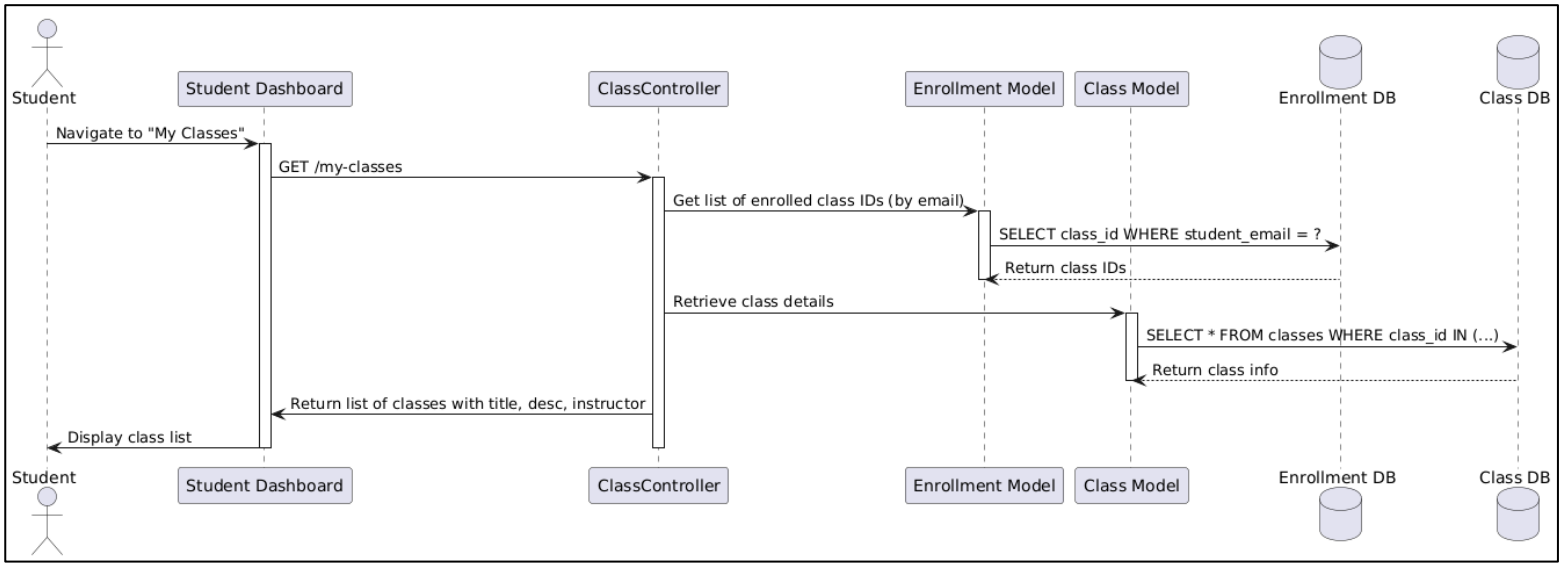

Figure B.14 Sequence Diagram View My Classes

Access Class Materials

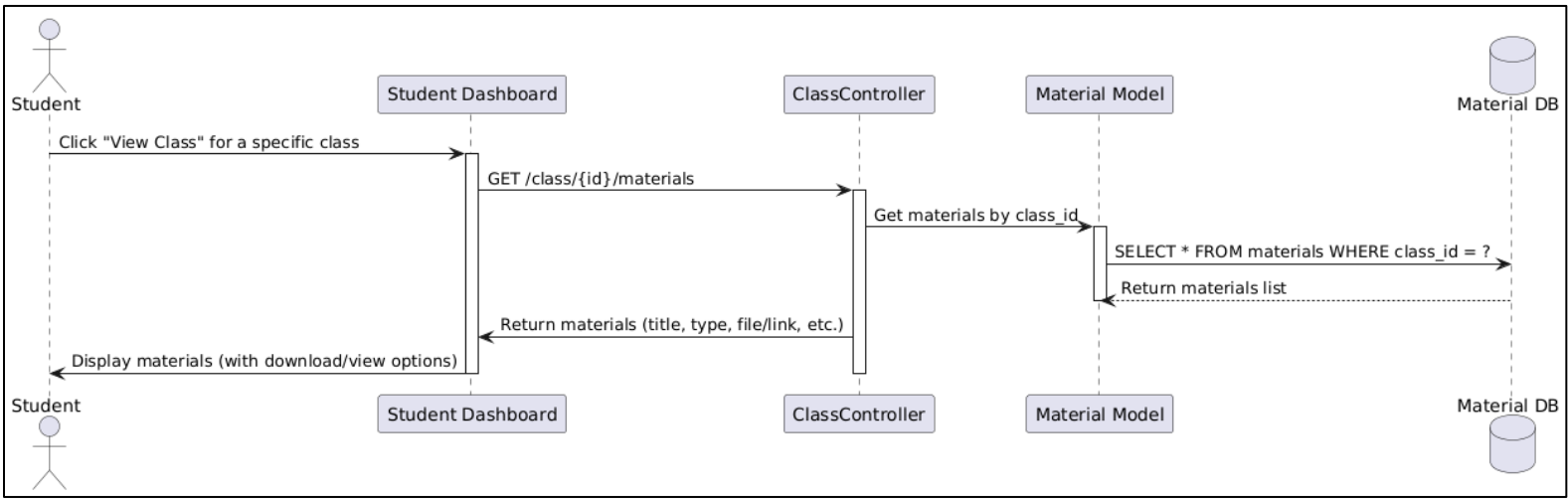

Figure B.15 Sequence Diagram Access Class Materials

Contact Admin

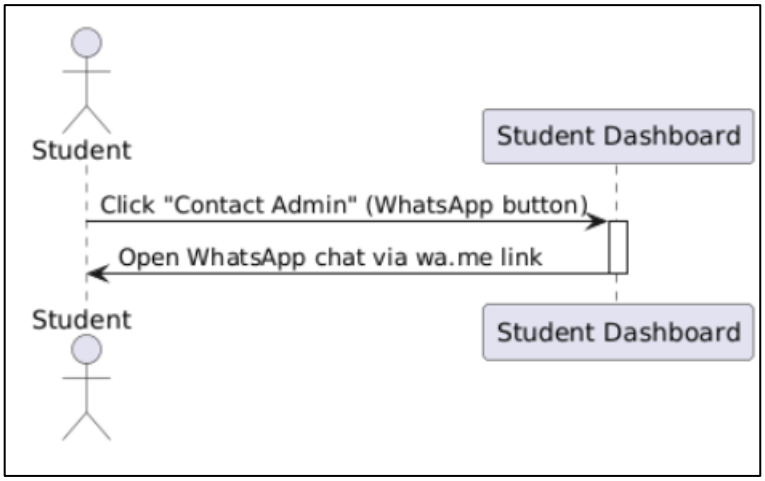

Figure B.16 Sequence Diagram Access Contact Admin

Manage Invoices

Figure B.17 Sequence Diagram Manage Invoice

Manage Classes

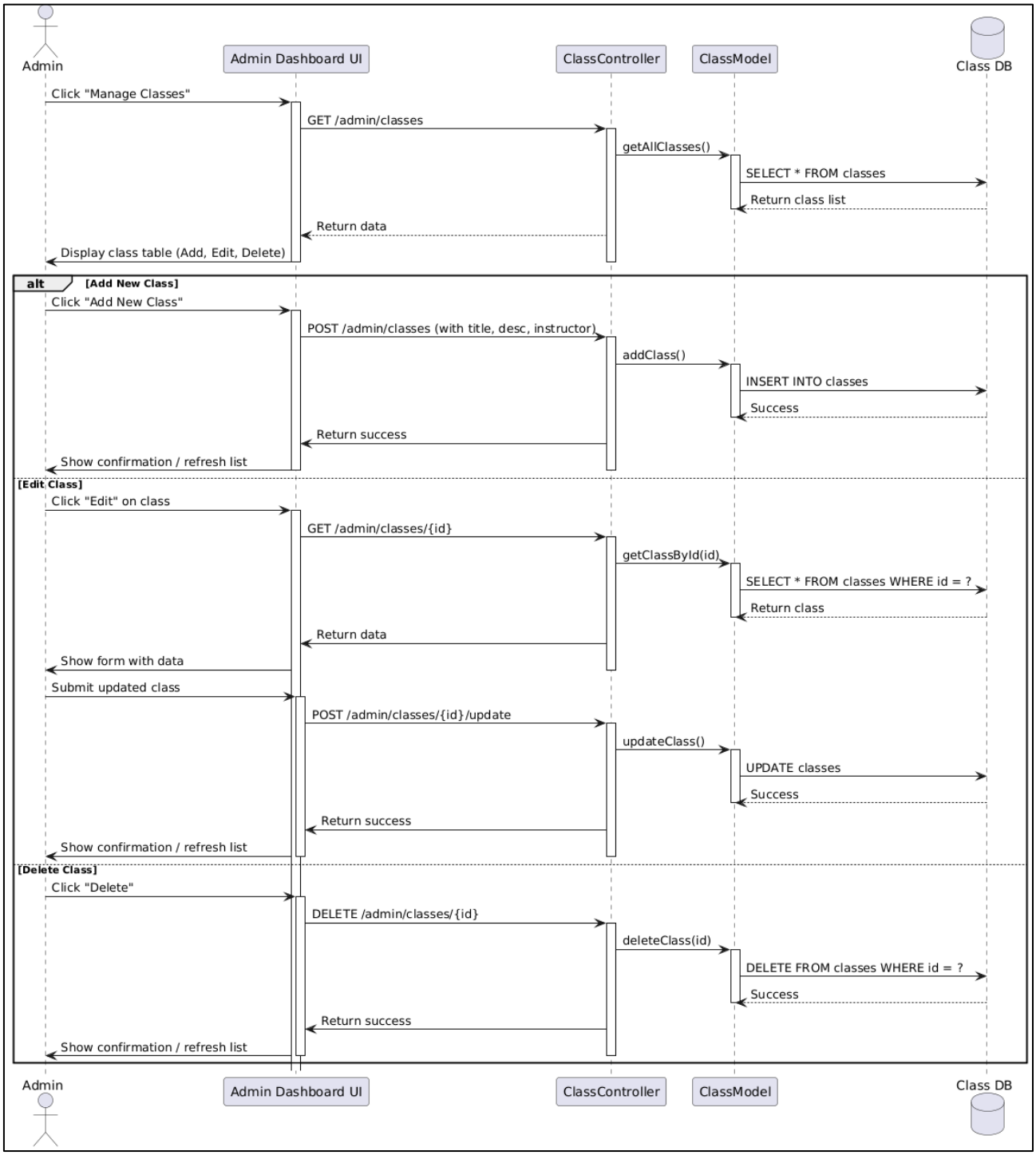

Figure B.18 Sequence Diagram Manage Classes

Manage Class Materials

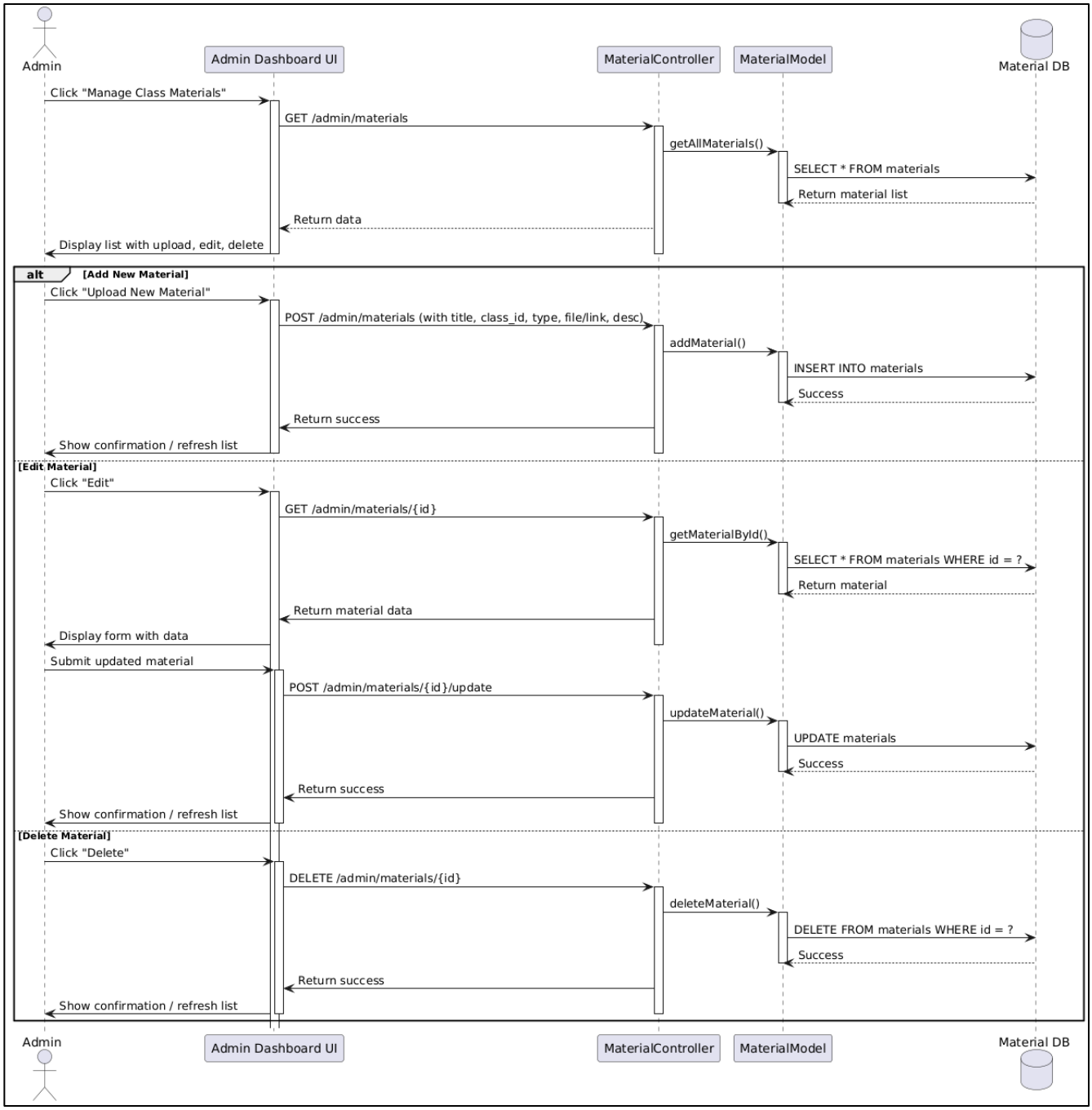

Figure B.19 Sequence Diagram Manage Class Materials

View Enrolled Students

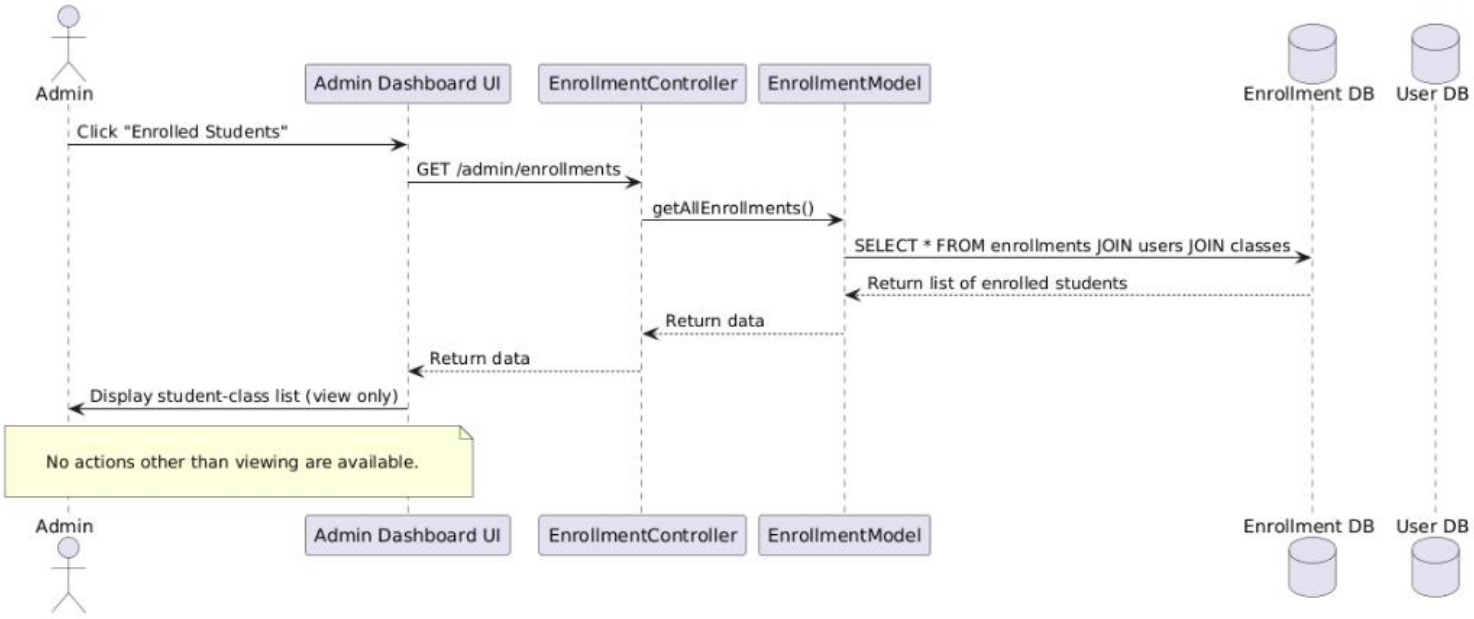

Figure B.20 Activity Diagram View Enrolled Students

Entity Relationship Diagram

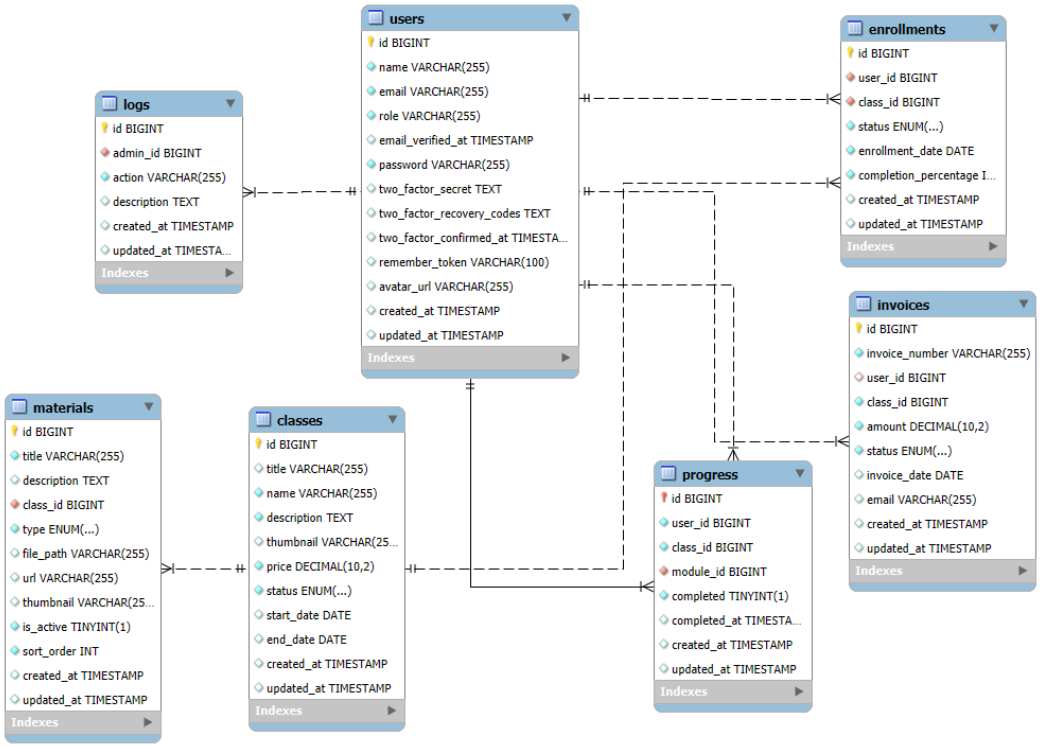

Figure B.21 Entity Relationship Diagram