Software Requirements Specification

for

<Online Course Management System>

**Version 2.0 approved**

**Prepared by <Tiến Mạnh, Quốc Huy, Đăng Sỹ , Minh Tuấn>**

**<Group 2>**

**<date created: 13/10/2024>**

*https://docs.google.com/document/d/10KU5oBjuq8lkHR55pohIDtiGyJvbRyt8fVZJWXljgkI/edit?tab=t.0*

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1.** **Introduction 2**

1.1 Purpose 2

1.2 Document Conventions 2

1.3 Project Scope 2

1.4 References 2

**2.** **Overall Description 2**

2.1 Product Perspective 2

2.2 User Classes and Characteristics 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 2

2.5 Assumptions and Dependencies 2

**3.** **System Features 2**

3.1 System Feature 1 2

3.2 System Feature 2 (and so on) 2

**4.** **Data Requirements 2**

4.1 Logical Data Model 2

4.2 Data Dictionary 2

4.3 Reports 2

4.4 Data Acquisition, Integrity, Retention, and Disposal 2

**5.** **External Interface Requirements 2**

5.1 User Interfaces 2

5.2 Software Interfaces 2

5.3 Hardware Interfaces 2

5.4 Communications Interfaces 2

**6.** **Quality Attributes 2**

6.1 Usability 2

6.2 Performance 2

6.3 Security 2

6.4 Safety 2

6.5 [Others as relevant] 2

**7.** **Internationalization and Localization Requirements 2**

**8.** **Other Requirements 2**

**Appendix A: Glossary 2**

**Appendix B: Analysis Models 2**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

This Software Requirements Specification (SRS) document details the **Online Course Management System (OCMS)**. The system is designed to help universities manage courses, student enrolment, and track academic progress. It describes the features and functionalities for students, instructors, and administrators, serving as a guide for developing and maintaining the system.

## Purpose

This document specifies the software requirements for the **Online Course Management System (OCMS)**. It covers all the functionalities and features needed for version 1.0 of the system. The intended audience for this document includes:

* Developers: To guide the implementation of the system’s features.
* Project Managers: To understand project scope and manage progress.
* Testers: To create test cases and ensure the system meets all requirements.
* Users (Students, Instructors, Administrators): To understand system capabilities and interactions.
* Documentation Writers: To provide user manuals and help guides.

## Document Conventions

## Font family: Times New Roman,

## Font size: 13.

## Font weight heading: bold.

## Font size heading 1: 18.

## Font size heading 2: 14.

## The functional requirements for this project are organized by use case within the user class.

## Project Scope

The **Online Course Management System (OCMS)** helps universities manage courses, enrollment, and academic progress. It allows instructors to create course materials, students to enroll and track their progress, and administrators to manage courses and users. The system aims to streamline academic processes, supporting efficient online learning and improving operational effectiveness.

## References

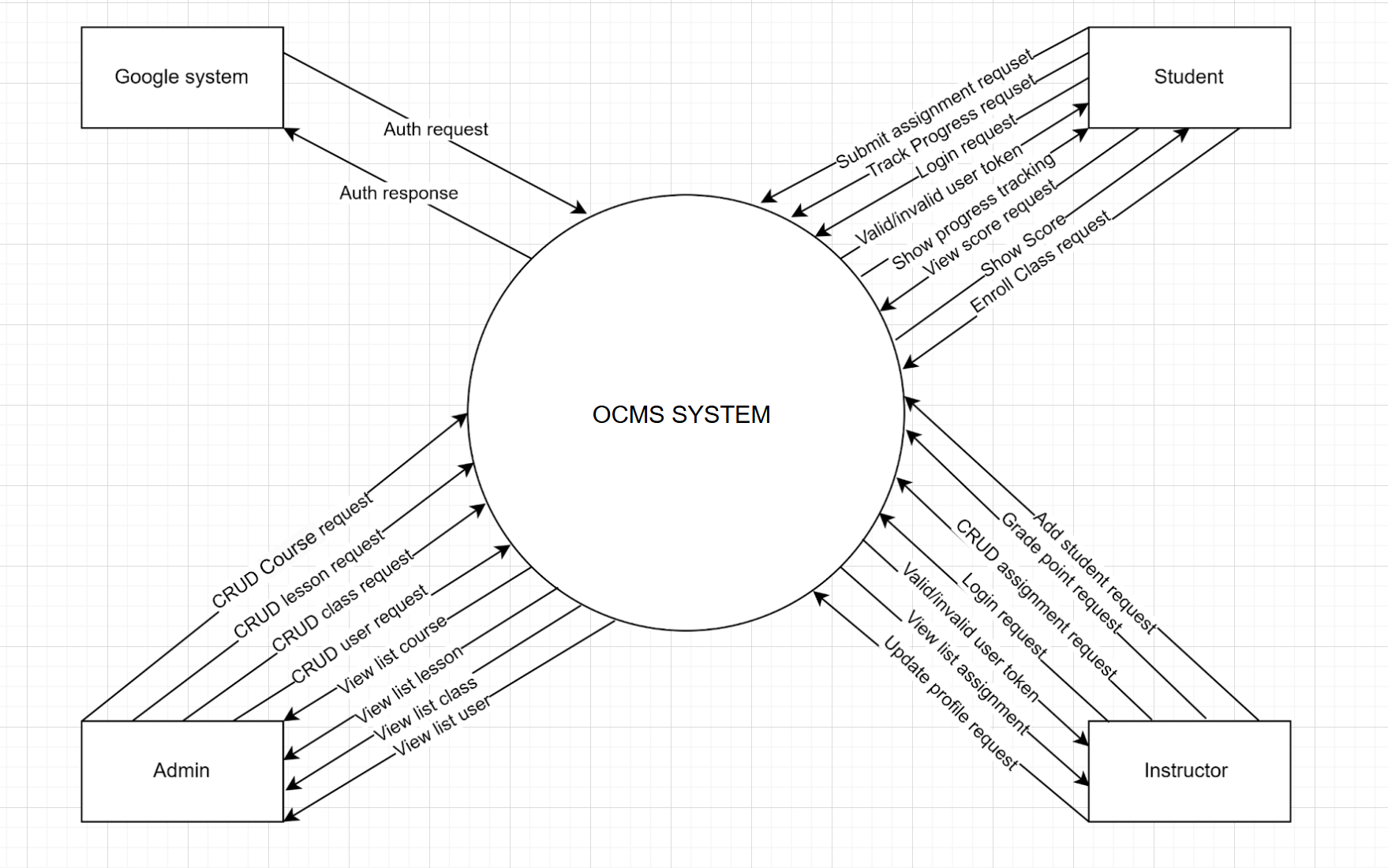
## Software requirement specification form

## SRS of Cafeteria Ordering System (by Karl Wiegers)

# Overall Description

The **Online Course Management System (OCMS)** helps universities manage courses, student enrollment, and track academic progress. Users include students (enroll in courses, submit assignments, track grades), instructors (create content, grade), and administrators (manage courses, users, and generate reports). This web-based system operates online, is compatible with various devices, and must comply with security and legal standards.

## Product Perspective



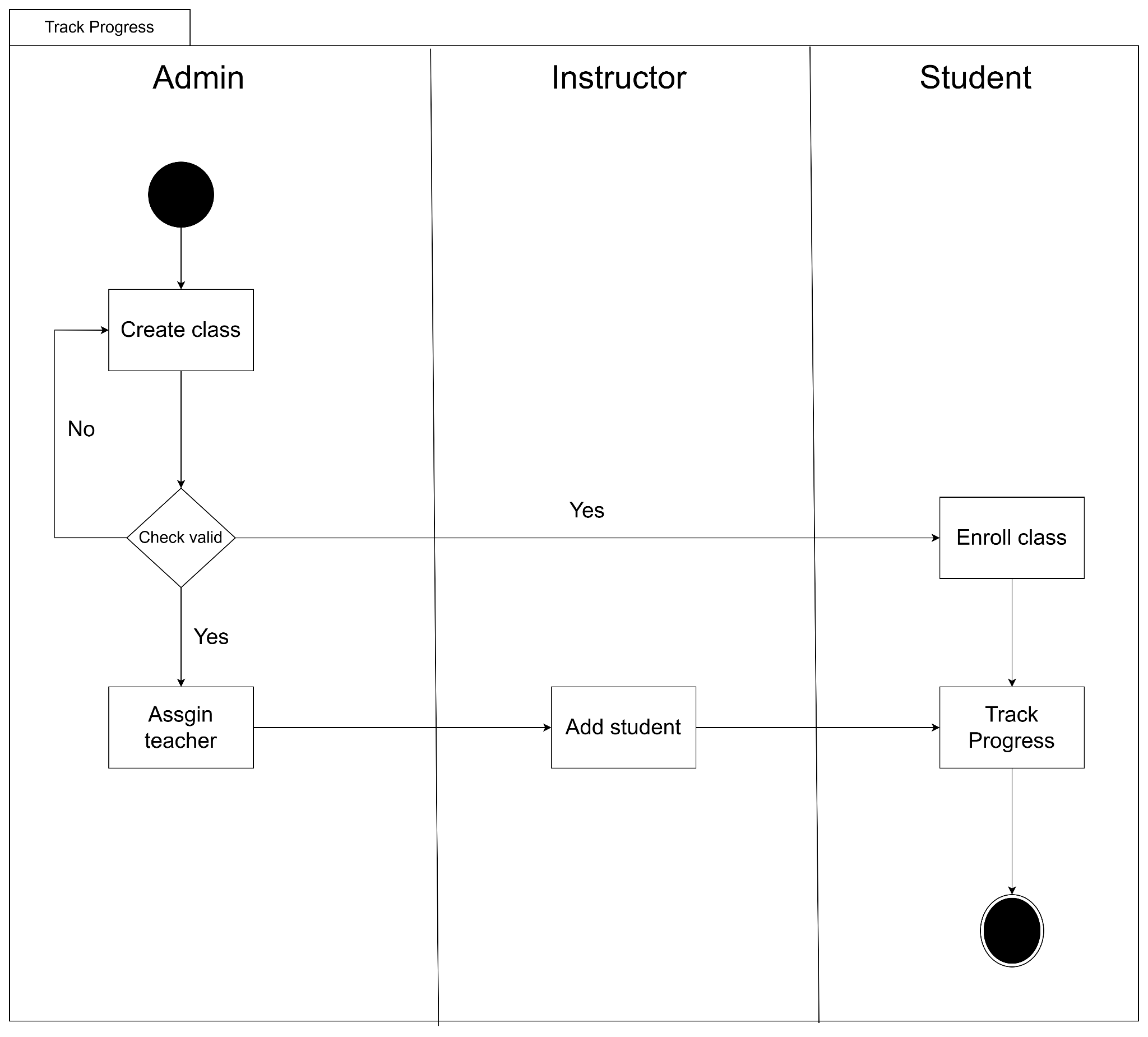
## User Classes and Characteristics

### 

|  |  |  |
| --- | --- | --- |
| **#** | **Actor** | **Description** |
| 01 | Administrator | The administrator manages the overall system. They have permissions to perform CRUD (Create, Read, Update, Delete) operations on courses, lessons, classes, and user accounts. Additionally, they can view lists of courses, lessons, classes, and users to oversee the system's operations and maintain the database. |
| 02 | Instructor | The instructor is responsible for managing classes and assignments. They can perform actions such as adding students, grading assignments, managing assignment content (CRUD operations), and tracking student progress. Instructors can also view and update their profile information. |
| 03 | Student | The student interacts with the system to participate in classes, view assignments, and submit their work. They can track their progress, view scores, enroll in classes, and perform assignment submissions. Additionally, they can log in to the system and view their progress and grades. |
| 04 | Google System | This external system is used for authentication purposes. It provides login and authentication services by validating user credentials via OAuth2 authorization protocols. |

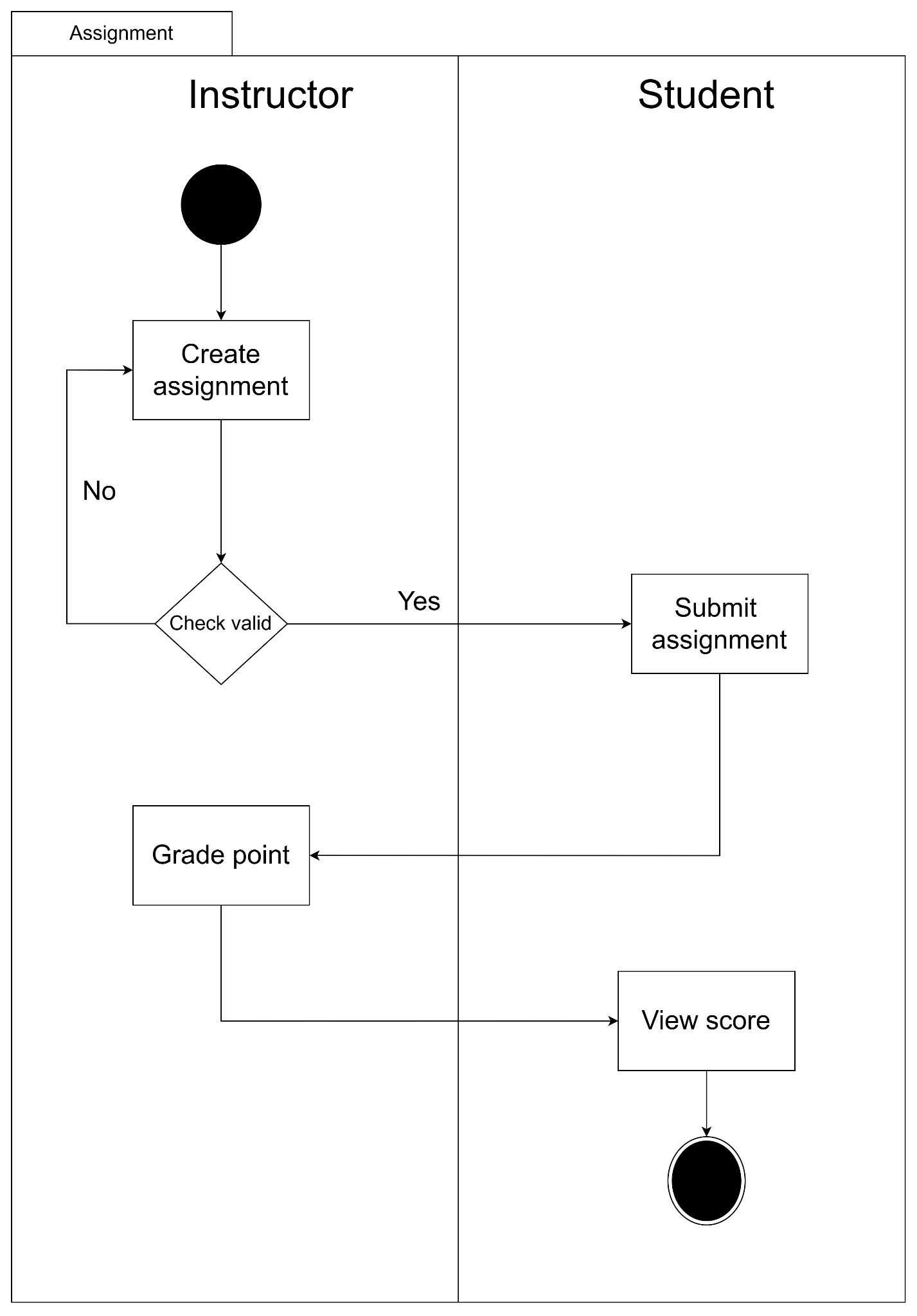
## Main Workflows

### 2.3.1 Tracking Student Progress



### 2.3.2 Students Do Assignment

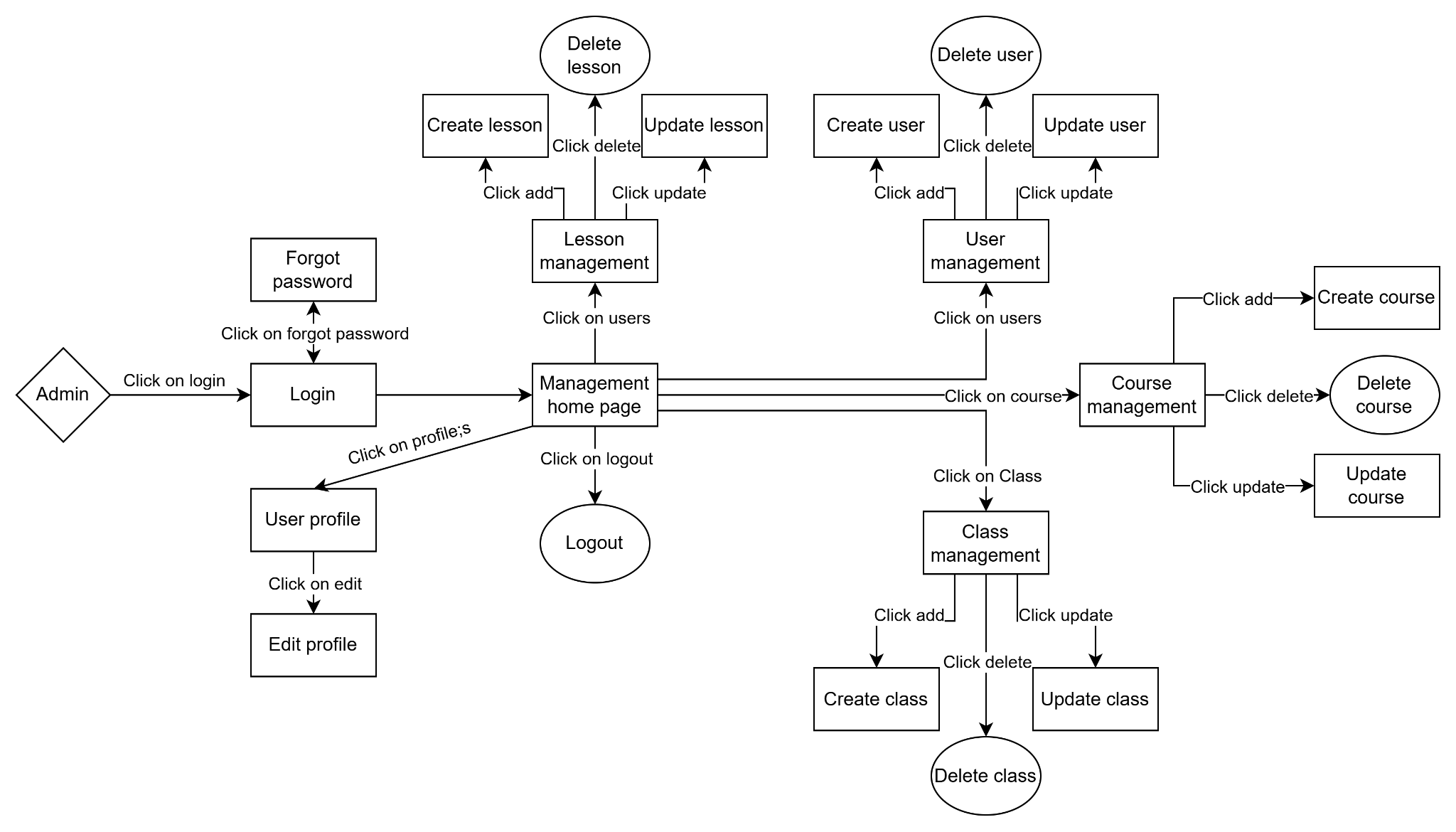
### 



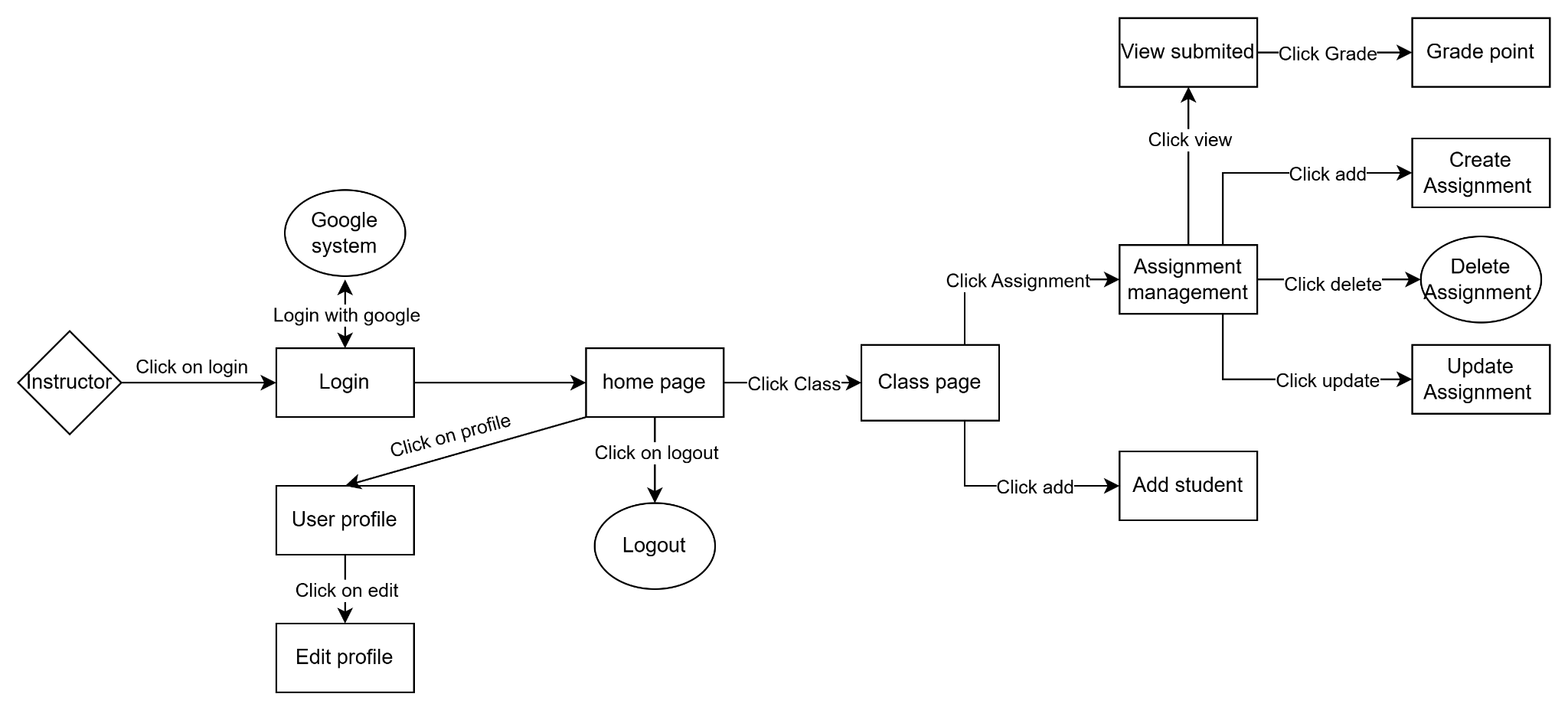
## System Functionalities

### 2.4. Screens Flow

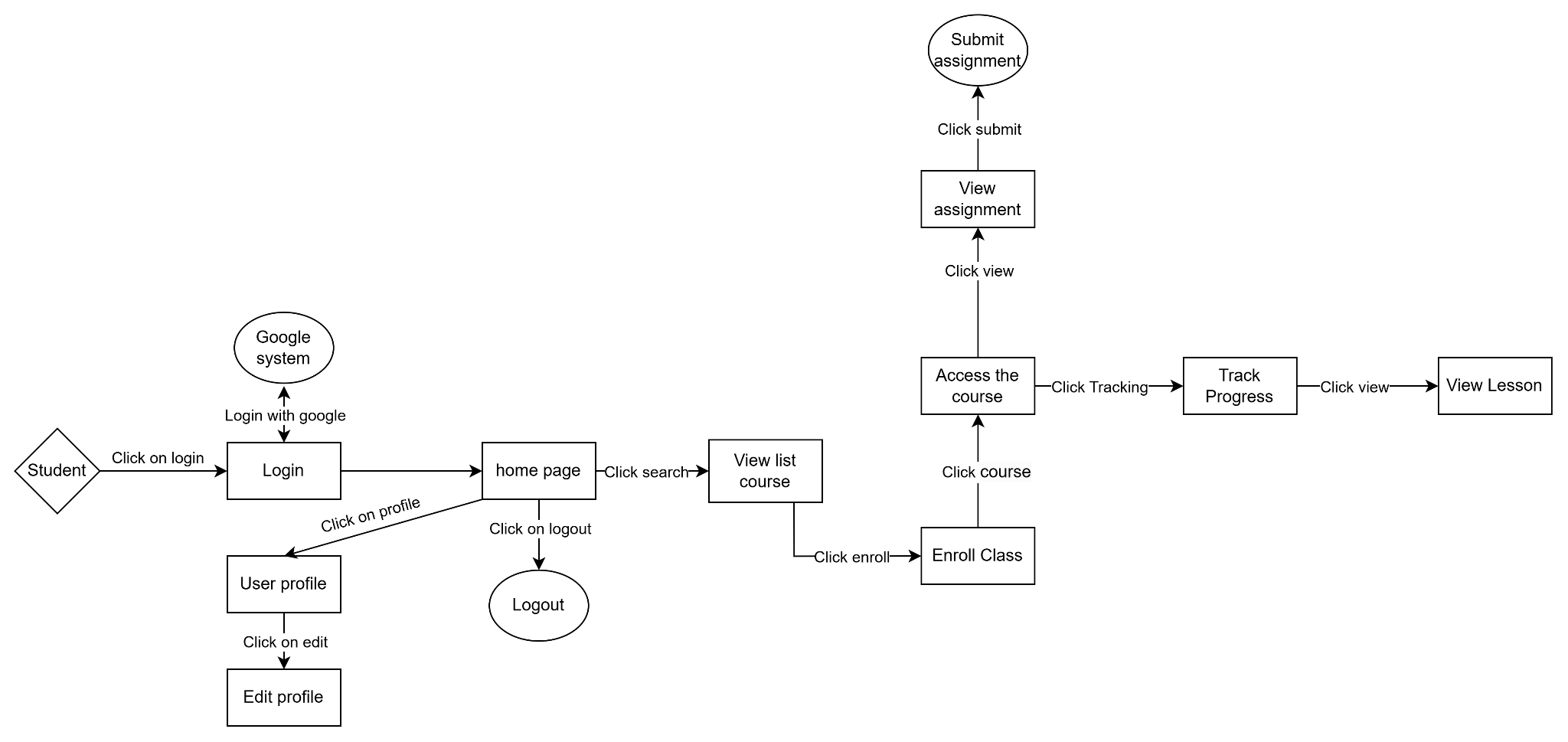
### 2.4.1 Screens Flow for admin



### 2.4.2 Screens Flow for instructor



### 2.4.3 Screens Flow for student



### 2.4.4 Screen Authorization

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen** | **Admin** | **Instructor** | **Student** |
| Login | x | x | x |
| Home Page | x | x | x |
| User Profile | x | x | x |
| Logout | x | x | x |
| Course Management | x |  |  |
| Class Management | x |  |  |
| Lesson Management | x |  |  |
| Assignment Management |  | x |  |
| Add Student to Class |  | x |  |
| Submit Assignment |  |  | x |
| View Assignment |  |  | x |
| Track Progress |  |  | x |
| Enroll Class |  |  | x |

### 3.3 Non-UI Functions

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Feature** | **System Function** | **Description** |
| 1 | Login with google | Login | Students and instructors can login with google. |

## Operating Environment

The Online Course Management System (OCMS) will operate in the following environment:

* **Hardware Platform**: The system will run on standard personal computers or university servers, supporting basic hardware requirements.
* **Operating Systems**: The platform will be developed using Java with JSP and Servlets, and can run on operating systems like Windows, macOS, and Linux.
* **Geographical Locations**: Users will primarily be students and instructors from local or university networks.
* **Database and Hosting**: The system will use a simple relational database (e.g., MySQL) hosted locally or on a university-managed server.
* **Software Components**: The system will coexist with existing university applications such as email services for notifications and simple authentication systems.

## Design and Implementation Constraints

**Corporate or Regulatory Policies**: The project should follow the university’s guidelines for data management and user privacy.

**Hardware Limitations**: The system should be lightweight, considering limited hardware resources available for student projects.

**Technology Stack**: The system will be built using Java, JSP, Servlets, and a simple database like MySQL.

**Database Constraints**: The relational database will be used to store course and user data with basic normalization.

**Security**: Basic security measures like password hashing and session management will be implemented.

## Assumptions and Dependencies

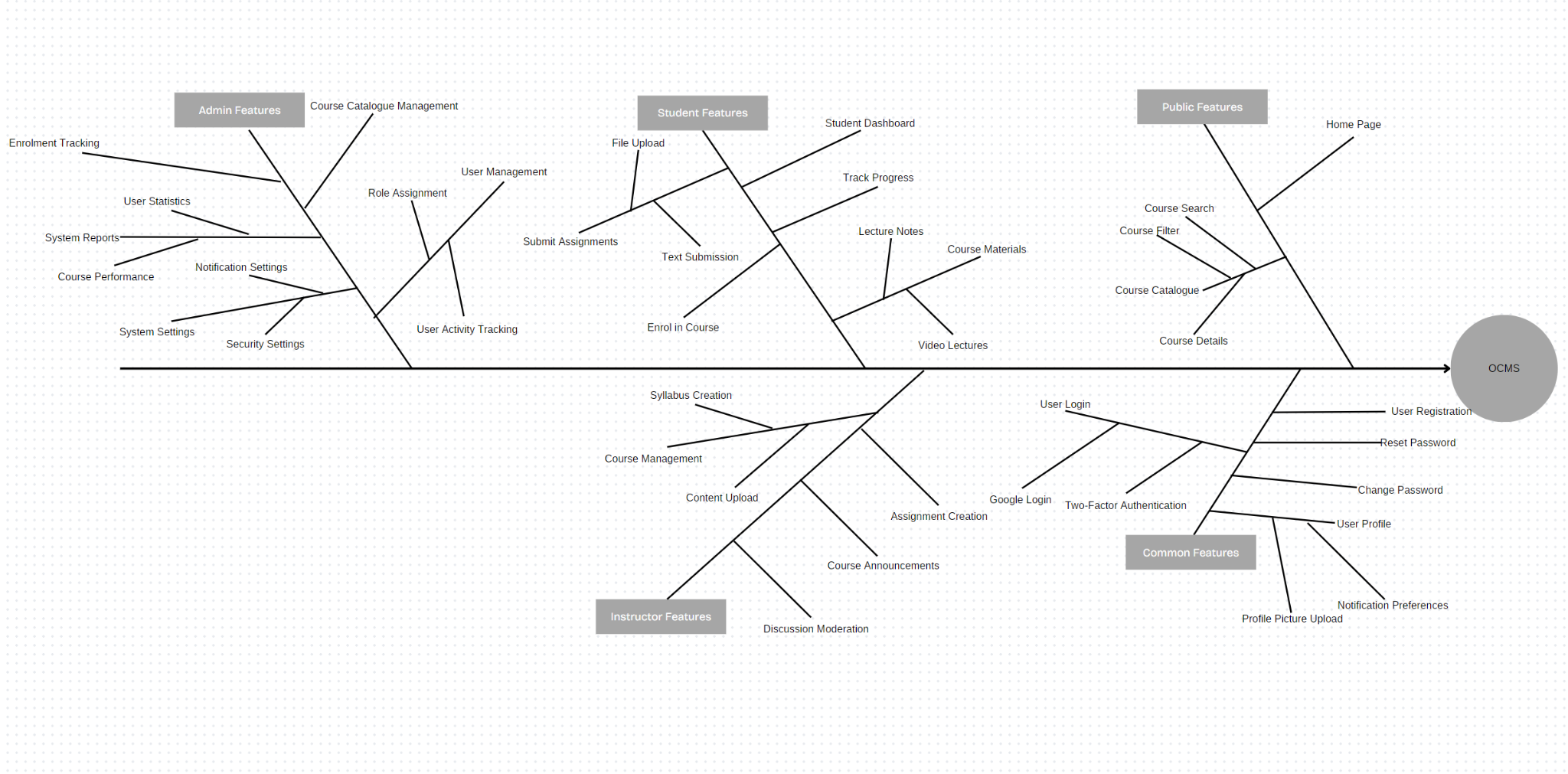
**Third-Party Components**: Assumes that standard Java libraries and tools (e.g., JDBC) will be available without additional dependencies.

**User Access**: Assumes students and instructors will access the system via university networks with standard web browsers.

**External Dependencies**: No reliance on third-party services, only basic dependencies like MySQL and the Java runtime environment.

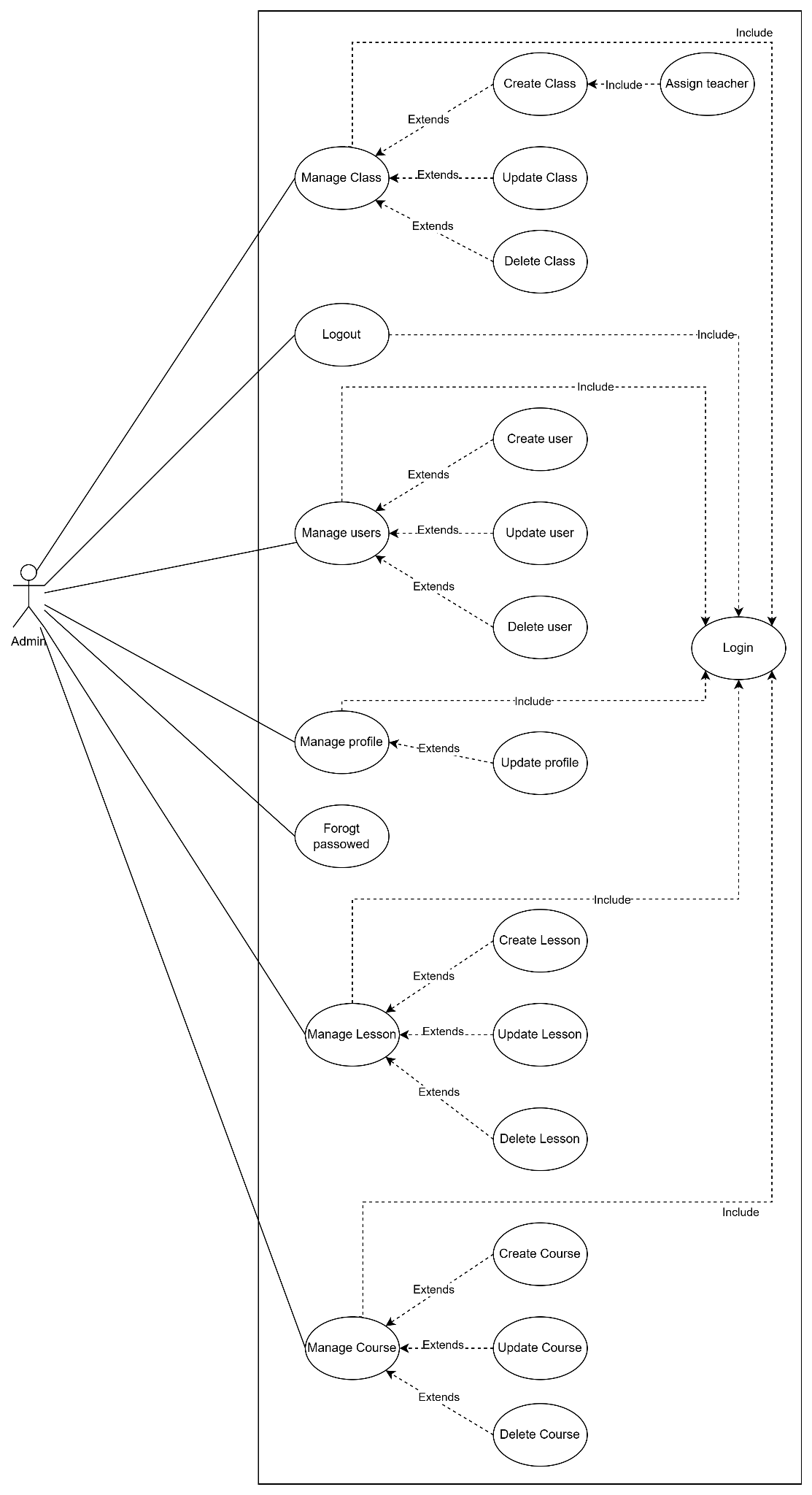
# System Features

## 3.1 System Features Trees

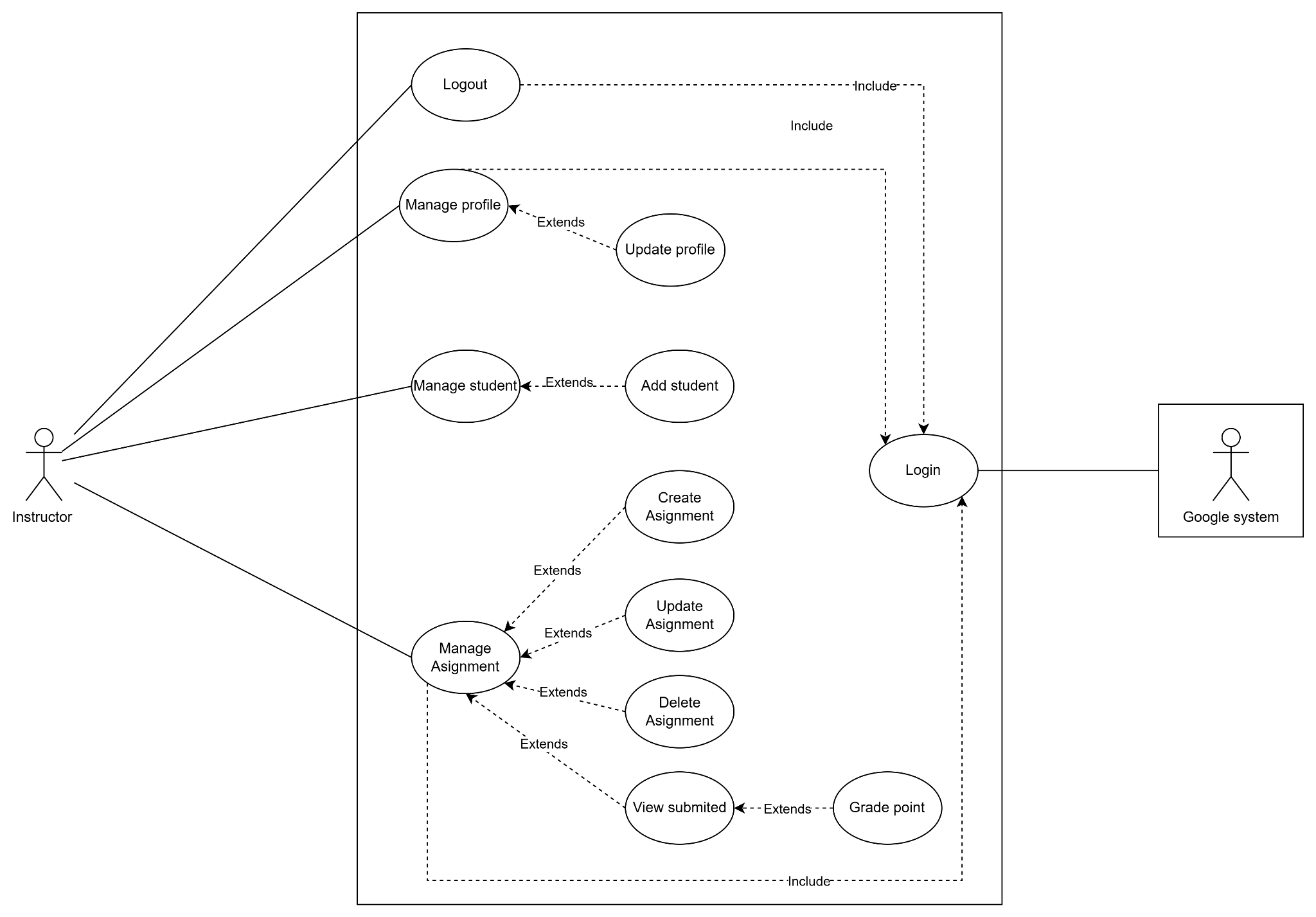


## 3.2 Use Case Diagram

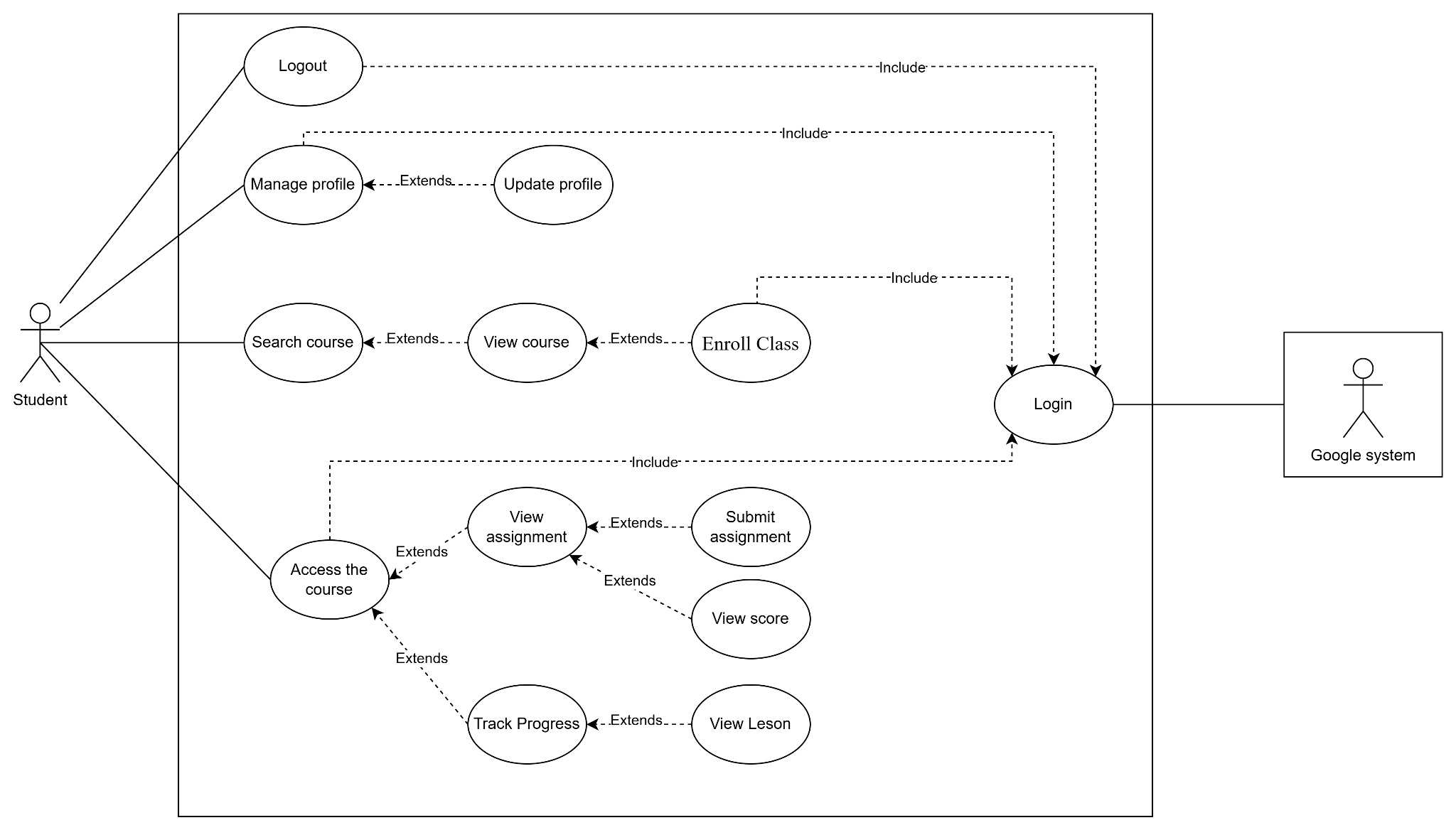
### 3.2.1 UCs for Admin

**

### 3.2.2 Ucs for Instructors

**

### 3.2.3 Ucs for Students

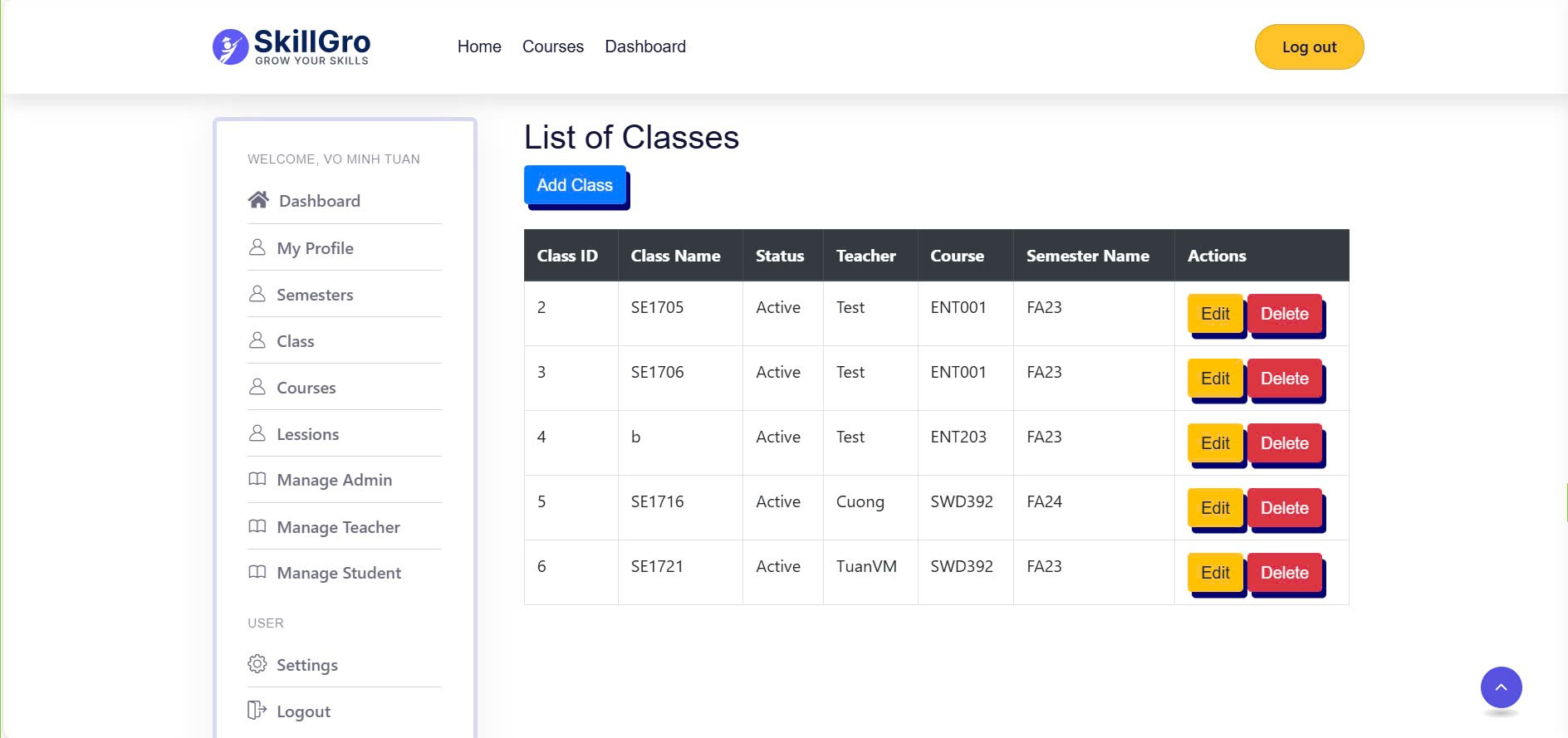


### 3.2.4 UCs Description

### 

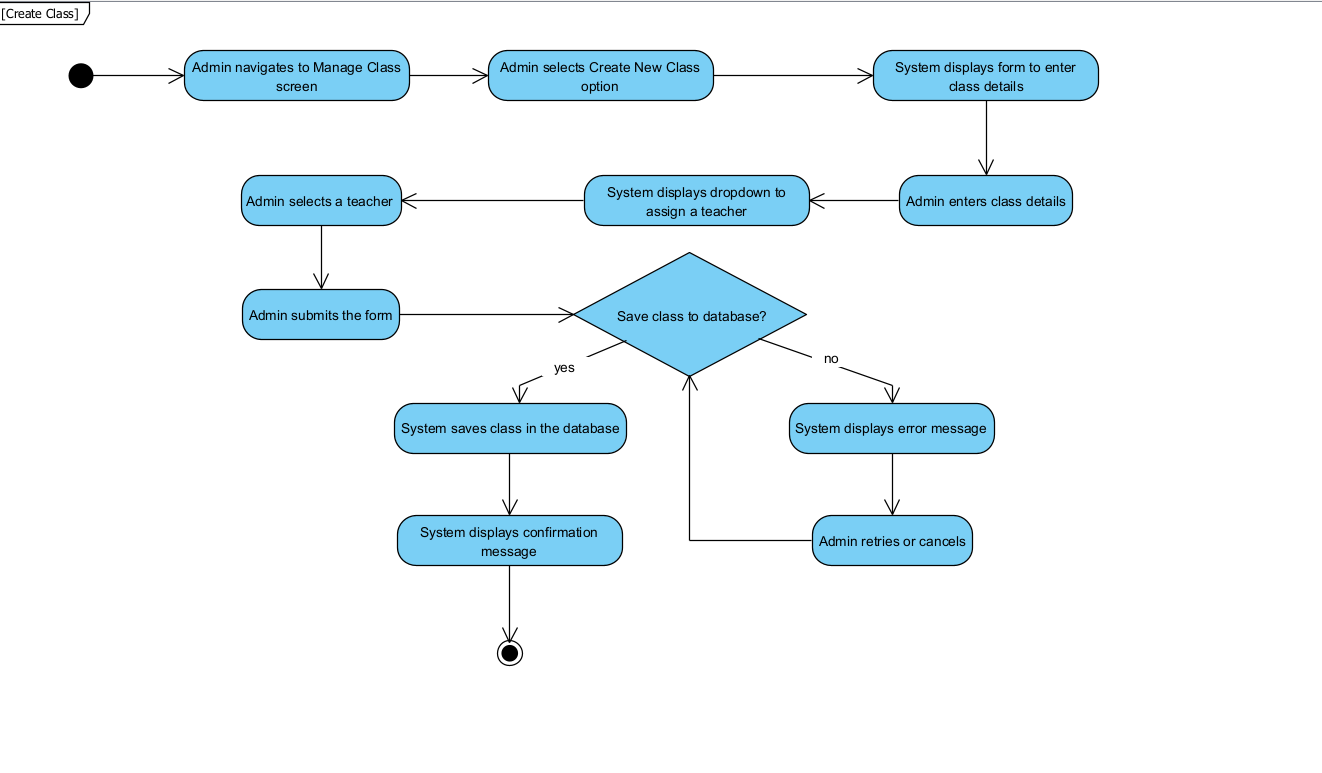
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Use Case** | **Actors** | **Use Case Description** |
| 01 | Manage Class | Admin | The admin can perform actions related to class management, including creating, updating, deleting, and assigning teachers to classes. |
| 02 | Create Class | Admin | Admin can create a new class. This use case includes assigning a teacher to the class as a subtask. |
| 03 | Update Class | Admin | Allows admin to modify the details of an existing class, such as changing the name, schedule, or assigned teacher. |
| 04 | Delete Class | Admin | Admin can remove a class from the system. |
| 05 | Manage Users | Admin | Admin has the ability to create, update, or delete user accounts. This includes managing both student and instructor accounts. |
| 06 | Create User | Admin | Admin can add a new user to the system. |
| 07 | Update User | Admin | Allows admin to update user details such as name, email, role, and status. |
| 08 | Delete User | Admin | Admin can remove a user from the system. |
| 09 | Manage Profile | Admin, Instructor, Student | All actors can update their own profile information. |
| 10 | Update Profile | Admin, Instructor, Student | Allows users to update their personal details, such as name, email, and contact information. |
| 11 | Forgot Password | Admin | Users can reset their password if they forget it. |
| 12 | Manage Lesson | Admin | Admin can create, update, or delete lessons for their courses. |
| 13 | Create Lesson | Admin | Allows instructors to create a new lesson for a course. |
| 14 | Update Lesson | Admin | Instructors can modify the content or details of an existing lesson. |
| 15 | Delete Lesson | Admin | Instructors can remove a lesson from the course. |
| 16 | Manage Course | Admin | Allows for the creation, updating, and deletion of courses. |
| 17 | Create Course | Admin | Allows for adding a new course to the system. |
| 18 | Update Course | Admin | Allows for modifying course details. |
| 19 | Delete Course | Admin | Allows for removing a course from the system. |
| 20 | Manage Assignment | Instructor | Instructors can create, update, delete, and view assignments submitted by students. |
| 21 | Create Assignment | Instructor | Instructors can add a new assignment to a class. |
| 22 | Update Assignment | Instructor | Instructors can change the details or content of an assignment. |
| 23 | Delete Assignment | Instructor | Instructors can remove an assignment from the class. |
| 24 | View Submitted Assignments | Instructor | Instructors can view assignments submitted by students for grading. |
| 25 | Grade Assignment | Instructor | Instructors can assign grades to submitted assignments. |
| 26 | Login | Admin, Instructor, Student | All users can log in to the system using their credentials. |
| 27 | Logout | Admin, Instructor, Student | All users can log out of the system. |
| 28 | Search Course | Student | Students can search for available courses. |
| 29 | Enroll in Class | Student | Students can enroll in classes they are interested in. |
| 30 | Access Course | Student | Students can access course materials after enrolling. |
| 31 | View Assignment | Student | Students can view assignments related to their courses. |
| 32 | Submit Assignment | Student | Students can submit completed assignments for grading. |
| 33 | View Score | Student | Students can view their grades for assignments. |
| 34 | Track Progress | Student | Students can monitor their progress in enrolled courses. |

## 3.3 Functional Requirements



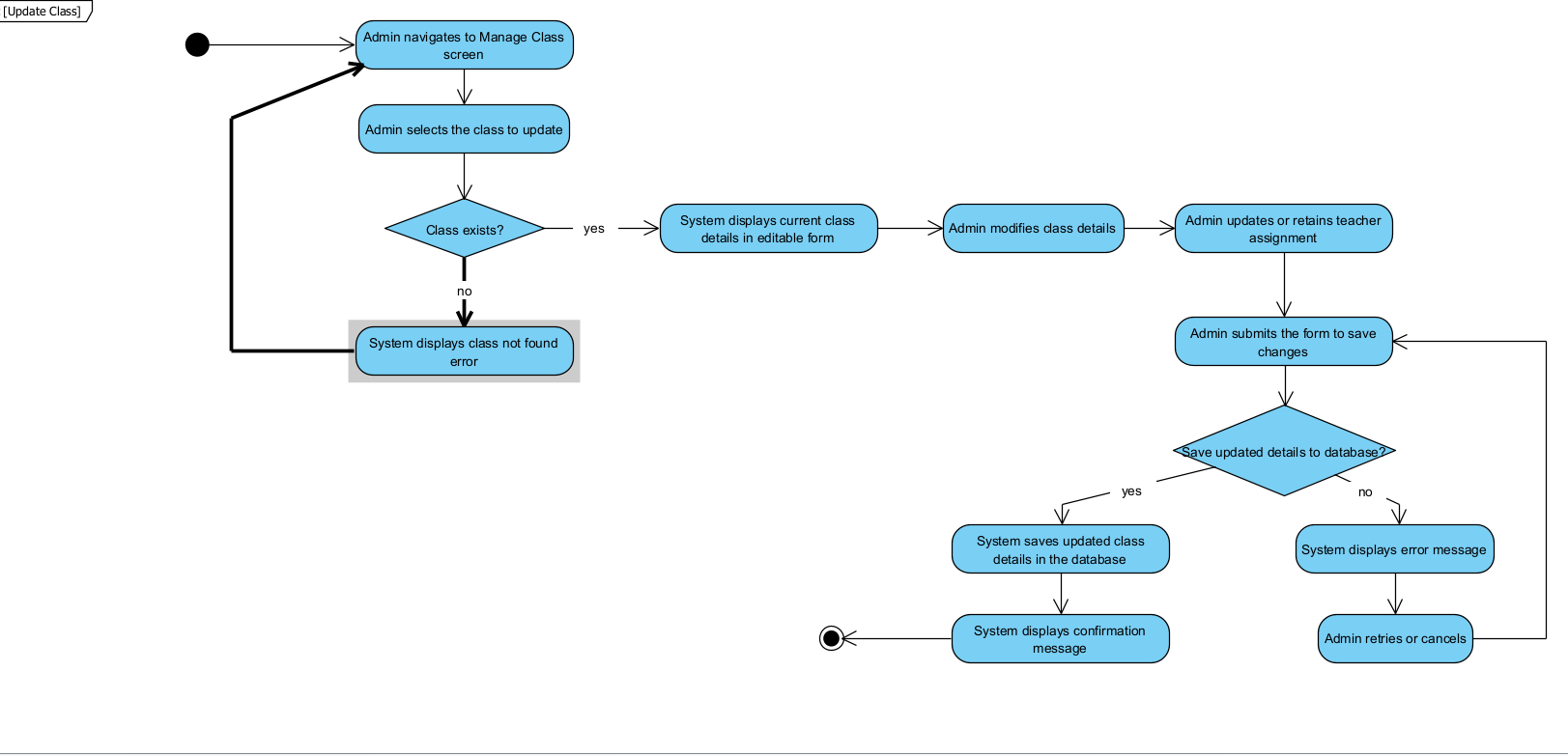
### 3.3.1 Create Class

|  |  |
| --- | --- |
| UC ID and Name: | Create Class |
| Trigger: | The Admin wants to create a new class in the system. |
| Description: | This Use Case allows the Admin to create a new class, including specifying class details and assigning a teacher to manage it. |
| Preconditions: | * The Admin must be logged into the system with permissions to manage classes. * Necessary information for creating a class, such as class name, schedule, and teacher assignment, must be available. |
| Postconditions: | The new class is saved in the database and is accessible in the course catalogue for students and instructors. |
| Normal Flow: | 1. The Admin navigates to the "Manage Class" screen and selects the option to create a new class. 2. The system displays a form prompting the Admin to enter class details (e.g., class name, description, schedule). 3. The Admin fills in the necessary information for the new class. 4. The system displays a dropdown or search option to assign a teacher to the class. 5. The Admin selects a teacher from the list or assigns a new teacher if necessary. 6. The Admin submits the form to create the class. 7. The system saves the new class in the database (E1). 8. The system displays a confirmation message indicating that the class has been successfully created. |
| Alternative Flows: | A1: If the Admin decides not to create the class:   * The Admin cancels the form submission. * The system returns to the "Manage Class" screen without saving any information. |
| Exceptions: | E1: If there is an issue with saving the class in the database (e.g., database connection issue):   * The system displays an error message indicating that the class could not be created. * The Admin is returned to the form to attempt creation again. |
| Priority: | High |
| Frequency of Use: | Occasionally, when new classes need to be added to the system. |
| Business Rules: | * The class name must be unique in the system. * A teacher must be assigned to every class. * All mandatory fields must be completed before the class can be created. |
| Other Information: | None |
| Assumptions: | The Admin has verified all necessary details for the class before proceeding with creation. |



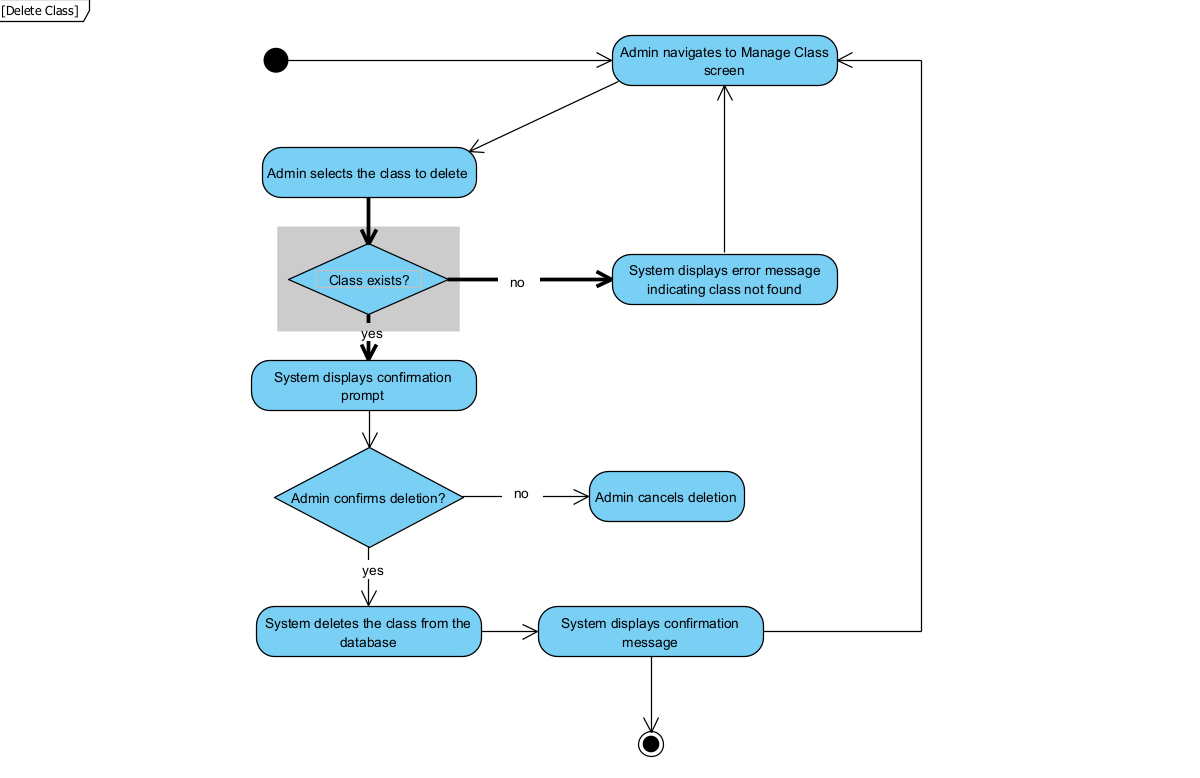
### 3.3.2 Update Class

|  |  |
| --- | --- |
| UC ID and Name: | Update Class |
| Trigger: | The Admin wants to update information about an existing class in the system. |
| Description: | This Use Case allows the Admin to edit and update details of an existing class, such as class name, description, schedule, and teacher assignment. The updated information will replace the existing data in the database. |
| Preconditions: | * The Admin must be logged into the system with permissions to manage classes. * The class to be updated must exist in the system. |
| Postconditions: | The system saves the updated class details in the database, making the revised information available for students and instructors. |
| Normal Flow: | 1. The Admin navigates to the "Manage Class" screen and selects the class to update. 2. The system displays the current details of the selected class in an editable form. 3. The Admin modifies the necessary details (e.g., class name, description, schedule). 4. The system displays a dropdown or search option for updating the assigned teacher if required. 5. The Admin updates the teacher assignment or retains the current assignment. 6. The Admin submits the form to save the changes. 7. The system saves the updated class details in the database (E1). 8. The system displays a confirmation message indicating that the class has been successfully updated. |
| Alternative Flows: | A1: If the Admin decides not to update the class:   * The Admin cancels the form submission. * The system returns to the "Manage Class" screen without making any changes. |
| Exceptions: | E1: If there is an issue with saving the updated class information in the database (e.g., database connection issue):   * The system displays an error message indicating that the class could not be updated. * The Admin is returned to the form to attempt the update again.   E2: If the selected class does not exist in the system (e.g., another Admin deleted it):   * The system displays an error message indicating that the class could not be found. * The Admin is returned to the "Manage Class" screen without making any changes. |
| Priority: | High |
| Frequency of Use: | Occasionally, when class details need to be revised. |
| Business Rules: | * The class name must remain unique in the system. * Only users with Admin permissions can update class information. * All mandatory fields must be completed before the updated information can be saved. |
| Other Information: | None |
| Assumptions: | The Admin has verified that the information being updated is accurate and necessary before submitting the changes. |



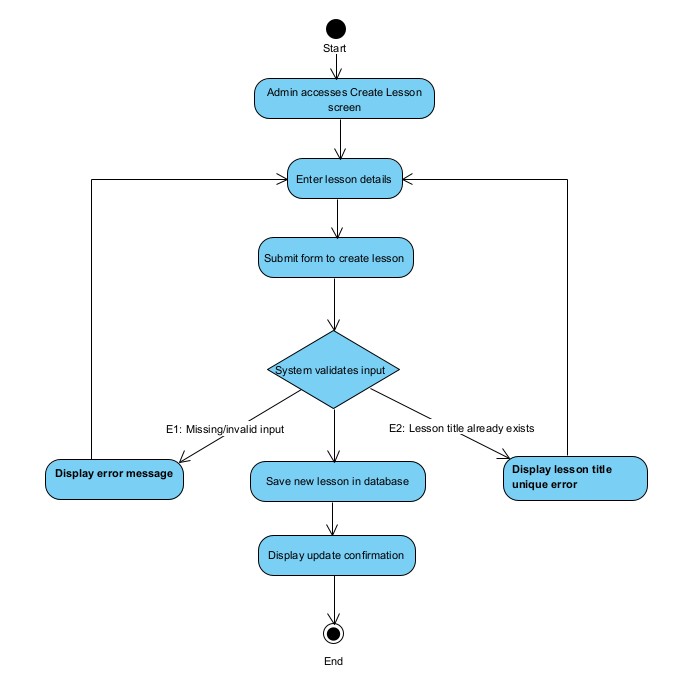
### 3.3.3 Delete Class

|  |  |
| --- | --- |
| UC ID and Name: | Delete Class |
| Trigger: | The Admin wants to delete an existing class from the system. |
| Description: | This Use Case allows the Admin to permanently delete an existing class from the system, removing it from the database and making it no longer accessible to students and instructors. |
| Preconditions: | * The Admin must be logged into the system with permissions to manage classes. * The class to be deleted must already exist in the system. |
| Postconditions: | The class is permanently removed from the database and is no longer available in the system. |
| Normal Flow: | 1. The Admin navigates to the "Manage Class" screen and selects the class to delete. 2. The system displays a confirmation prompt asking if the Admin is sure about deleting the class. 3. The Admin confirms the deletion. 4. The system deletes the class from the database (E1). 5. The system displays a confirmation message indicating that the class has been successfully deleted. |
| Alternative Flows: | A1: If the Admin decides not to delete the class:   * The Admin cancels the confirmation prompt. * The system returns to the "Manage Class" screen without making any changes. |
| Exceptions: | E1: **If the class does not exist in the database** (e.g., another Admin already deleted it):   * The system displays an error message indicating that the class could not be found. * The Admin is returned to step 1 to reselect a class if desired. |
| Priority: | Medium |
| Frequency of Use: | Rare, only when classes need to be removed. |
| Business Rules: | * Deleted classes cannot be recovered. * The system should prompt for confirmation before deleting a class. |
| Other Information: | None |
| Assumptions: | The Admin has verified that the class is no longer needed before proceeding with the deletion. |



### 3.3.4 Create new lesson

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Create new Lesson | | |
| Created By: | TuanVM | Date Created: | 30/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | The Admin wants to add a new lesson to the system. | | |
| Description: | This Use Case allows the Admin to create a new lesson in the system, providing essential information such as the lesson title, description. | | |
| Preconditions: | The Admin must be logged into the system with permissions to manage lessons. | | |
| Postconditions: | The new lesson is successfully added and stored in the system's database. | | |
| Normal Flow: | 1.The Admin navigates to the "Create Lesson" screen.  2.The Admin enters the required information for the new lesson:   * Title of the lesson * Description of the lesson content   3.The Admin submits the form to create the lesson.  4.The system validates the input to ensure all required fields are filled and that the information is formatted correctly (E1, E2).  5.The system saves the new lesson in the database.  6.The system confirms that the lesson has been successfully created and provides a confirmation message to the Admin. | | |
| Alternative Flows: | **A1:** If required fields are missing:   1. The system displays an error message indicating which fields are missing (E1). 2. The Admin re-enters the missing information and resubmits the form, returning to step 4. | | |
| Exceptions: | **E1:** If required fields are missing or improperly formatted:   * The system displays an error message indicating which fields are missing or incorrectly formatted. * The Admin is prompted to re-enter the information and return to step 4.   **E2:** If the lesson title already exists in the system:   * The system displays an error message indicating that the lesson title must be unique. * The Admin is prompted to enter a different title and return to step 4. | | |
| Priority: | High | | |
| Frequency of Use: | Occasionally, when new lessons are introduced. | | |
| Business Rules: | The lesson title must be unique.  All required fields (title, description) must be completed. | | |
| Other Information: | None | | |
| Assumptions: | The Admin has a clear understanding of the lesson details to be created. | | |



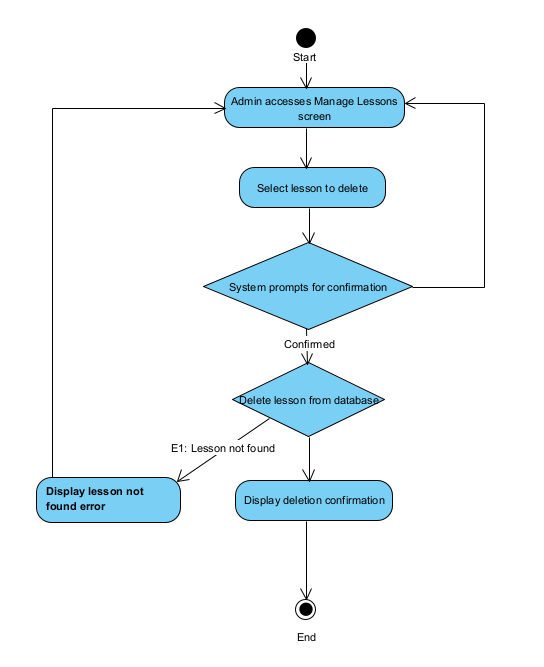
### 3.3.5 Update lesson

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Update Lesson | | |
| Created By: | TuanVM | Date Created: | 30/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | The Admin wants to update the details of an existing lesson. | | |
| Description: | This Use Case allows the Admin to modify the information of an existing lesson in the system. | | |
| Preconditions: | The Admin must be logged into the system with permissions to manage lessons, and the lesson to be updated must already exist in the system. | | |
| Postconditions: | The lesson's details are successfully updated in the database. | | |
| Normal Flow: | 1.The Admin navigates to the "Manage Lessons" screen and selects the lesson to update.  2.The system retrieves the current details of the selected lesson and displays them on the "Update Lesson" screen.  3.The Admin modifies the desired information, such as:   * Title of the lesson * Description of the lesson content   4.The Admin submits the updated information.  5.The system validates the input to ensure all required fields are filled and that the information is formatted correctly (E1, E2).  6.The system saves the updated details of the lesson in the database.  7.The system confirms that the lesson has been successfully updated and provides a confirmation message to the Admin. | | |
| Alternative Flows: | **A1:** If required fields are missing:   1. The system displays an error message indicating which fields are missing (E1). 2. The Admin re-enters the missing information and resubmits the form, returning to step 5. | | |
| Exceptions: | **E1:** If required fields are missing or improperly formatted:   * The system displays an error message indicating which fields are missing or incorrectly formatted. * The Admin is prompted to re-enter the information and return to step 5.   **E2:** If the updated lesson title already exists in the system:   * The system displays an error message indicating that the lesson title must be unique. * The Admin is prompted to enter a different title and return to step 5. | | |
| Priority: | High | | |
| Frequency of Use: | Occasionally, when updates to lesson details are necessary. | | |
| Business Rules: | The lesson title must be unique.  All required fields (title, description) must be completed. | | |
| Other Information: | None | | |
| Assumptions: | The Admin has a clear understanding of the lesson details to be updated. | | |

### 

### 3.3.6 Delete lesson

| UC ID and Name: | Delete Lesson | | |
| --- | --- | --- | --- |
| Created By: | TuanVM | Date Created: | 30/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | The Admin wants to delete an existing lesson. | | |
| Description: | This Use Case allows the Admin to delete an existing lesson from the system, permanently removing it from the database. | | |
| Preconditions: | The Admin must be logged into the system with permissions to manage lessons, and the lesson to be deleted must already exist in the system. | | |
| Postconditions: | The lesson is permanently removed from the database and is no longer available in the system. | | |
| Normal Flow: | 1.The Admin navigates to the "Manage Lessons" screen and selects the lesson to delete.  2.The system displays a confirmation prompt asking if the Admin is sure 3.about deleting the lesson.  4.The Admin confirms the deletion.  5.The system deletes the lesson from the database (E1).  6.The system displays a confirmation message indicating that the lesson has been successfully deleted. | | |
| Alternative Flows: | **A1:** If the Admin decides not to delete the lesson:   1. The Admin cancels the confirmation prompt. 2. The system returns to the "Manage Lessons" screen without making any changes. | | |
| Exceptions: | **E1:** If the lesson does not exist in the database (e.g., another Admin already deleted it):   * The system displays an error message indicating that the lesson could not be found. * The Admin is returned to step 1. | | |
| Priority: | Medium | | |
| Frequency of Use: | Rare, only when lessons need to be removed. | | |
| Business Rules: | Deleted lessons cannot be recovered.  The system should prompt for confirmation before deleting a lesson. | | |
| Other Information: | None | | |
| Assumptions: | The Admin has verified that the lesson is no longer needed before proceeding with the deletion. | | |



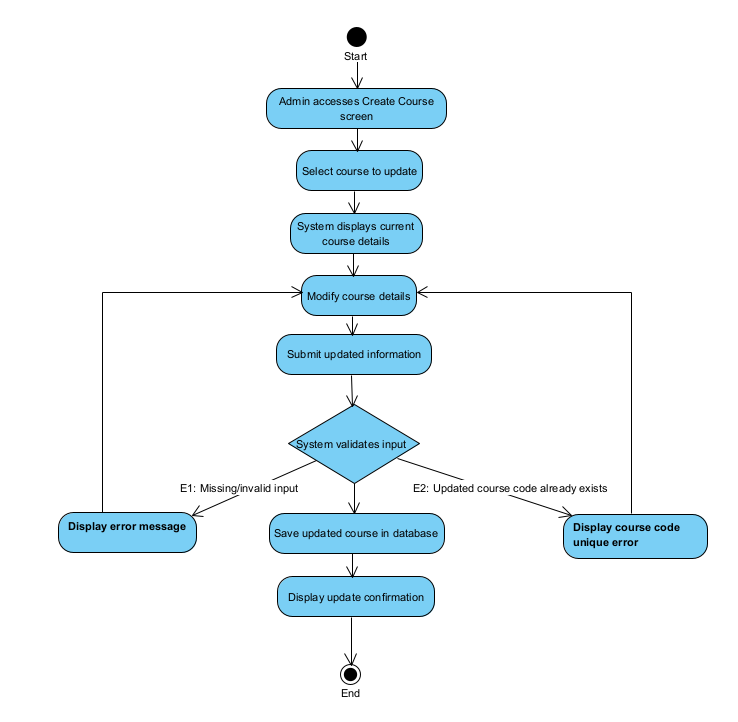
### 

### 3.3.7 Create New Course

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Create new Course | | |
| Created By: | TuanVM | Date Created: | 30/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | The Admin wants to add a new course to the system. | | |
| Description: | This Use Case allows the Admin to create a new course in the system by providing essential information such as the course name, code, description. | | |
| Preconditions: | The Admin must be logged into the system with permissions to manage courses. | | |
| Postconditions: | The new course is successfully added and stored in the system's database. | | |
| Normal Flow: | 1.The Admin navigates to the "Create Course" screen.  2.The Admin enters the required information for the new course:   * Course name * Code * Description of the course content   3.The Admin submits the form to create the course.  4.The system validates the input to ensure all required fields are filled and that the information is formatted correctly (E1, E2).  5.The system saves the new course in the database.  6.The system confirms that the course has been successfully created and provides a confirmation message to the Admin. | | |
| Alternative Flows: | **A1:** If required fields are missing:   1. The system displays an error message indicating which fields are missing (E1). 2. The Admin re-enters the missing information and resubmits the form, returning to step 4. | | |
| Exceptions: | **E1**: If required fields are missing or improperly formatted:   * The system displays an error message indicating which fields are missing or incorrectly formatted. * The Admin is prompted to re-enter the information and return to step 4.   **E2**: If the course name already exists in the system:   * The system displays an error message indicating that the course name must be unique. * The Admin is prompted to enter a different name and return to step 4. | | |
| Priority: | High | | |
| Frequency of Use: | Occasionally, when new courses are introduced. | | |
| Business Rules: | The course code must be unique.  All required fields (course name,code , description) must be completed. | | |
| Other Information: | None | | |
| Assumptions: | The Admin has a clear understanding of the course details to be created. | | |

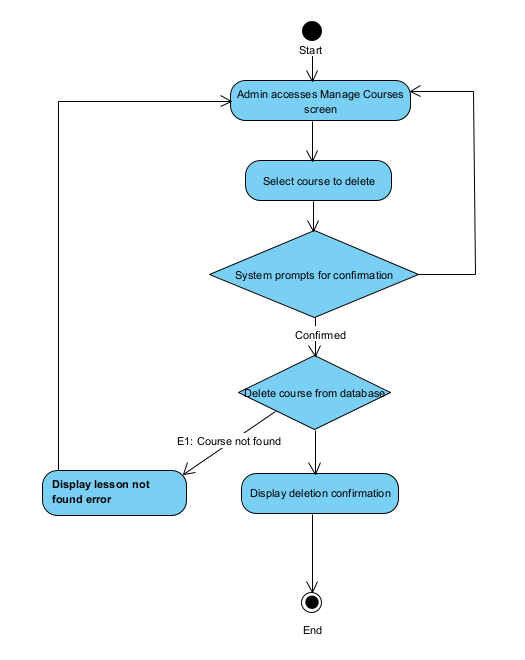
### 3.3.8 Update Course

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Update Course | | |
| Created By: | TuanVM | Date Created: | 30/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | The Admin wants to update the details of an existing course. | | |
| Description: | This Use Case allows the Admin to modify the information of an existing course in the system. | | |
| Preconditions: | The Admin must be logged into the system with permissions to manage courses, and the course to be updated must already exist in the system. | | |
| Postconditions: | The course's details are successfully updated in the database. | | |
| Normal Flow: | 1.The Admin navigates to the "Manage Courses" screen and selects the course to update.  2.The system retrieves the current details of the selected course and displays them on the "Update Course" screen.  3.The Admin modifies the desired information, such as:   * Course name * Code * Description of the course content   4.The Admin submits the updated information.  5.The system validates the input to ensure all required fields are filled and that the information is formatted correctly (E1, E2).  6.The system saves the updated details of the course in the database.  7.The system confirms that the course has been successfully updated and provides a confirmation message to the Admin. | | |
| Alternative Flows: | **A1:** If required fields are missing:   1. The system displays an error message indicating which fields are missing (E1). 2. The Admin re-enters the missing information and resubmits the form, returning to step 5. | | |
| Exceptions: | **E1:** If required fields are missing or improperly formatted:   * The system displays an error message indicating which fields are missing or incorrectly formatted. * The Admin is prompted to re-enter the information and return to step 5.   **E2:** If the updated course name already exists in the system:   * The system displays an error message indicating that the course name must be unique. * The Admin is prompted to enter a different name and return to step 5. | | |
| Priority: | High | | |
| Frequency of Use: | Occasionally, when updates to course details are necessary. | | |
| Business Rules: | The course code must be unique (if changed).  All required fields (course name,code ,description) must be completed. | | |
| Other Information: | None | | |
| Assumptions: | The Admin knows the course details that need to be updated. | | |



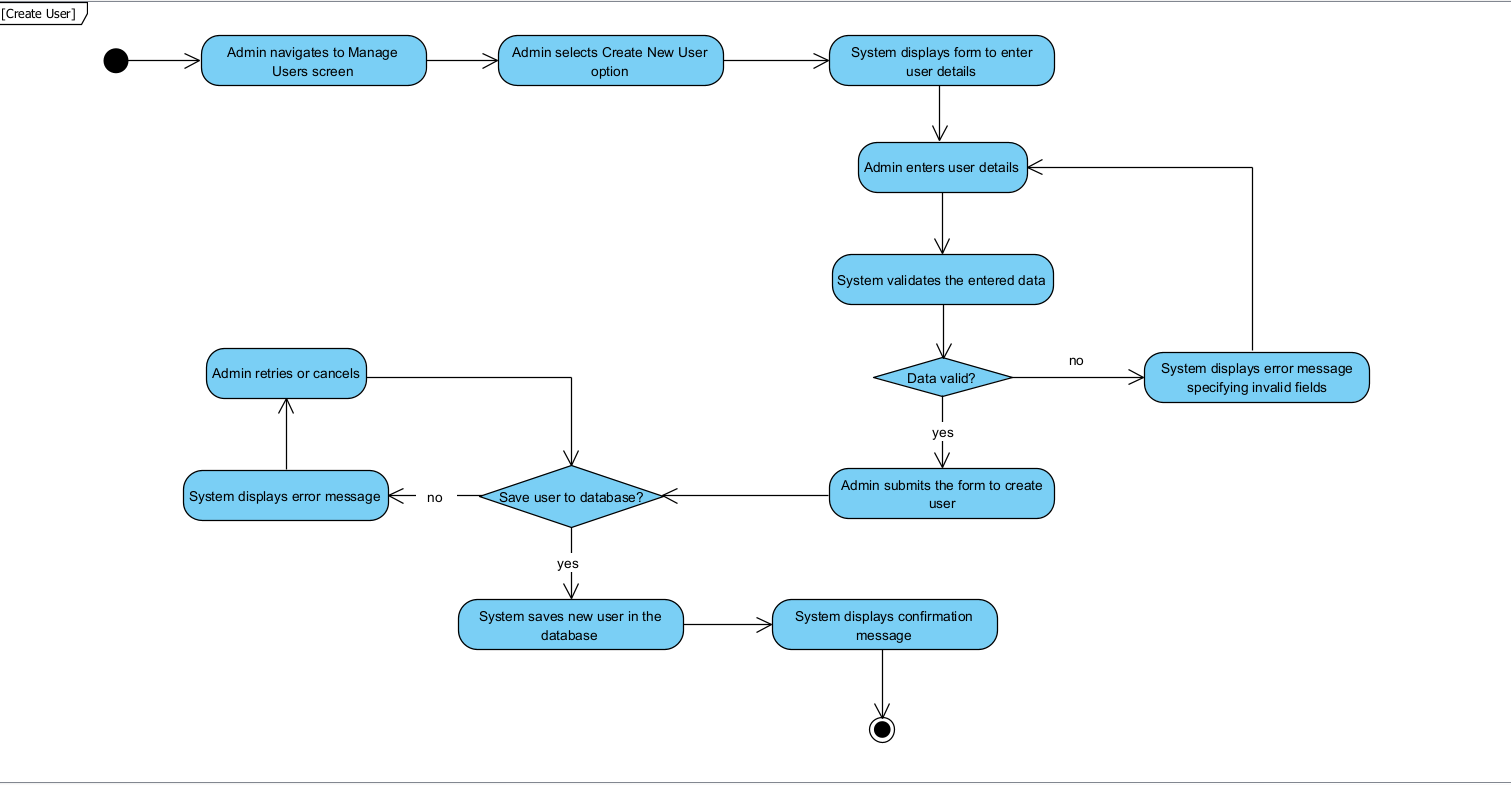
### 3.3.9 Delete Course

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Delete Course | | |
| Created By: | TuanVM | Date Created: | 30/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | The Admin wants to delete an existing course from the system. | | |
| Description: | This Use Case allows the Admin to delete an existing course from the system, permanently removing it from the database. | | |
| Preconditions: | The Admin must be logged into the system with permissions to manage courses, and the course to be deleted must already exist in the system. | | |
| Postconditions: | The course is permanently removed from the database and is no longer available in the system. | | |
| Normal Flow: | 1.The Admin navigates to the "Manage Courses" screen and selects the course to delete.  2.The system displays a confirmation prompt asking if the Admin is sure about deleting the course.  3.The Admin confirms the deletion.  4.The system deletes the course from the database (E1).  5.The system displays a confirmation message indicating that the course has been successfully deleted. | | |
| Alternative Flows: | **A1:** If the Admin decides not to delete the course:   1. The Admin cancels the confirmation prompt. 2. The system returns to the "Manage Courses" screen without making any changes. | | |
| Exceptions: | **E1:** If the course does not exist in the database (e.g., another Admin already deleted it):   * The system displays an error message indicating that the course could not be found. * The Admin is returned to step 1. | | |
| Priority: | Medium | | |
| Frequency of Use: | Rare, only when courses need to be removed. | | |
| Business Rules: | Deleted courses cannot be recovered.  The system should prompt for confirmation before deleting a course. | | |
| Other Information: | None | | |
| Assumptions: | The Admin has verified that the course is no longer needed before proceeding with the deletion. | | |



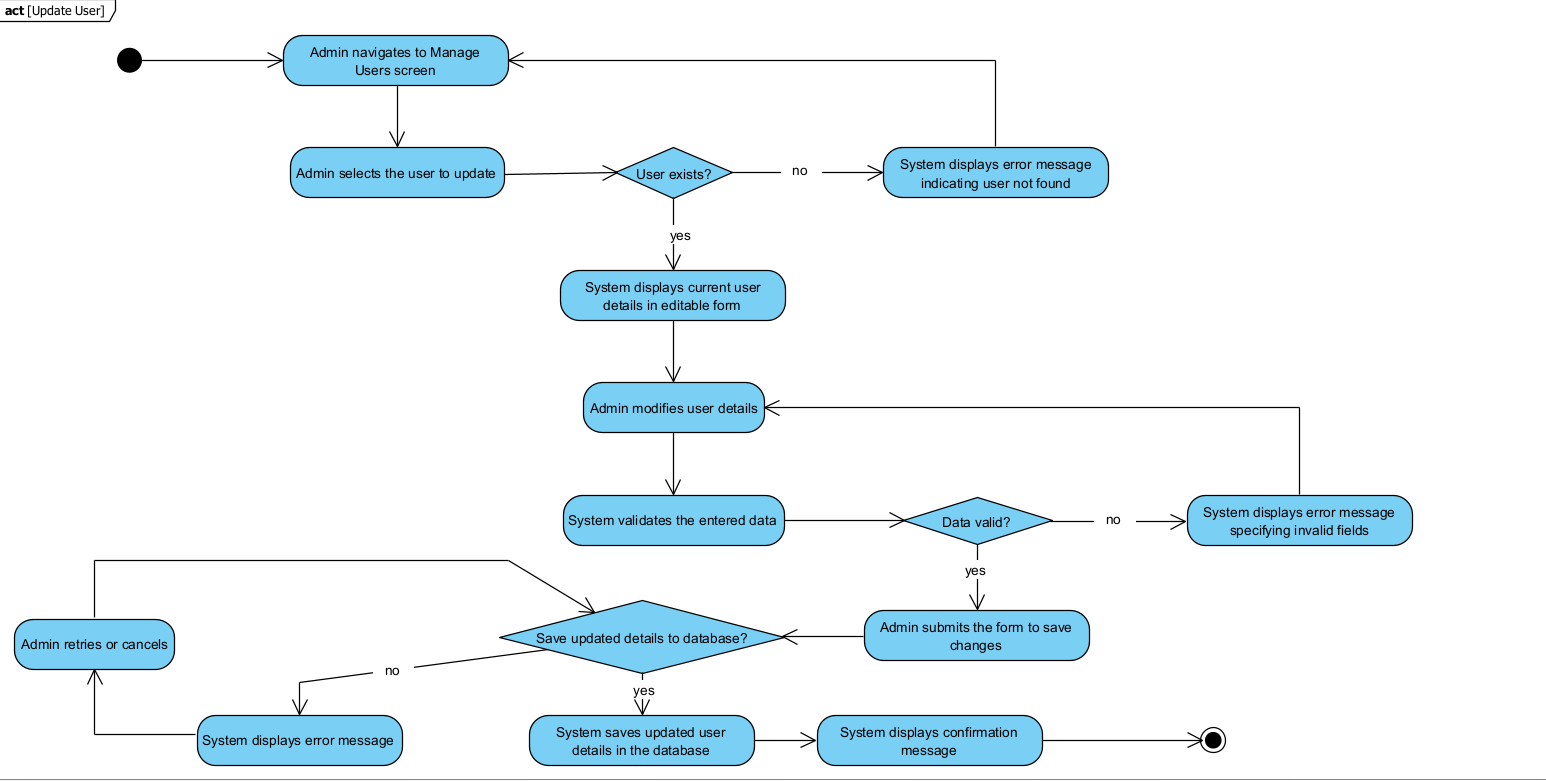
### 3.3.10 Create User

|  |  |
| --- | --- |
| UC ID and Name: | Create User |
| Trigger: | The Admin wants to create a new user account in the system. |
| Description: | This Use Case allows the Admin to create a new user account, which can be for a student, instructor, or another admin. The Admin will input necessary details for the user and assign an appropriate role. |
| Preconditions: | * The Admin must be logged into the system with permissions to manage users. * The Admin should have all necessary information about the user to be created (e.g., name, email, role). |
| Postconditions: | A new user account is created in the database with the specified role and credentials, allowing the new user to access the system based on their assigned role. |
| Normal Flow: | 1. The Admin navigates to the "Manage Users" screen and selects the option to create a new user. 2. The system displays a form prompting the Admin to enter the user details (e.g., name, email, role). 3. The Admin fills in the necessary information for the new user. 4. The system validates the entered data to ensure it meets format and length requirements. 5. The Admin submits the form to create the user account. 6. The system saves the new user account in the database (E1). 7. The system displays a confirmation message indicating that the user account has been successfully created. |
| Alternative Flows: | A1: If the Admin decides not to create the user:   * The Admin cancels the form submission. * The system returns to the "Manage Users" screen without saving any information. |
| Exceptions: | E1: If there is an issue with saving the user account in the database (e.g., database connection issue):   * The system displays an error message indicating that the user could not be created. * The Admin is returned to the form to attempt creation again.   E2: If the entered data is invalid (e.g., email format is incorrect):   * The system displays an error message specifying the invalid field(s). * The Admin is prompted to correct the data before resubmitting the form. |
| Priority: | High |
| Frequency of Use: | Occasionally, when new user accounts need to be added. |
| Business Rules: | * The email for each user must be unique in the system. * The Admin must select an appropriate role for the user (e.g., student, instructor, admin). * All mandatory fields must be completed before the account can be created. |
| Other Information: | None |
| Assumptions: | The Admin has verified the accuracy of the information provided for the user before submitting. |



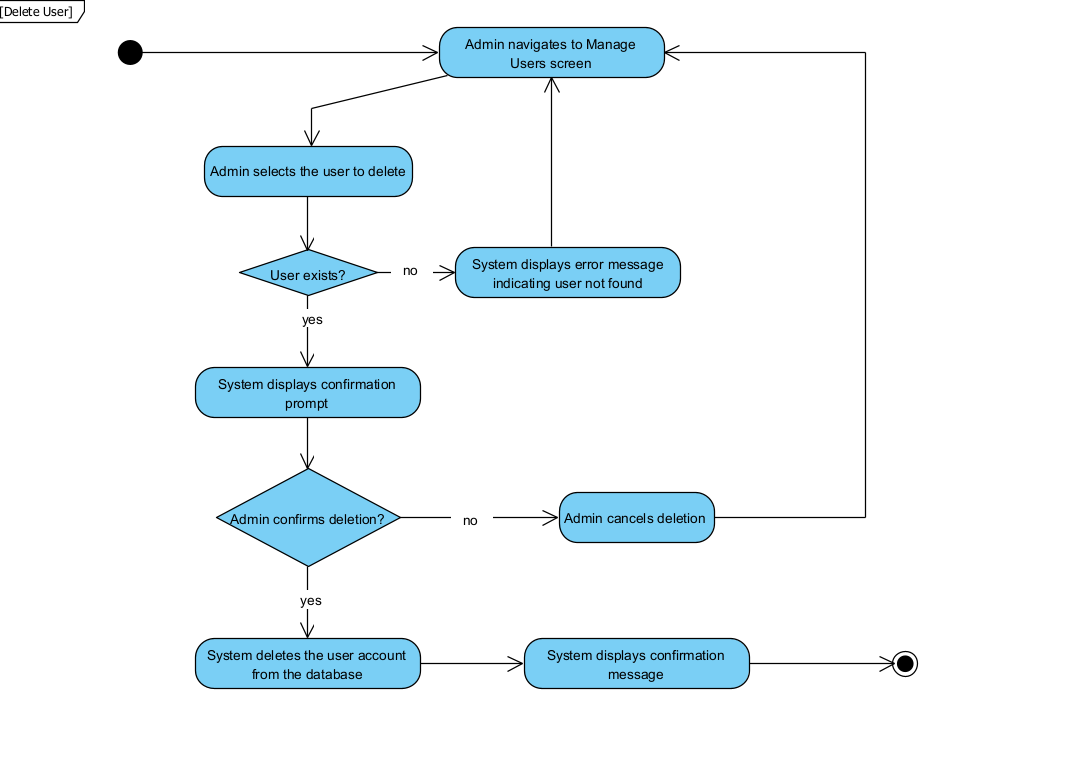
### 3.3.11 Update User

|  |  |
| --- | --- |
| UC ID and Name: | Update User |
| Trigger: | The Admin wants to update information for an existing user in the system. |
| Description: | This Use Case allows the Admin to edit and update details of an existing user account, such as name, email, role, and other necessary information. The updated information will replace the existing data in the database, allowing the user to access the system with their updated details. |
| Preconditions: | * The Admin must be logged into the system with permissions to manage users. * The user account to be updated must already exist in the system. |
| Postconditions: | The system saves the updated user details in the database, making the revised information available immediately. |
| Normal Flow: | 1. The Admin navigates to the "Manage Users" screen and selects the user account to update. 2. The system displays the current details of the selected user in an editable form. 3. The Admin modifies the necessary details (e.g., name, email, role). 4. The system validates the entered data to ensure it meets format and length requirements. 5. The Admin submits the form to save the changes. 6. The system saves the updated user details in the database (E1). 7. The system displays a confirmation message indicating that the user details have been successfully updated. |
| Alternative Flows: | A1: If the Admin decides not to update the user:   * The Admin cancels the form submission. * The system returns to the "Manage Users" screen without making any changes. |
| Exceptions: | E1: If there is an issue with saving the updated user information in the database (e.g., database connection issue):   * The system displays an error message indicating that the user could not be updated. * The Admin is returned to the form to attempt the update again.   E2: If the selected user account does not exist in the system (e.g., another Admin deleted it):   * The system displays an error message indicating that the user could not be found. * The Admin is returned to the "Manage Users" screen.   E3: If the entered data is invalid (e.g., email format is incorrect):   * The system displays an error message specifying the invalid field(s). * The Admin is prompted to correct the data before resubmitting the form. |
| Priority: | High |
| Frequency of Use: | Occasionally, when user information needs to be revised. |
| Business Rules: | * The email for each user must be unique in the system. * Only users with Admin permissions can update user information. * All mandatory fields must be completed before the information can be saved. |
| Other Information: | None |
| Assumptions: | The Admin has verified the accuracy and necessity of the changes before submitting. |

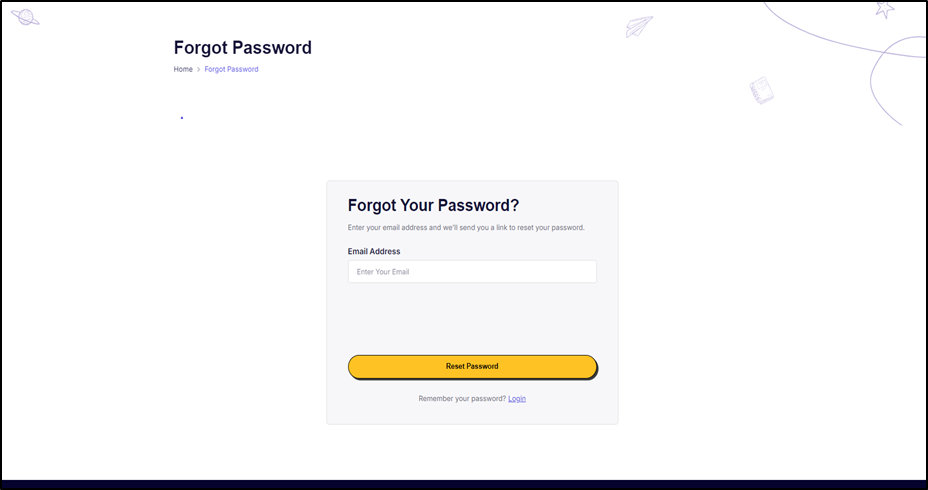


### 3.3.12 Delete User

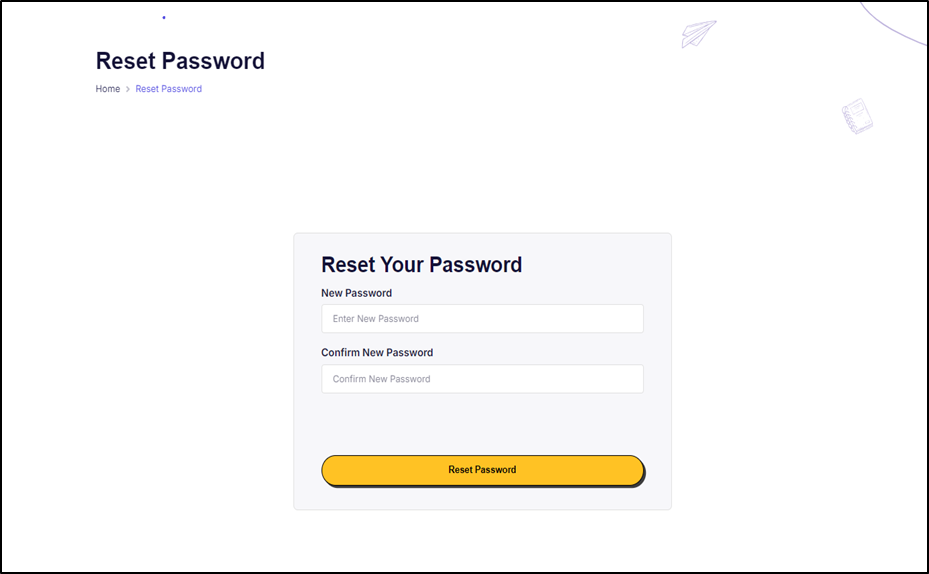
| UC ID and Name: | Delete Users |
| --- | --- |
| Trigger: | The Admin wants to delete an existing user account from the system. |
| Description: | This Use Case allows the Admin to permanently delete a user account from the system, removing it from the database and making it no longer accessible to the user. |
| Preconditions: | * The Admin must be logged into the system with permissions to manage users. * The user account to be deleted must already exist in the system. |
| Postconditions: | The user account is permanently removed from the database and is no longer available in the system. |
| Normal Flow: | 1. The Admin navigates to the "Manage Users" screen and selects the user account to delete. 2. The system displays a confirmation prompt asking if the Admin is sure about deleting the user account. 3. The Admin confirms the deletion. 4. The system deletes the user account from the database (E1). 5. The system displays a confirmation message indicating that the user account has been successfully deleted. |
| Alternative Flows: | A1: If the Admin decides not to delete the user:   * The Admin cancels the confirmation prompt. * The system returns to the "Manage Users" screen without making any changes. |
| Exceptions: | E1: If the user account does not exist in the database (e.g., another Admin already deleted it):   * The system displays an error message indicating that the user account could not be found. * The Admin is returned to step 1 to reselect a user if desired. |
| Priority: | Medium |
| Frequency of Use: | Rare, only when user accounts need to be removed. |
| Business Rules: | * Deleted user accounts cannot be recovered. * The system should prompt for confirmation before deleting a user account. |
| Other Information: | None |
| Assumptions: | The Admin has verified that the user account is no longer needed before proceeding with the deletion. |



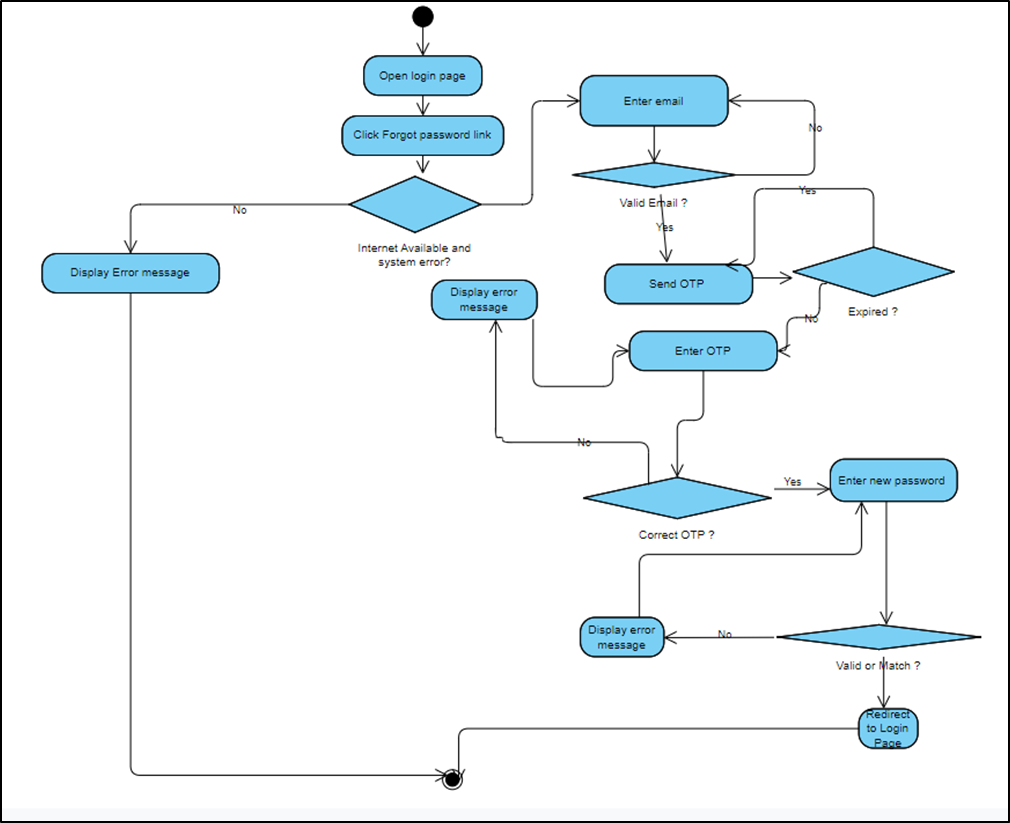
3.3.13 Forgot password





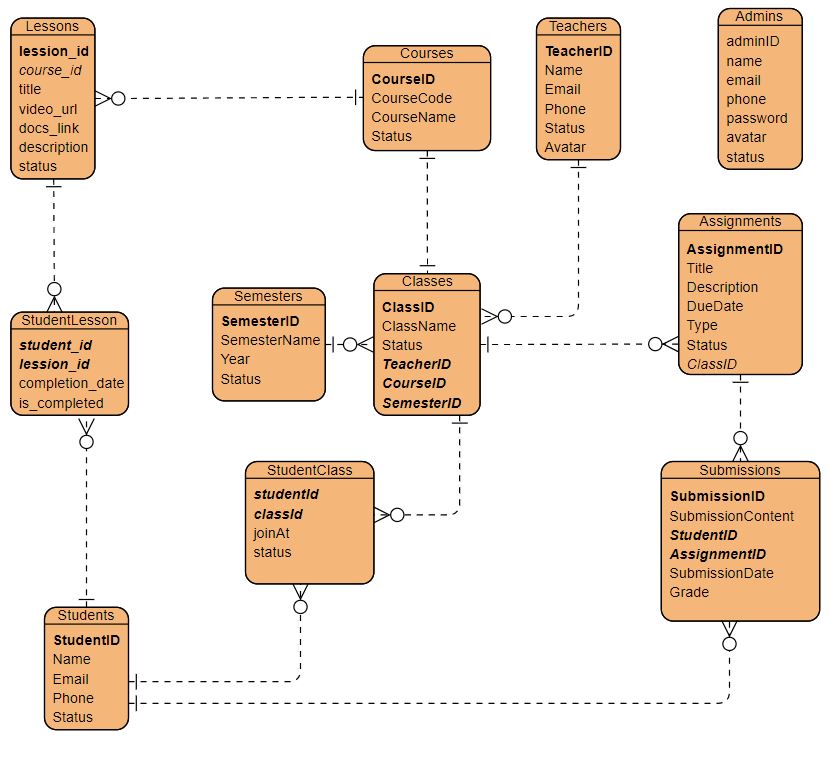


|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Forgot Password | | |
| Created By: | Phan Tiến Mạnh | Date Created: | 10/10/2024 |
| Primary Actor: | Student, Instructor | Secondary Actors: |  |
| Trigger: | The student/instructor has forgotten their password and needs to recover or reset it to access the OCMS platform. | | |
| Description: | This use case describes how the student/instructor resets their password when they have forgotten it. | | |
| Preconditions: | 1. The student/instructor has an active account on OCMS.  2. The student/instructor has access to the email linked to their account. | | |
| Postconditions: | 1. The student/instructor successfully resets their password and can log into the OCMS platform. | | |
| Normal Flow: | 1. The student/instructor opens the OCMS login page.  2. The student/instructor clicks the "Forgot Password" link.  3. The system prompts the student/instructor to enter their registered email address.  4. The system sends a password reset link or code to the provided email. (E2)  5. The student/instructor enters the code sent to email to reset the password another page . (E1)  6. The student/instructor creates a new password and confirms it.  7. The system validates the new password and updates the account.  8. The system confirms that the password reset is successful, and the student/instructor can now log in with the new credentials. | | |
| Alternative Flows: | None | | |
| Exceptions: | E1. **Invalid email provided**: If the user enters an unregistered email, the system displays an error message and prompts the user to try again.  E2. **Expired reset link/code**: If the user tries to use an expired link or code, the system displays an error and prompts the user to restart the process. | | |
| Priority: | High | | |
| Frequency of Use: | This feature will be used occasionally, as students or instructors might forget their passwords once every few months. | | |
| Business Rules: | 1. The password must meet the institution’s security requirements (e.g., length, special characters). | | |
| Other Information: | Password recovery is a critical security feature to ensure users can regain access to the platform. | | |
| Assumptions: | The student/instructor has access to the email/phone associated with their account | | |

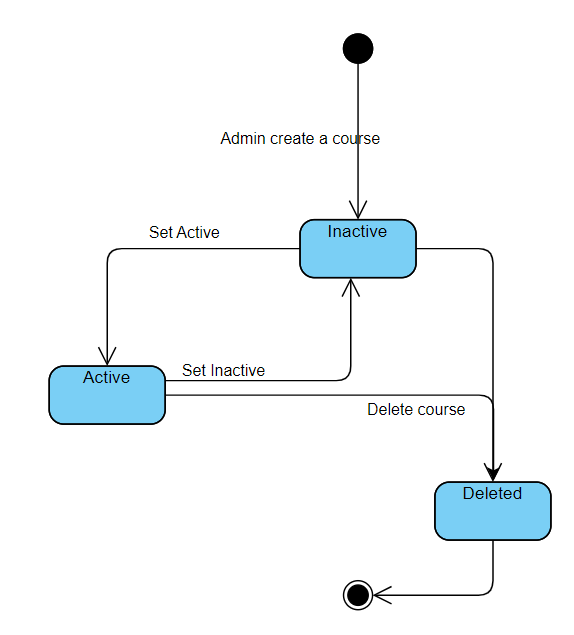


# Data Requirements

## Logical Data Model

**

*Entity relationship diagram*



*State Diagram for a couse object*

## Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Element** | **Description** | **Composition/Data Type** | **Length** | **Values** |
| Admin |  |  |  |  |
| id | Unique identifier for each admin | int | 4 | Auto-incremented, PK |
| name | Name of the admin | nvarchar | 100 | Alphanumeric characters |
| email | Email address of the admin | varchar | 200 | Valid email format |
| phone | Contact number of the admin | nvarchar | 12 | Numeric values, may include country code if specified |
| password | Password for admin account access | nvarchar | 100 | Encrypted string |
| status | Status of admin account (e.g., active/inactive) | int | 4 | 1 = Active, 0 = Inactive |
| avatar | URL or path to admin profile picture | nvarchar | 500 | URL to avatar image |
| **Assignments** |  |  |  |  |
| AssignmentID | Unique identifier for each assignment | int | 4 | Auto-incremented, PK |
| Title | Title of the assignment | nvarchar | 100 | Alphanumeric characters |
| Description | Detailed description of the assignment | nvarchar (max) | Unlimited | Free text |
| DueDate | Due date for the assignment | datetime | 8 | Date format |
| ClassID | Identifier for the class associated with the assignment | int | 4 | FK from Classes table |
| Type | Type of assignment (e.g., homework, quiz) | int | 4 | 1 = Homework, 2 = Quiz, etc. |
| Status | Status of assignment (e.g., published, unpublished) | int | 4 | 1 = Published, 0 = Unpublished |
| **Classes** |  |  |  |  |
| ClassID | Unique identifier for each class | int | 4 | Auto-incremented, PK |
| ClassName | Name of the class | varchar | 100 | Alphanumeric characters |
| Status | Status of the class (e.g., active/inactive) | int | 4 | 1 = Active, 0 = Inactive |
| TeacherID | Identifier for the teacher assigned to the class | int | 4 | FK from Teachers table |
| CourseID | Identifier for the course associated with the class | int | 4 | FK from Courses table |
| SemesterID | Identifier for the semester in which the class is held | int | 4 | FK from Semesters table |
| **Courses** |  |  |  |  |
| CourseID | Unique identifier for each course | int | 4 | Auto-incremented, PK |
| CourseCode | Code for the course | nvarchar | 100 | Alphanumeric string |
| CourseName | Name of the course | varchar | 100 | Alphanumeric characters |
| Status | Status of the course (e.g., active/inactive) | int | 4 | 1 = Active, 0 = Inactive |
| **Lessons** |  |  |  |  |
| lesson\_id | Unique identifier for each lesson | int | 4 | Auto-incremented, PK |
| course\_id | Identifier for the course associated with the lesson | int | 4 | FK from Courses table |
| title | Title of the lesson | nvarchar | 255 | Alphanumeric characters |
| video\_url | URL link to the video for the lesson | nvarchar | 255 | Valid URL format |
| docs\_link | URL link to the documentation for the lesson | nvarchar | 255 | Valid URL format |
| description | Detailed description of the lesson | nvarchar (max) | Unlimited | Free text |
| status | Status of the lesson (e.g., published/unpublished) | nvarchar | 50 | Published,  Unpublished |
| **Semesters** |  |  |  |  |
| SemesterID | Unique identifier for each semester | int | 4 | Auto-incremented, PK |
| SemesterName | Name of the semester | varchar | 100 | Alphanumeric characters |
| Year | Year associated with the semester | int | 4 | Year in format YYYY |
| Status | Status of the semester (e.g., active/inactive) | int | 4 | 1 = Active, 0 = Inactive |
| **StudentClass** |  |  |  |  |
| studentId | Unique identifier for each student | int | 4 |  |
| classId | Unique identifier for each class | int | 4 |  |
| joinAt | Date and time the student joined the class | datetime | 8 | Date and time format |
| status | Status of the student's enrollment in the class | int | 4 | 1 = Enrolled, 0 = Dropped |
| **StudentLesson** |  |  |  |  |
| student\_id | Unique identifier for each student | int | 4 |  |
| lession\_id | Unique identifier for each lesson | int | 4 |  |
| completion\_date | Date of lesson completion | date | 3 | Date format |
| is\_completed | Indicates if the lesson is completed | bit | 1 | 1 = Completed, 0 = Not Completed |
| **Students** |  |  |  |  |
| StudentID | Unique identifier for each student | int | 4 | Auto-incremented, PK |
| Name | Name of the student | nvarchar | 100 | Alphanumeric characters |
| Email | Email address of the student | varchar | 100 | Valid email format |
| Phone | Contact number of the student | varchar | 20 | Numeric values, may include country code if specified |
| Status | Status of the student account | int | 4 | 1 = Active, 0 = Inactive |
| **Submissions** |  |  |  |  |
| SubmissionID | Unique identifier for each submission | int | 4 | Auto-incremented, PK |
| SubmissionContent | Content of the student submission | nvarchar | 1000 | Free text |
| StudentID | Identifier for the student submitting the work | int | 4 | FK from Students table |
| AssignmentID | Identifier for the related assignment | int | 4 | FK from Assignments table |
| SubmissionDate | Date and time of submission | datetime | 8 | Date and time format |
| Grade | Grade given for the submission | float | 8 | Numeric value, range defined by institution |
| **Teachers** |  |  |  |  |
| TeacherID | Unique identifier for each teacher | int | 4 | Auto-incremented, PK |
| Name | Name of the teacher | nvarchar | 100 | Alphanumeric characters |
| Email | Email address of the teacher | varchar | 100 | Valid email format |
| Phone | Contact number of the teacher | varchar | 20 | Numeric values, may include country code if specified |
| Status | Status of teacher account | int | 4 | 1 = Active, 0 = Inactive |
| Avatar | URL or path to teacher profile picture | nvarchar | 500 | URL to avatar image |

## Reports

|  |  |
| --- | --- |
| **Report ID** | **OCMS\_RP\_1** |
| Report Title | Course Progress |
| Report Purpose | Students want to see their enrolled courses ‘progress so that they can adjust their timetable and effort to meet the courses’ objectives. |
| Priority | Medium |
| Report Users | Students |
| Data sources | Database of the courses for each student. |
| Frequency and Disposition | Report is generated on demand by a student. |
| Latency | Complete report must be displayed to Student within 4 seconds after it is requested. |
| Visual Layout | Landscape mode |
| Header and Footer | Report header shall contain the report title, Student’s name, and date range specified.  If printed, report footer shall show the page number. |
| Report Body | Fields shown:   * Date Enrolled * Progress * Last learned time |
| End-of-Report Indicator | None |
| Interactivity | Student can drill down to see progress of each lessons in the course. |
| Security Access Restrictions | A student only receive his/her own tracking course progress report. |

## Data Acquisition, Integrity, Retention, and Disposal

**DI-1:** The OCMS shall ensure that all data entries (e.g., student records, course details, assignments) are validated at the time of input to maintain data integrity. Input fields will include appropriate data types, constraints, and validation checks (e.g., email format validation, and required fields).

**DI-2:** The system shall enforce referential integrity between related tables (e.g., students and classes, assignments and submissions) to ensure that relationships remain consistent and valid. Foreign key constraints will be used to maintain these relationships.

**DI-3:** The OCMS shall implement transaction management to ensure that all operations, such as Create, Read, Update, and Delete (CRUD), are atomic, consistent, isolated, and durable (ACID properties). This will prevent data corruption and maintain integrity during concurrent transactions.

**DI-4:** The OCMS shall conduct regular audits and integrity checks of the database to verify the accuracy and consistency of data stored in the system. Any discrepancies identified will trigger alerts for investigation and resolution.

# External Interface Requirements

## User Interfaces

**Description**:

* The system will provide a **web-based interface** with a consistent layout for easy navigation.
* **JSP pages** will serve as the primary interface for the user, while **Servlets** will handle backend processing.
* **Common UI elements**:
  + Navigation bar (links to home, course catalog, login)
  + Standard buttons (submit, cancel, help)
  + Error messages displayed inline with specific form fields or as pop-ups.
* **GUI Standards**:
  + A CSS framework (such as Bootstrap) will ensure responsiveness.
  + Each JSP page will extend a layout file for consistency.
* **Sample JSP Pages**:
  + index.jsp - Home page with navigation to other sections.
  + courseCatalog.jsp - Displays a list of available courses with descriptions.
  + login.jsp - Login page for users to authenticate.

**Example Files**:

* **header.jsp** and **footer.jsp** will be reusable components across JSP pages.
* **Error handling** for login, enrollment, or form submissions will be standardized.

## Software Interfaces

**System-to-System Communication**:

* **Database Connectivity**: Use **JDBC** to connect to a relational database (e.g., MySQL).
  + Servlet controllers will handle data processing and retrieval, while JSPs handle display.
* **Libraries**:
  + Use external libraries like **Jakarta Servlet API** and **JSTL (JSP Standard Tag Library)**.
  + Possible integration with **RESTful Web Services** for external data sources (e.g., fetching educational resources).
* **Format and Content**:
  + Data transfer between JSPs and Servlets via **request parameters** and **session attributes**.
  + JSON format may be used for AJAX requests (e.g., for search functionality).

## Hardware Interfaces

**Server Requirements:**

* **Java EE application server** (such as Apache Tomcat) to host JSP and Servlet components.
* **Database server** for storage of course data, user profiles, etc.

**Device Compatibility:**

* The interface should support various devices, especially mobile and desktop views.

## Communications Interfaces

**Network Protocols**:

* **HTTP/HTTPS**: All data communication will occur over HTTP/S, ensuring secure data transmission.
* **Email**: The system will use JavaMail API to send notifications (e.g., registration confirmations).
* **Network Constraints**:
  + Ensure fast response times for requests to enhance the user experience.

# Quality Attributes

## Usability

**Key Aspects**:

* Consistent navigation and simplified workflows.
* Use of tooltips, on-page help, and error messages for ease of use.
* Keyboard shortcuts for common actions like submit, cancel, and reset.

## Performance

**Specifics**:

* Optimize database queries to ensure efficient data retrieval.
* Cache frequently requested pages (like the course catalog) where possible.

## Security

**Security Measures**:

* **Authentication**: Login via session management (using HTTPS for secure connections).
* **Authorization**: Role-based access to different JSP pages (e.g., students, instructors, admins).
* **Data Validation**: Validate form inputs on both client and server sides.
* **CSRF and XSS Protection**: Use Java EE's built-in protection mechanisms and JSP tags to escape input/output.

## Safety

* **Error Handling**: Redirect users to an error page or display messages if a critical failure occurs (e.g., database outage).
* **Backups and Failovers**: Regular database backups to prevent data loss.

# Internationalization and Localization Requirements

* **Language Support**: Design JSPs to support multiple languages via resource bundles.
* **Formatting Standards**: Use Java’s Locale class to format dates, times, and currencies according to the user’s region.

# Other Requirements

**Compliance**

* **Data Privacy**: Ensure compliance with regulations like GDPR for European users.
* **Audit Trails**: Log all major user actions for audit purposes (e.g., logins, enrollments).

**Installation and Setup**

* **Setup Script**: A script to configure the database schema and populate initial data (e.g., course catalog).
* **Deployment**: Detailed documentation for deploying the system on a Tomcat server with a MySQL database.

**Appendix A: Glossary**

* **JSP** - JavaServer Pages, used for dynamic web pages in Java.
* **Servlet** - Java class that handles HTTP requests and generates responses.
* **DAO** - Data Access Object, pattern for abstracting data interactions.
* **MVC** - Model-View-Controller, design pattern used here with JSP as View, Servlets as Controller, and Java classes as Model.

**Appendix B: Analysis Models**

* **Data Flow Diagrams**: To depict user requests from JSPs and the corresponding data flow through Servlets and DAOs.
* **Entity-Relationship Diagram**: Showing relationships between courses, users, and enrollments in the database.