Software Requirements Specification

for

<Online Course Management System>

**Version 1.0 approved**

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

**<Group 2>**

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

**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

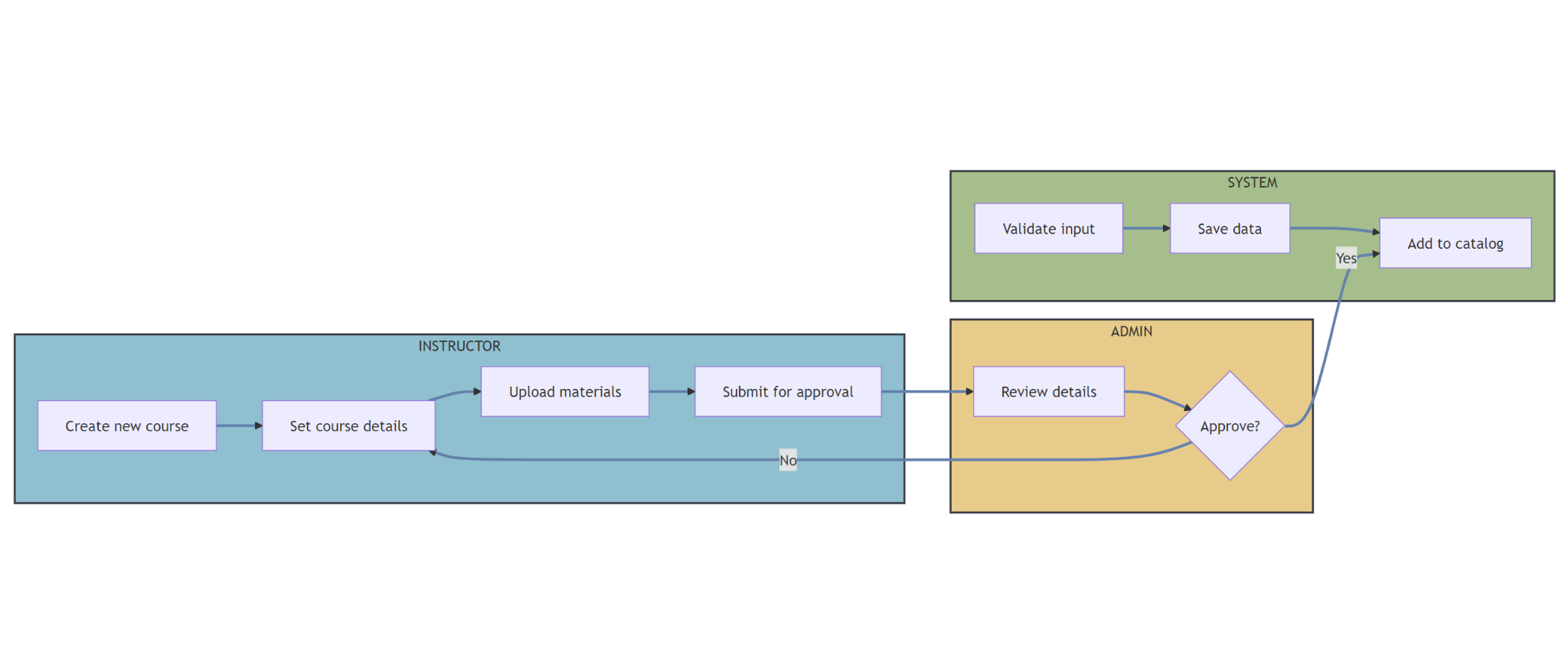
**

## User Classes and Characteristics

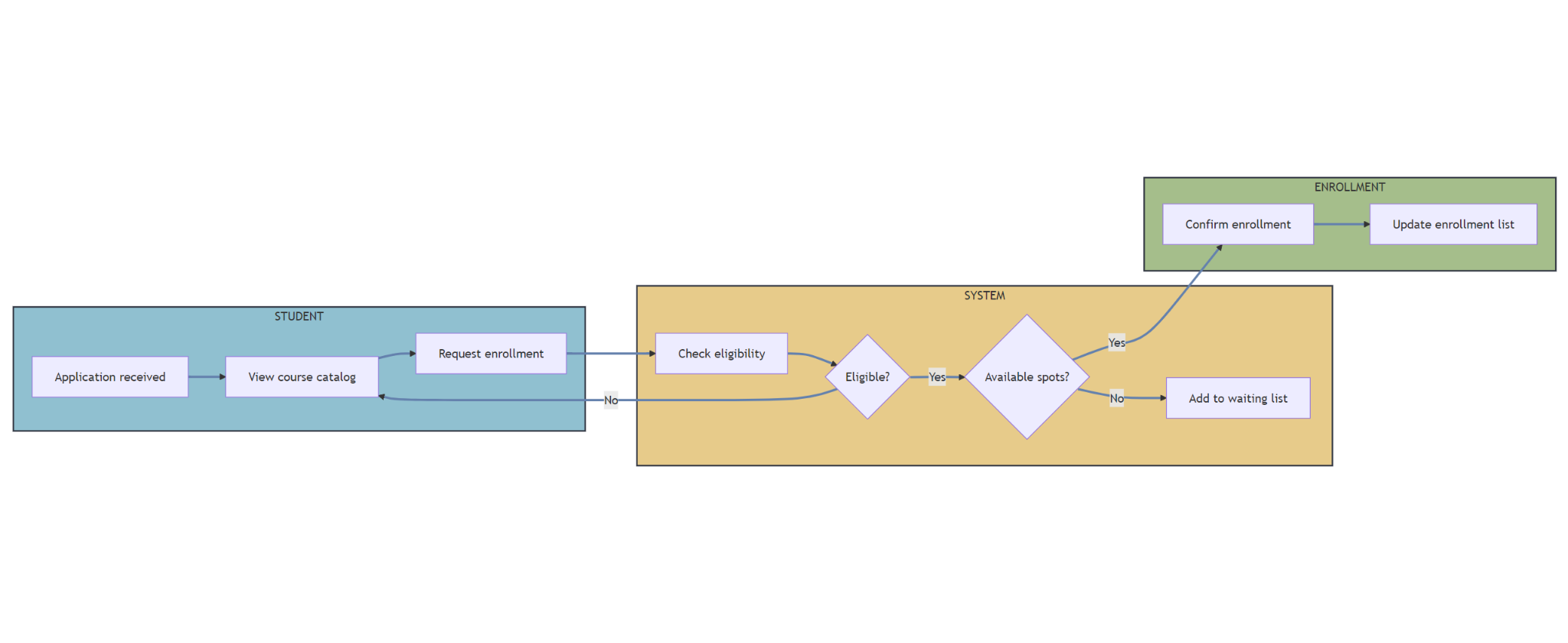
|  |  |
| --- | --- |
| *Students (favored)* | *Students are the primary users of the system. They enroll in courses, participate in learning activities, and track their academic progress.*  *There could be* ***thousands of students*** *using the system, depending on the size of the university or institution. This user class will likely grow as the institution expands or as more courses are added.* |
| *Instructors* | *Instructors are responsible for creating, managing, and delivering course content, grading students, and facilitating discussions. They have greater access rights than students to manage their respective courses. There are possibly up to hundreds of instructor. Some instructors may require initial training on how to manage courses and gradebooks.* |
| *Administrators* | *Administrators oversee the entire system, manage user roles (students, instructors, other admins), and maintain the course catalogue. They ensure the smooth operation of the system and resolve technical issues when necessary. There are possibly 10-20 admins depending on the institution size.* |

## Main Workflows

### Course Creation Process



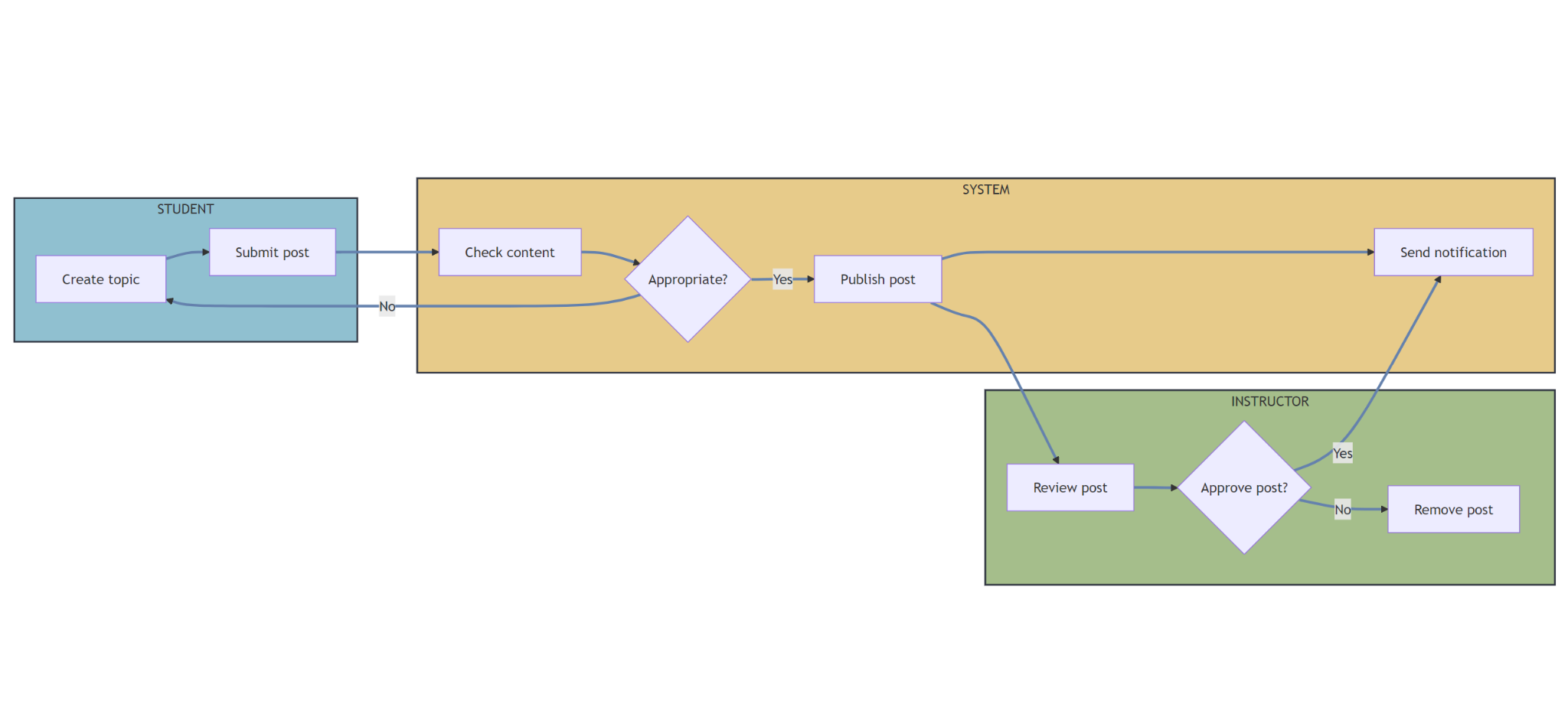
### Course Enrollment Process



* + 1. **Assignment Process**

****

* + 1. **Discussion Management Process**

****

## 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.

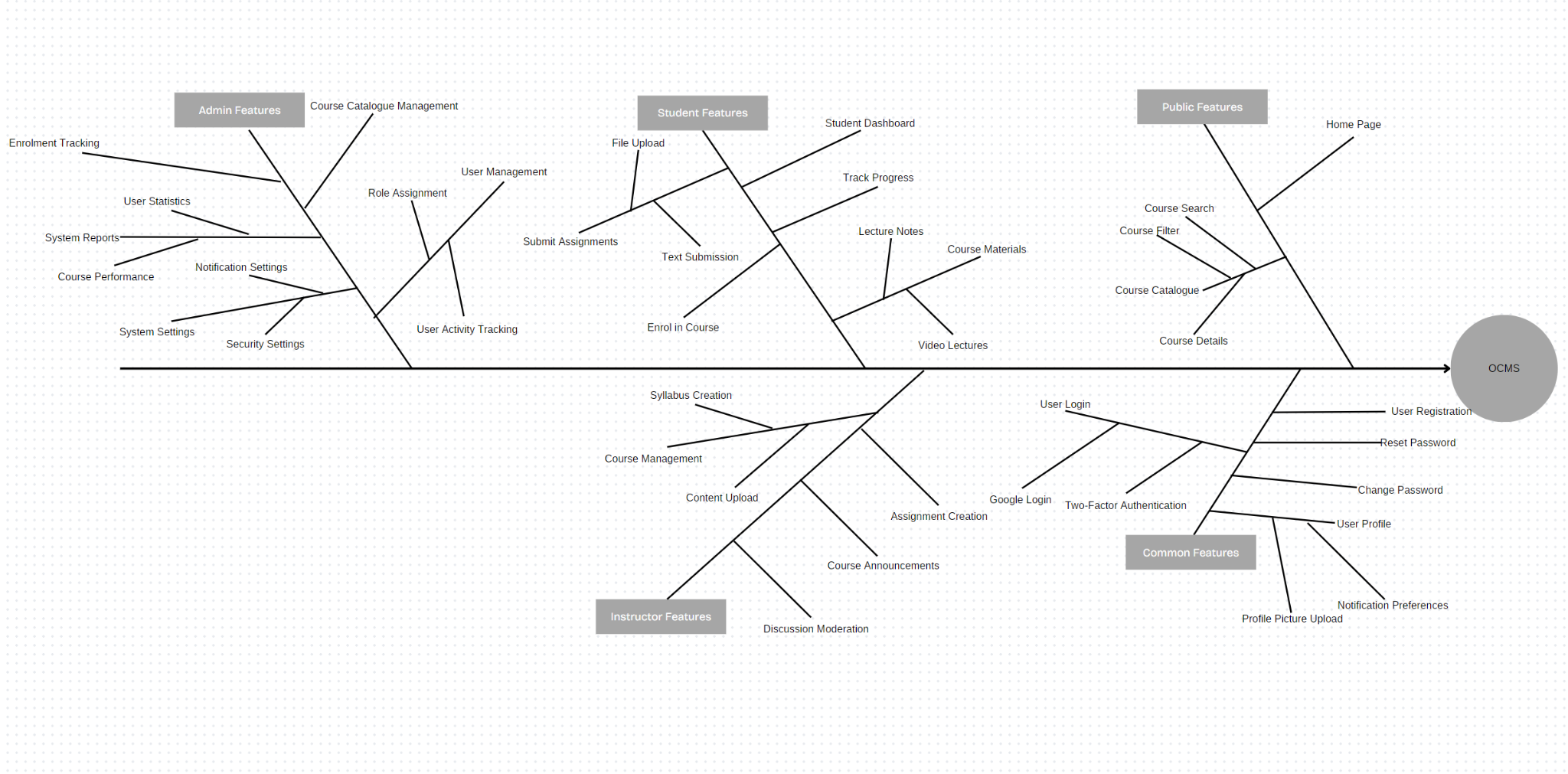
## Assumptions and Dependencies

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

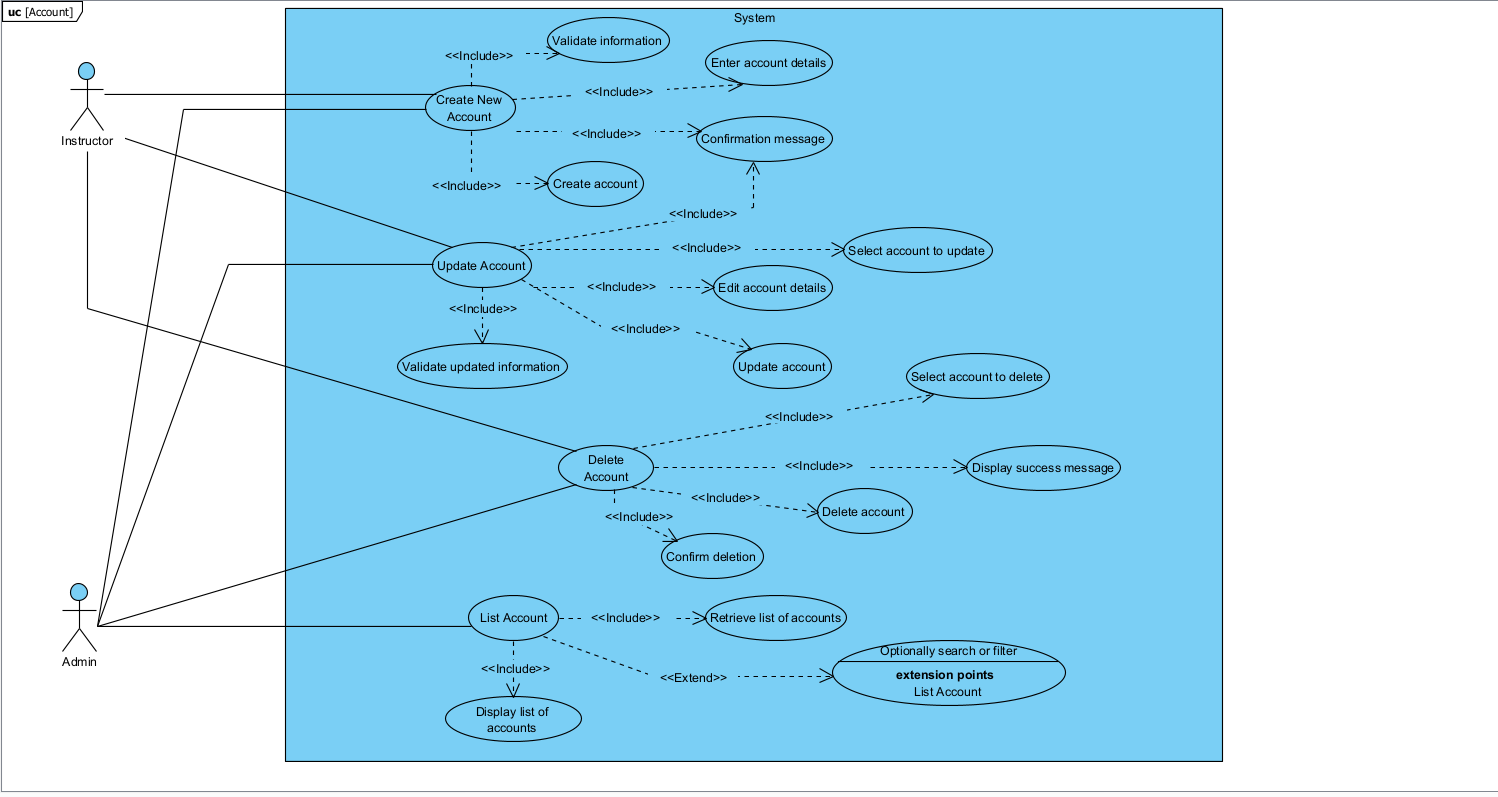
**User Access**: Assumes students and instructors will access the system via university networks with standard web browsers.

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

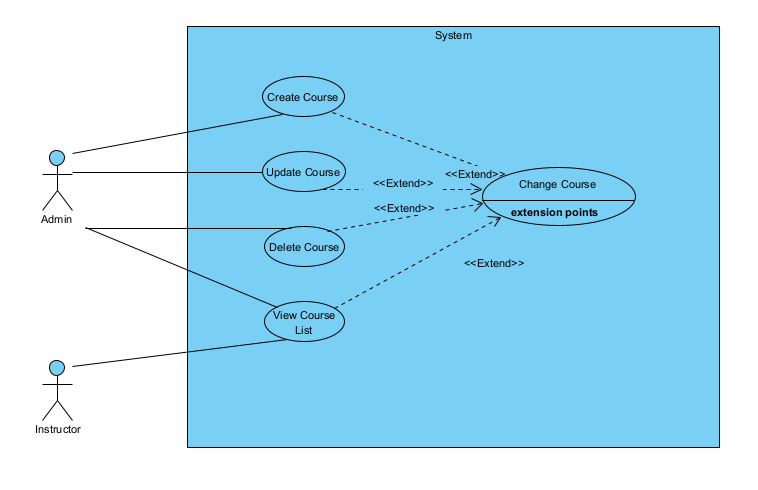
# System Features



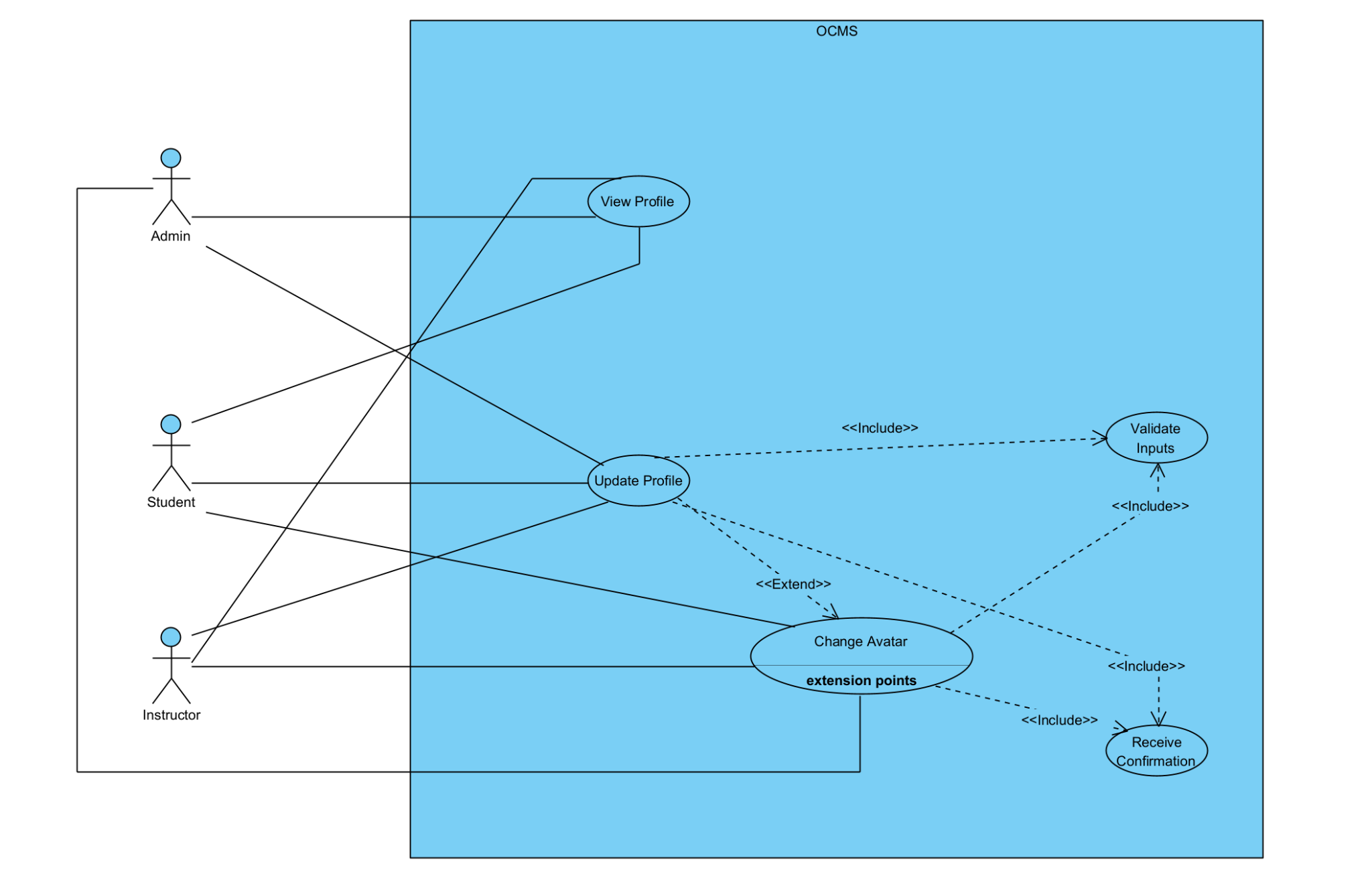
a)Manage Account



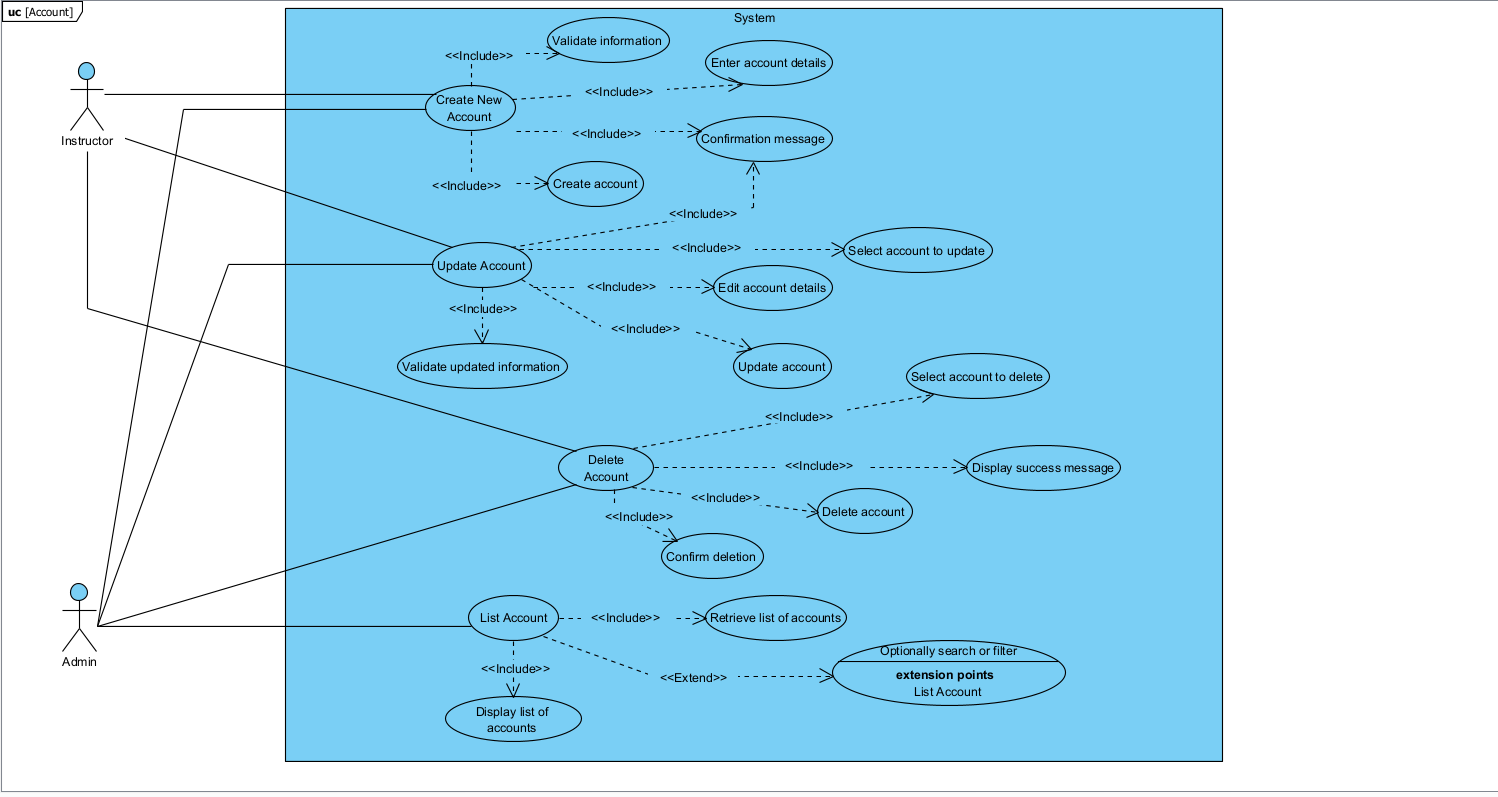
b) Manage Course



c) Update and View Profile



## System Feature Manage Account:



### Description:

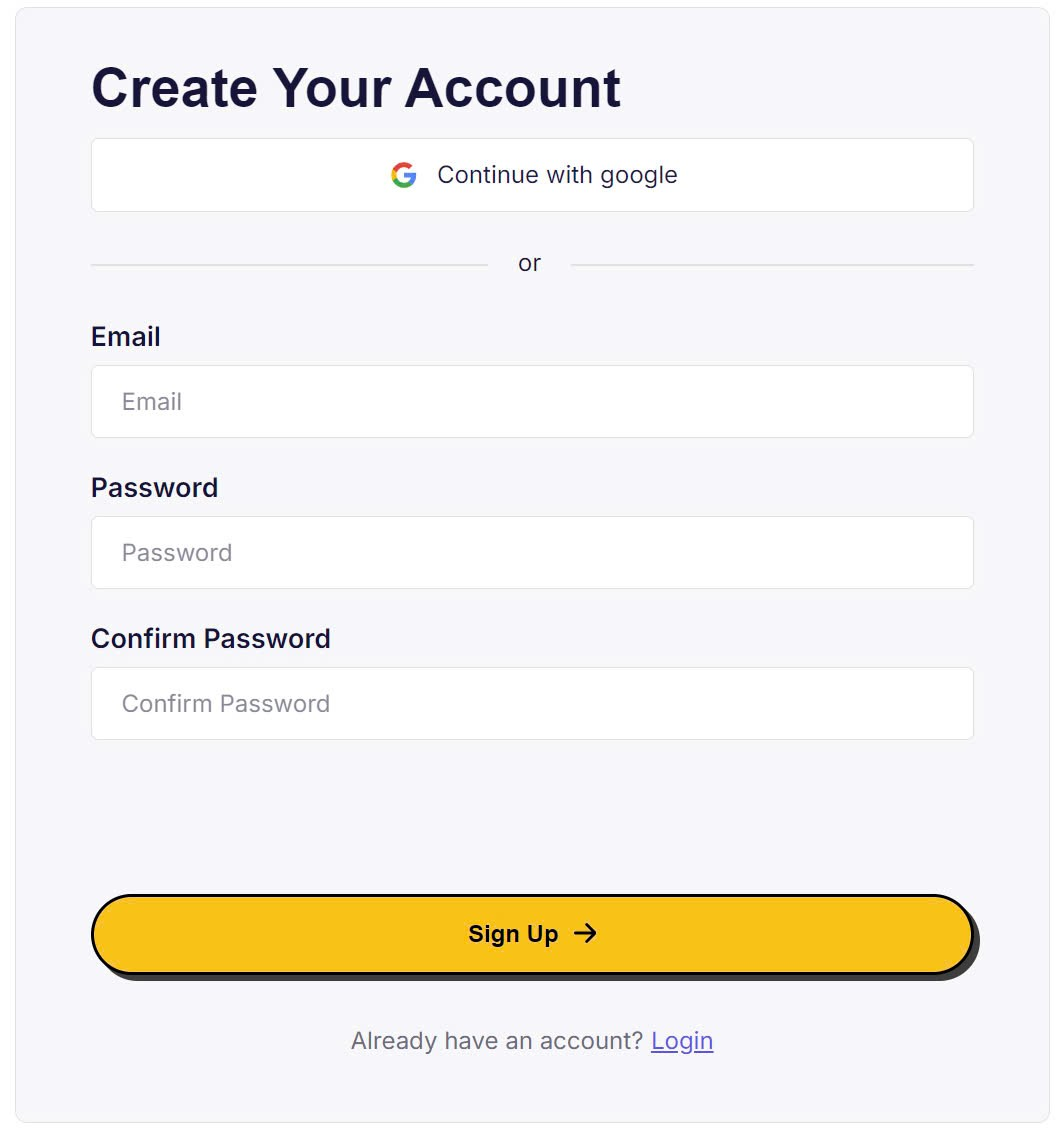
This feature allows administrators to manage user accounts, including creating, updating, deleting accounts, and displaying account lists to students, faculty, and other staff. The system ensures that administrators can efficiently handle user information, assign appropriate roles, and update account details.

### Stimulus/Response Sequences:

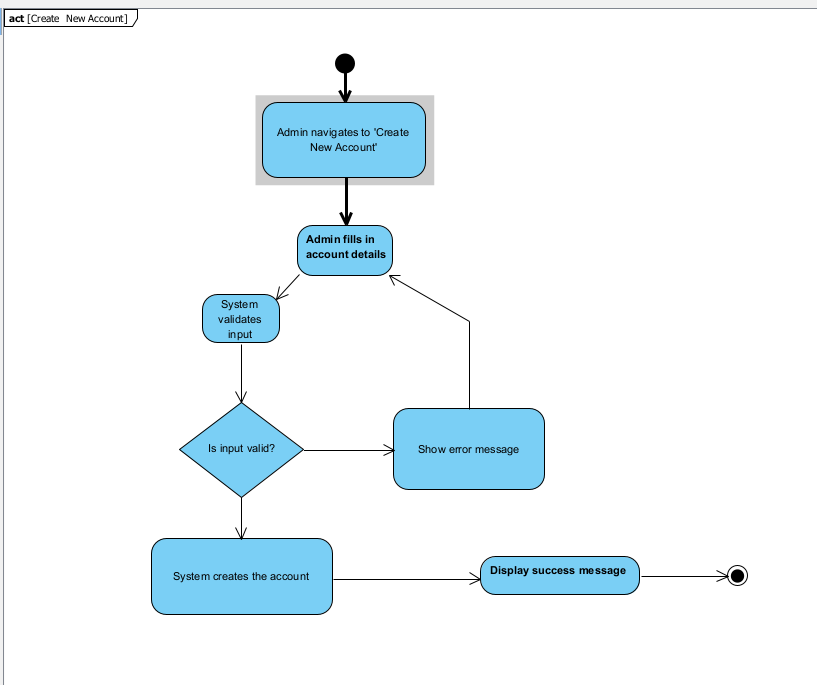
List all event and FAP system must response with the event. Should write in table formate

|  |  |
| --- | --- |
| Event | System Response |
| Admin requests to create a new user account | Display a form to create a new account with input fields for username, role (student/instructor/admin) |
| Admin submits new user account details | Validate the input data, store the account in the database, and provide a confirmation message. |
| Admin requests to update user account information | Display current user account details with editable fields (name, role, status). |
| Admin submits updated account details | Validate changes, update the account information in the database, and show a success confirmation. |
| Admin requests to delete a user account | Display a confirmation prompt before proceeding with deletion. |
| Admin confirms account deletion | Remove the user account from the database and show a success message. |

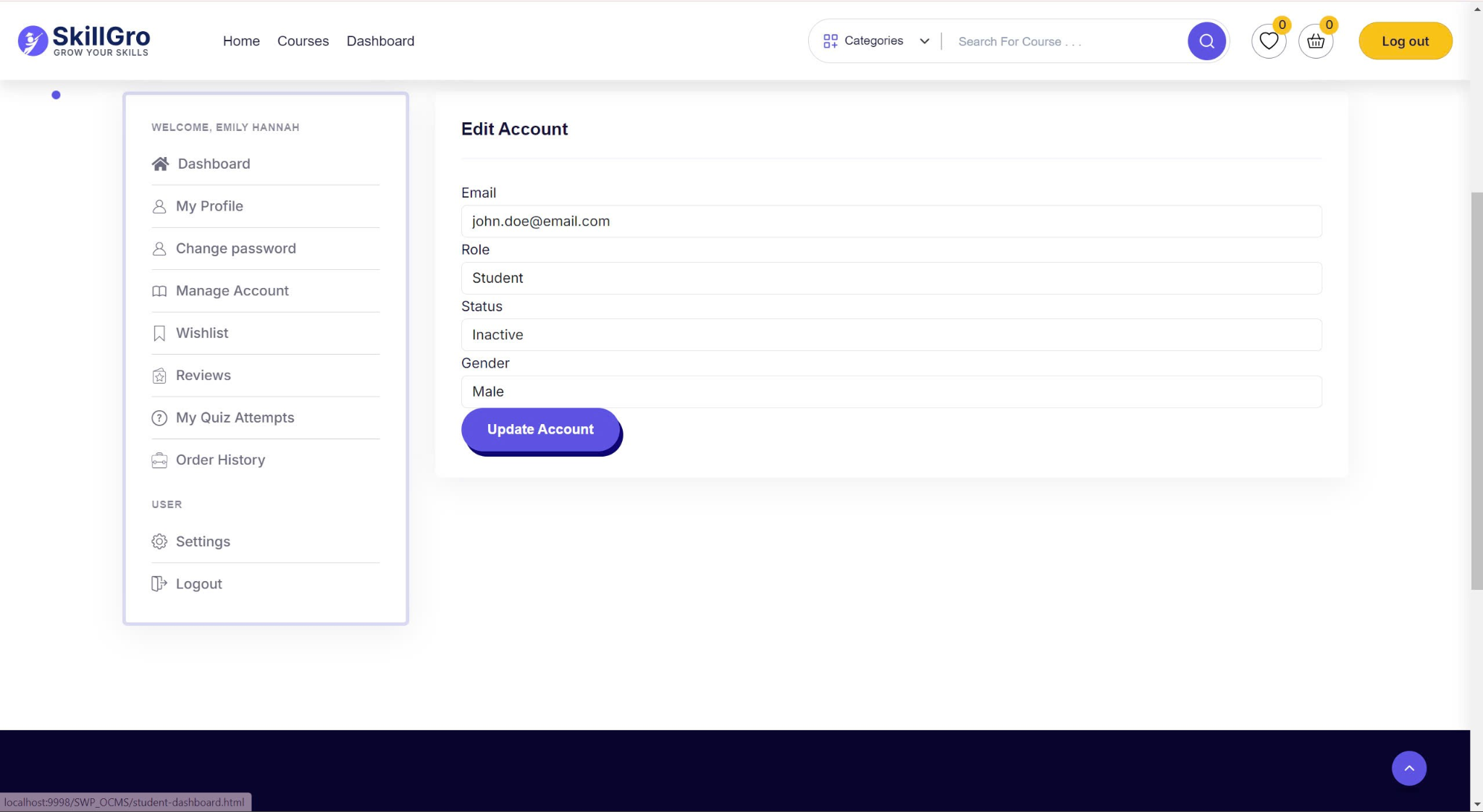
### Functional Requirements Create New Account:



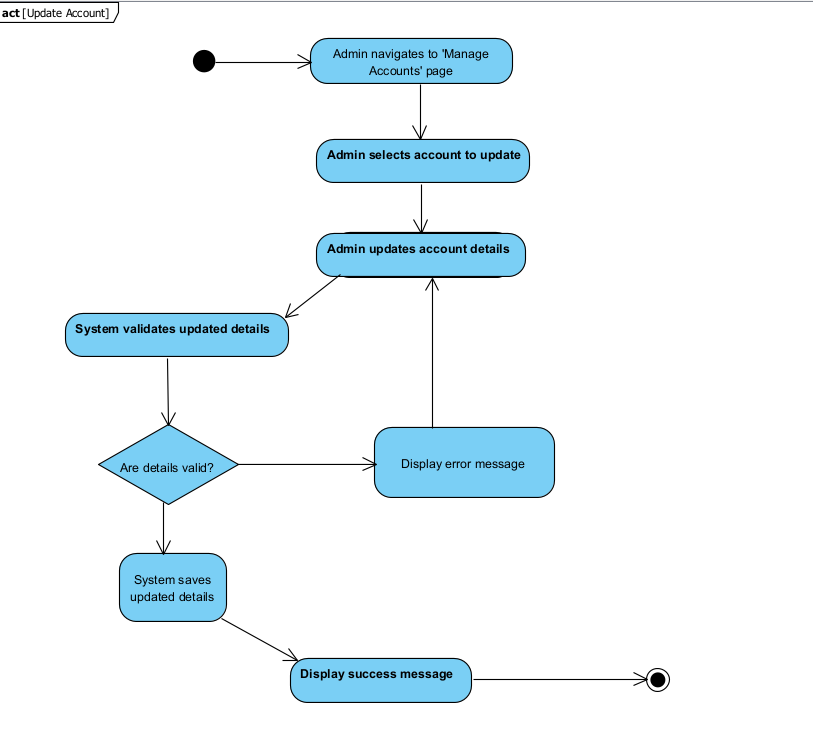
|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Create New Account | | |
| Created By: | SyTD | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin selects the option to create a new account from the system. | | |
| Description: | This use case describes the process an admin uses to create a new account for a user (student, instructor, or other). The admin will fill in details like username, role, and email, and the system will create the account with the provided information. | | |
| Preconditions: | The admin must be logged in to the system. The system must be connected to the database where user data is stored. | | |
| Postconditions: | A new account is created and saved in the system. The new user can log in with the credentials provided. | | |
| Normal Flow: | 1. The admin navigates to the "Create New Account" form.  2. The admin enters details such as username, email, password, and role.  3. The system validates the input.  4. If valid, the system saves the new user account to the database.  5. The system displays a success message. | | |
| Alternative Flows: | If any required information is missing or invalid, the system shows an error message, and the admin must correct the input. | | |
| Exceptions: | - The system is offline or the database is not accessible.  - The email or username is already taken.  - Password requirements are not met. | | |
| Priority: | High | | |
| Frequency of Use: | As needed, when new users need to be registered by admin. | | |
| Business Rules: | - Usernames must be unique.  - Passwords must meet security requirements (e.g., minimum length, complexity).  - Admins can assign roles (e.g., student, instructor, admin). | | |
| Other Information: | None | | |
| Assumptions: | The admin has the necessary permissions to create accounts. The system is properly configured to handle user roles and account creation. | | |



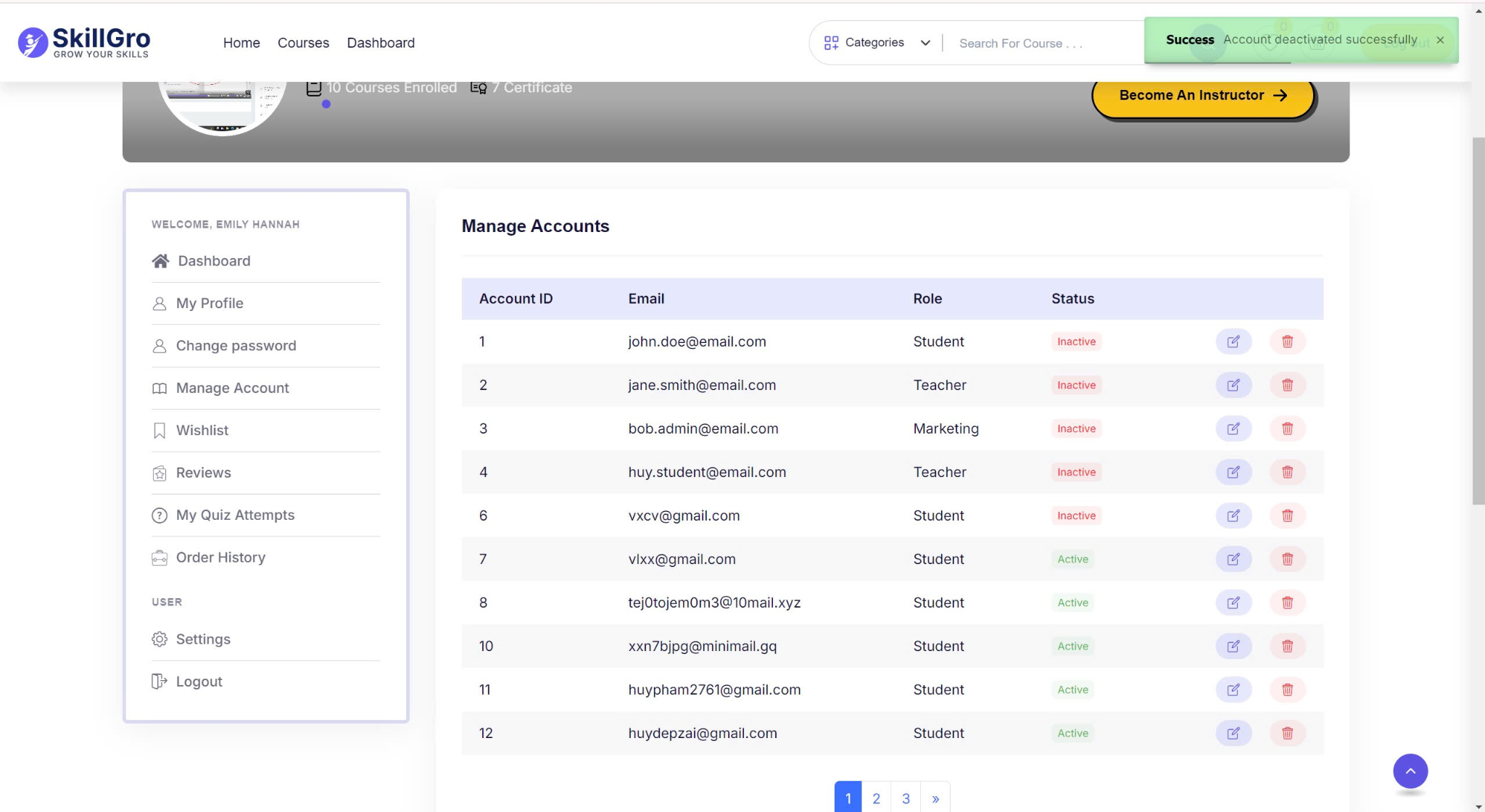
### Function Update Account



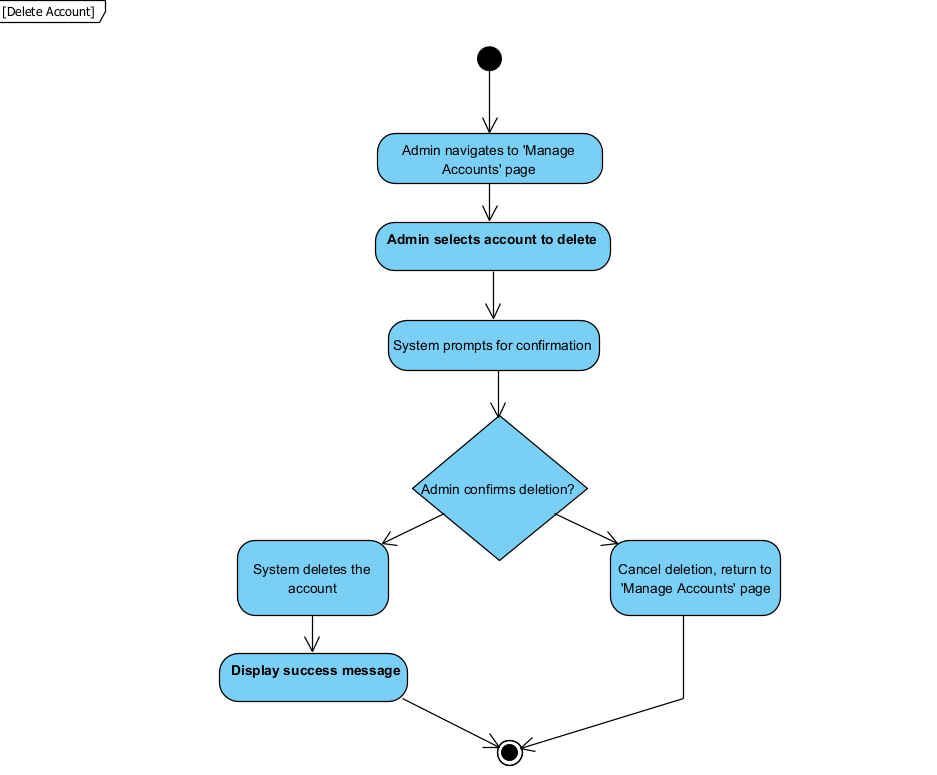
|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Update Account | | |
| Created By: | SyTD | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin selects the option to update an existing account. | | |
| Description: | This use case describes the process for updating an existing account's details such as name, email, role, or password. The admin selects a user account, makes the necessary changes, and submits the update to the system. | | |
| Preconditions: | The admin must be logged in to the system and have the necessary permissions to update accounts. The account to be updated must exist in the system. | | |
| Postconditions: | The selected account is updated with the new details in the system. The changes are saved to the database, and the updated details are available for the user on their next login. | | |
| Normal Flow: | 1. The admin navigates to the "Manage Accounts" page.  2. The admin selects the account to update.  3. The admin makes changes to the account details (e.g., name, role, password).  4. The system validates the updated details.  5. If valid, the system saves the changes to the database.  6. The system displays a success message confirming the update. | | |
| Alternative Flows: | If the update details are invalid (e.g., password not meeting requirements), the system displays an error message and prompts the admin to correct the input. | | |
| Exceptions: | - The system is offline or cannot access the database.  - The account does not exist or is locked.  - The new email is already associated with another account. | | |
| Priority: | Medium | | |
| Frequency of Use: | As needed, when account details require updating. | | |
| Business Rules: | - Usernames and emails must remain unique.  - Passwords must comply with security policies (e.g., minimum length, complexity).  - Only authorized users can update accounts.  - Certain fields (e.g., role) may only be editable by admins. | | |
| Other Information: | None | | |
| Assumptions: | The admin has the necessary permissions to update accounts. The system is online and connected to the database. | | |



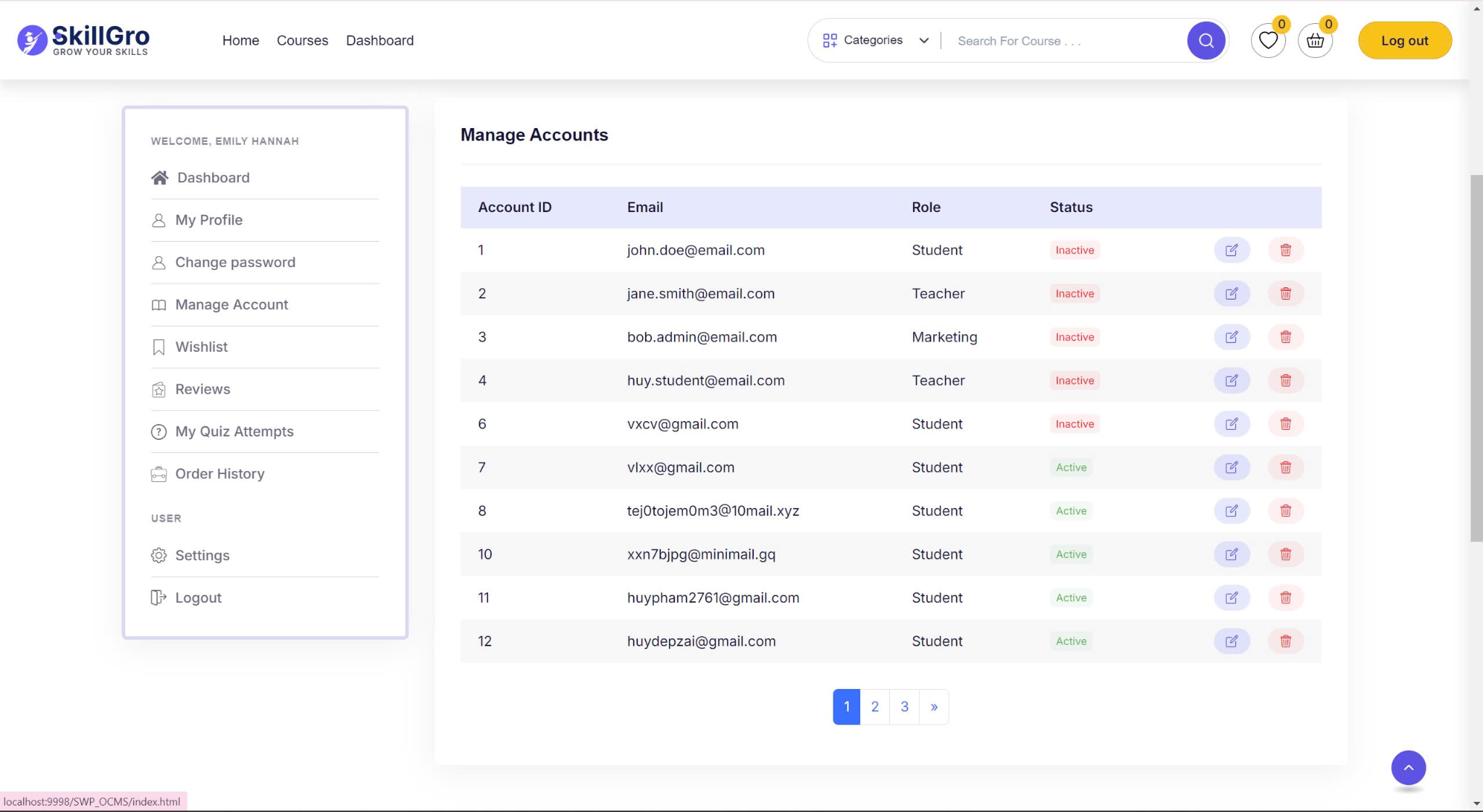
**3.1.5** **Function Delete Account:**

****

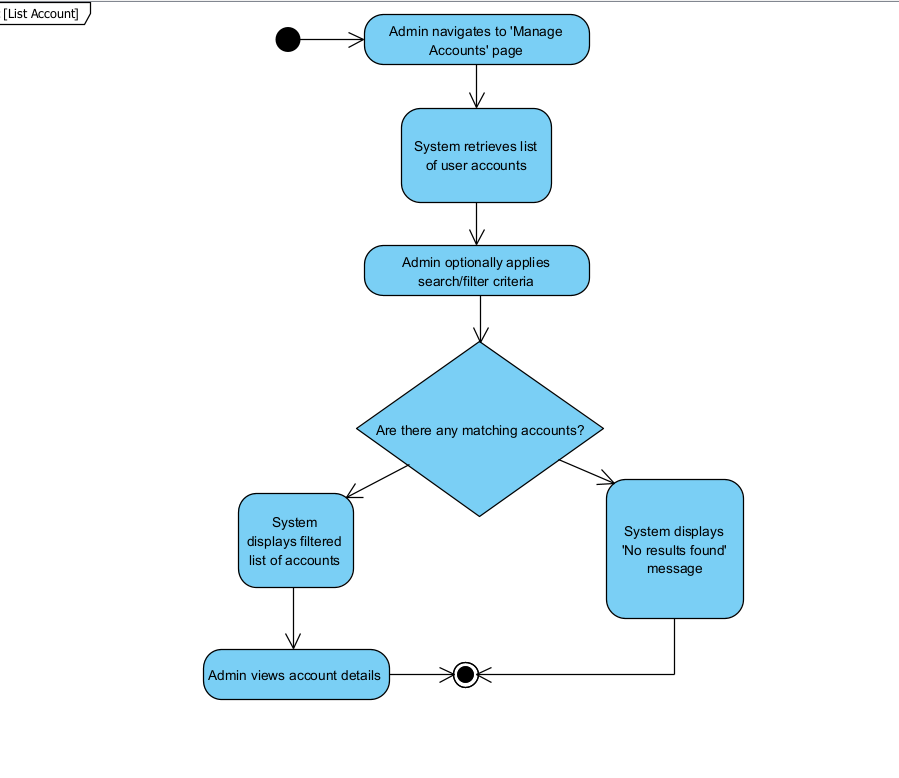
|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Delete Account | | |
| Created By: | SyTD | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin selects the option to delete an existing account from the system. | | |
| Description: | This use case describes the process of deleting an account from the system. The admin selects a user account, confirms the deletion, and the account is removed from the system along with associated data. | | |
| Preconditions: | The admin must be logged in with appropriate permissions to delete accounts. The account to be deleted must exist in the system. | | |
| Postconditions: | The selected account and its associated data are permanently removed from the system. | | |
| Normal Flow: | 1. The admin navigates to the "Manage Accounts" page.  2. The admin selects the account to delete.  3. The system prompts the admin to confirm the deletion.  4. The admin confirms the deletion.  5. The system deletes the account from the database.  6. The system displays a success message confirming the deletion. | | |
| Alternative Flows: | If the admin cancels the deletion confirmation, the account is not deleted, and the system returns to the "Manage Accounts" page. | | |
| Exceptions: | - The system is offline or cannot access the database.  - The account cannot be found in the system.  - The account is associated with ongoing tasks (e.g., courses, assignments) that prevent immediate deletion. | | |
| Priority: | High | | |
| Frequency of Use: | As needed, when accounts need to be removed from the system. | | |
| Business Rules: | - Only authorized users (e.g., admins) can delete accounts.  - Deleted accounts cannot be recovered.  - Associated data (e.g., course enrolments, grades) should be permanently removed or archived according to business policy. | | |
| Other Information: | The system may implement soft deletion where data is marked as deleted but not removed from the database for compliance purposes. | | |
| Assumptions: | The admin has sufficient permissions to delete accounts, and the system is functioning properly with access to the database. | | |

****

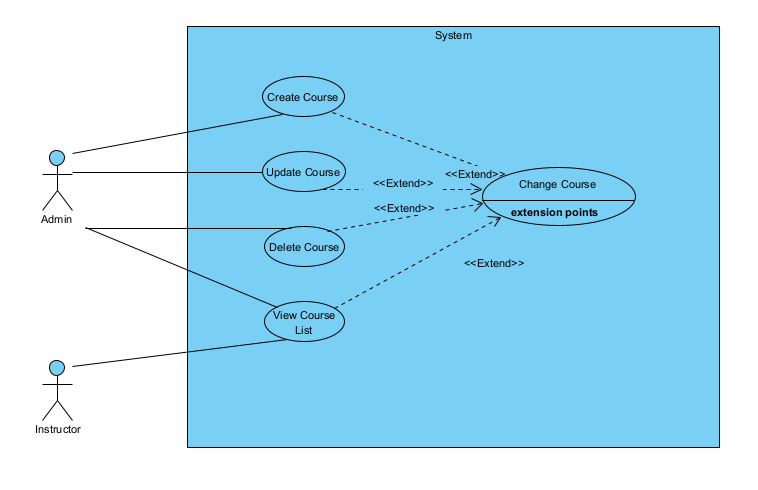
**3.1.6** **Function List Account:**

****

| UC ID and Name: | List Account | | |
| --- | --- | --- | --- |
| Created By: | SyTD | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin selects the option to view a list of all user accounts in the system. | | |
| Description: | This use case describes the process for an admin to view a list of all user accounts, including their basic details. The admin can search for specific users or filter the list based on criteria. | | |
| Preconditions: | The admin must be logged in with the necessary permissions to view user accounts. There must be at least one user account in the system. | | |
| Postconditions: | The system displays a list of user accounts with the relevant details, possibly filtered by the admin’s search criteria. | | |
| Normal Flow: | 1. The admin navigates to the "Manage Accounts" page.  2. The system retrieves the list of user accounts from the database.  3. The admin can optionally apply filters or search for specific users.  4. The system displays the filtered or full list of accounts.  5. The admin can view basic details such as username, email, and role for each account. | | |
| Alternative Flows: | If no accounts match the search criteria, the system displays a message indicating that no results were found. | | |
| Exceptions: | - The system is offline or cannot retrieve data from the database.  - The admin does not have permission to view user accounts. | | |
| Priority: | Medium | | |
| Frequency of Use: | Regularly, when the admin needs to view or manage user accounts. | | |
| Business Rules: | - Only authorized users (e.g., admins) can view the full list of accounts.  - The list must include a search and filter function to help narrow down results.  - Sensitive details (e.g., passwords) must not be displayed. | | |
| Other Information: | Pagination or lazy loading may be implemented for large lists of accounts to improve performance. | | |
| Assumptions: | The system is functioning properly and has access to the database. The admin has sufficient permissions to view user accounts. | | |



## System Feature Manage Course

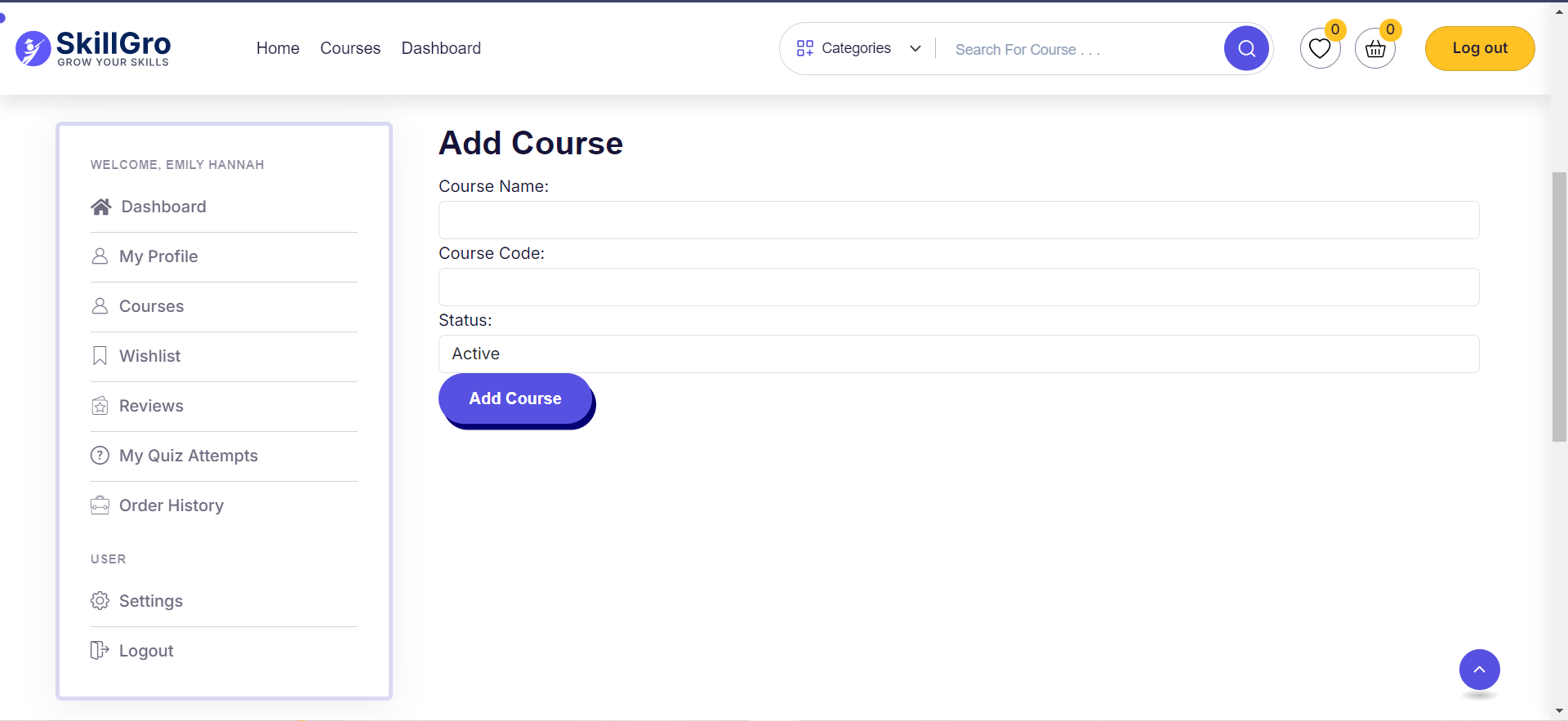


### Description : - This feature allows the admin to manage courses, including creating, updating, and deleting course information from the system. The system ensures that the admin can handle all course details accurately and efficiently.

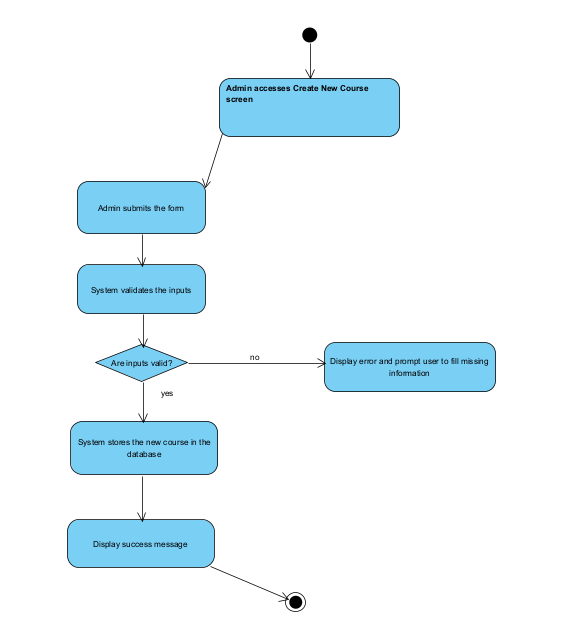
#### Stimulus/Response Sequences:

|  |  |
| --- | --- |
| Event | System Response |
| Admin requests to create a new course | Display form to create a new course with necessary input fields. |
| Admin submits new course details | Validate data, store in the database, and provide a confirmation message. |
| Admin requests to update a course | Display current course details with editable fields. |
| Admin submits updated course data | Validate updates, save changes in the database, and show success confirmation. |

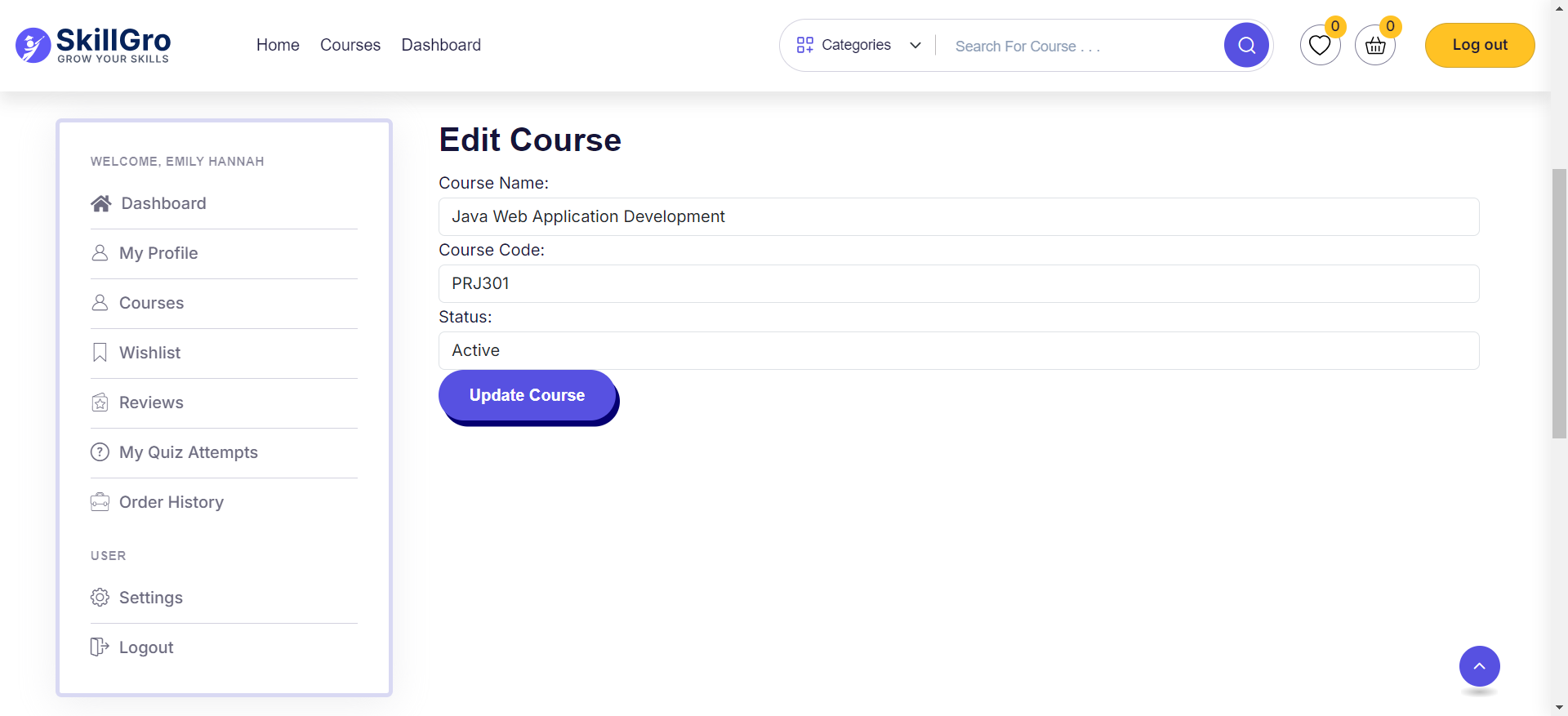
### Function create new course



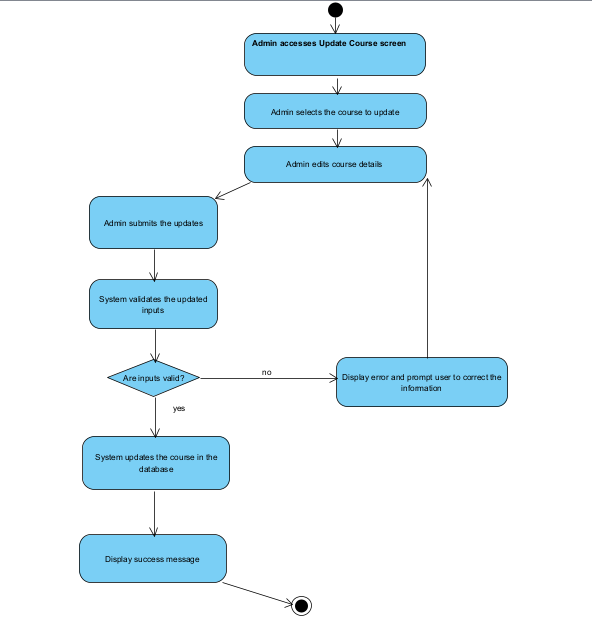
| UC ID and Name: | Create New Course | | |
| --- | --- | --- | --- |
| Created By: | TuanVM | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin initiates the process of adding a new course. | | |
| Description: | This use case describes the process for an admin to create a new course in the system. | | |
| Preconditions: | The admin must have valid access to the system with course management privileges. | | |
| Postconditions: | The system creates and stores the new course information in the database. | | |
| Normal Flow: | 1. Admin accesses the "Create New Course" screen. 2. Admin enters required course details (e.g., course name, description, credits). 3. Admin submits the form. 4. The system validates the inputs. 5. If valid, the system stores the new course in the database and displays a success message. | | |
| Alternative Flows: | Invalid Course Data: If the data entered is invalid, the system displays an error.  Missing Information: If required fields are missing, the system prompts the user to fill them. | | |
| Exceptions: | Database connection failure.  Duplicate course with the same name. | | |
| Priority: | High | | |
| Frequency of Use: | Frequently, as new courses will be added regularly. | | |
| Business Rules: | Email must belong to the FPT University domain.  The admin must ensure that all required fields are filled. | | |
| Other Information: | The form should be intuitive to use. | | |
| Assumptions: | The admin has permission to create accounts. | | |



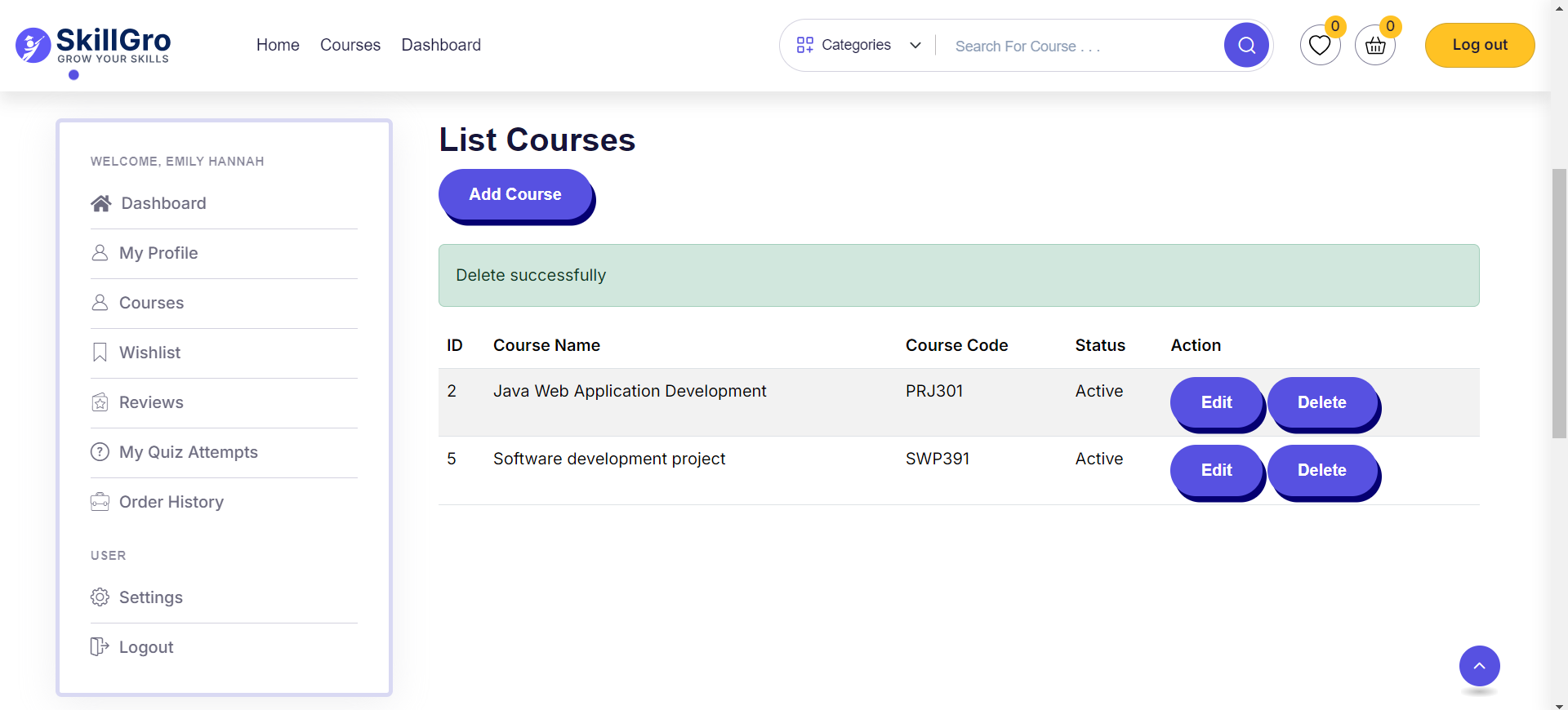
### Function update course



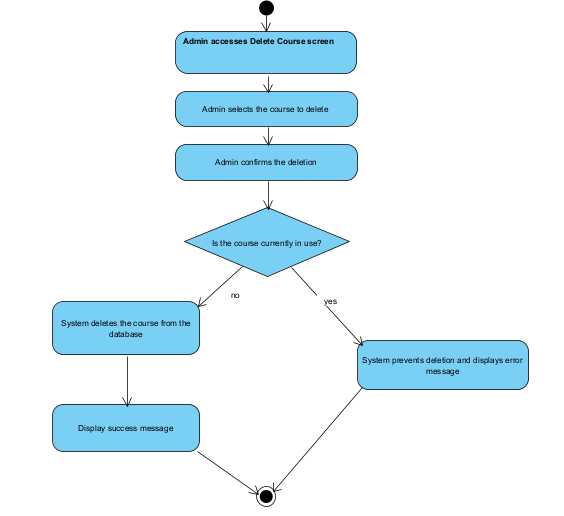
|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC02 - Update Course | | |
| Created By: | TuanVM | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin initiates the process of updating an existing course. | | |
| Description: | This use case describes how an admin can modify the details of an existing course in the system. | | |
| Preconditions: | The course to be updated must already exist in the system. | | |
| Postconditions: | The system updates the course details and stores the changes in the database. | | |
| Normal Flow: | 1. Admin accesses the "Update Course" screen. 2. Admin selects the course to update. 3. Admin edits the required information (e.g., course description, credits). 4. Admin submits the updated information. 5. The system validates the updated data. 6. If valid, the system saves the changes and displays a success message. 7. If valid, the system saves the changes and displays a success message. | | |
| Alternative Flows: | **Invalid Data:** If invalid information is provided, the system prompts the admin to correct it.  **Course Not Found:** If the course does not exist, the system displays an error message. | | |
| Exceptions: | Database connection failure.  Attempt to update a course that no longer exists. | | |
| Priority: | Medium | | |
| Frequency of Use: | Occasional, as updates will be made periodically. | | |
| Business Rules: | Only authorized users (admins) can update courses. | | |
| Other Information: | The update form should pre-fill with the current data. | | |
| Assumptions: | The system contains existing course data. | | |



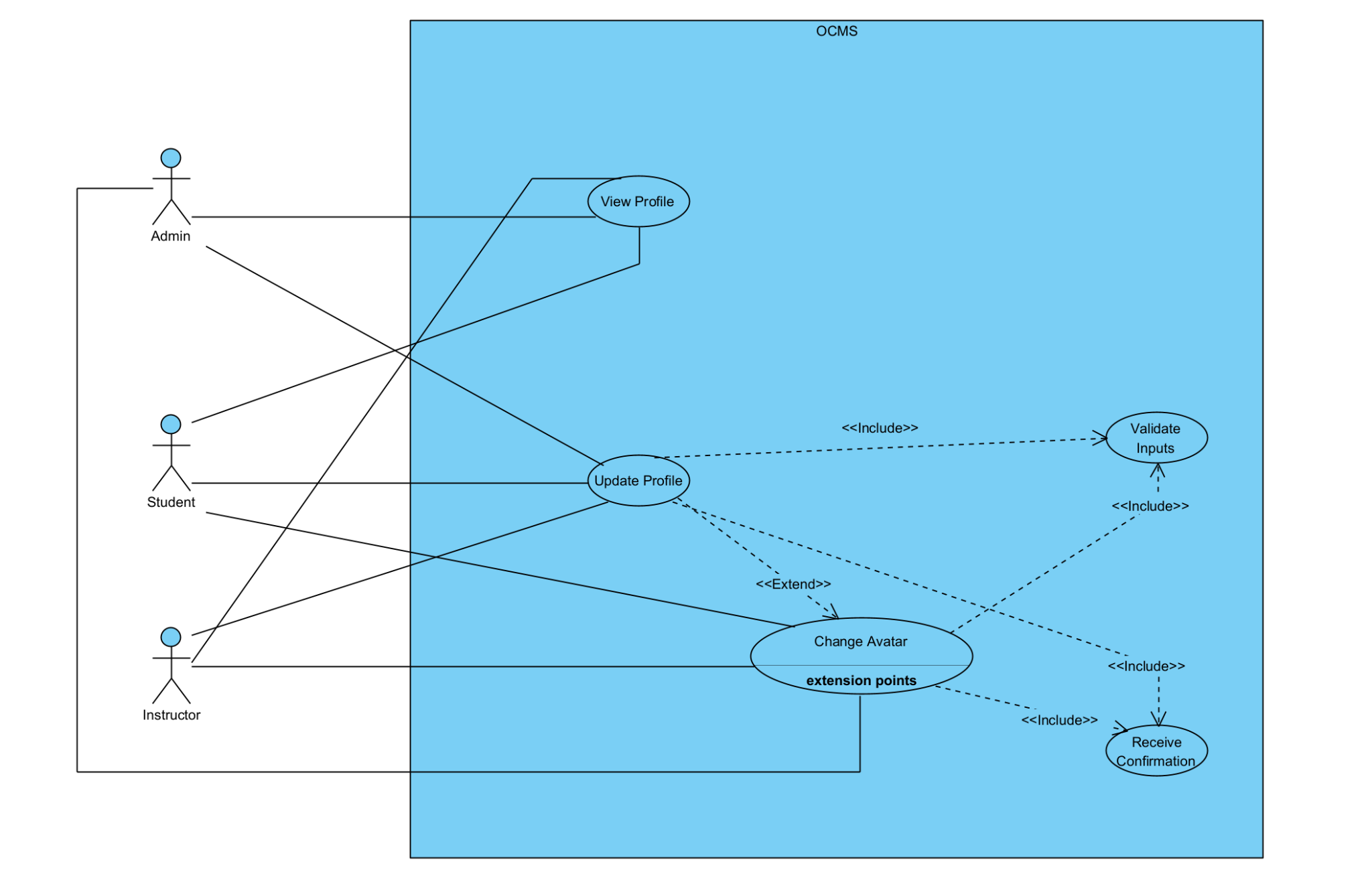
### Function delete course



|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | Delete Course | | |
| Created By: | TuanVM | Date Created: | 17/10/2024 |
| Primary Actor: | Admin | Secondary Actors: |  |
| Trigger: | Admin initiates the process of deleting a course. | | |
| Description: | This use case describes how an admin can remove an existing course from the system. | | |
| Preconditions: | The course to be deleted must exist in the system. | | |
| Postconditions: | The system deletes the course information from the database. | | |
| Normal Flow: | 1. Admin accesses the "Manage Courses" screen.   2. Admin selects the course to be deleted.  3. Admin confirms the deletion.  4. The system removes the course from the database and shows a success message. | | |
| Alternative Flows: | **Course Not Found:** If the course does not exist, the system displays an error message.  **Delete Canceled:** If the admin cancels the deletion, no changes are made. | | |
| Exceptions: | Database connection failure. | | |
| Priority: | Medium | | |
| Frequency of Use: | Occasionally, when courses need to be removed. | | |
| Business Rules: | Only authorized users (admins) can delete courses. | | |
| Other Information: | Deletion is permanent and cannot be undone. | | |
| Assumptions: | The course being deleted is no longer needed. | | |



## 3.3 System Feature Update,View Profile



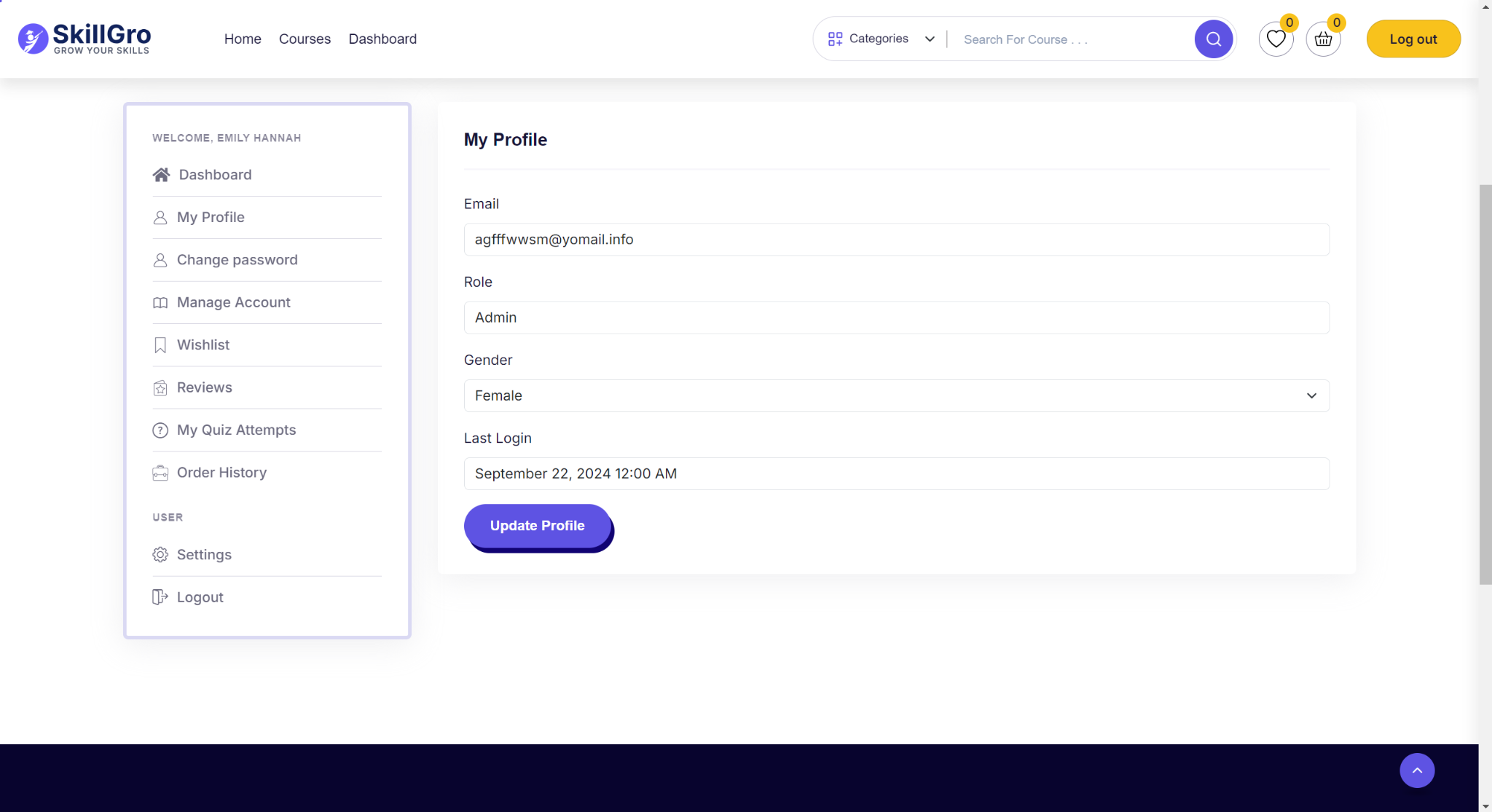
**3.3.1 Description**

The Update and View Profile feature allows users to view and update their personal information in the OCMS system. Users can modify basic information such as full name, email, phone number, and academic information. The system ensures secure validation and storage of these changes.

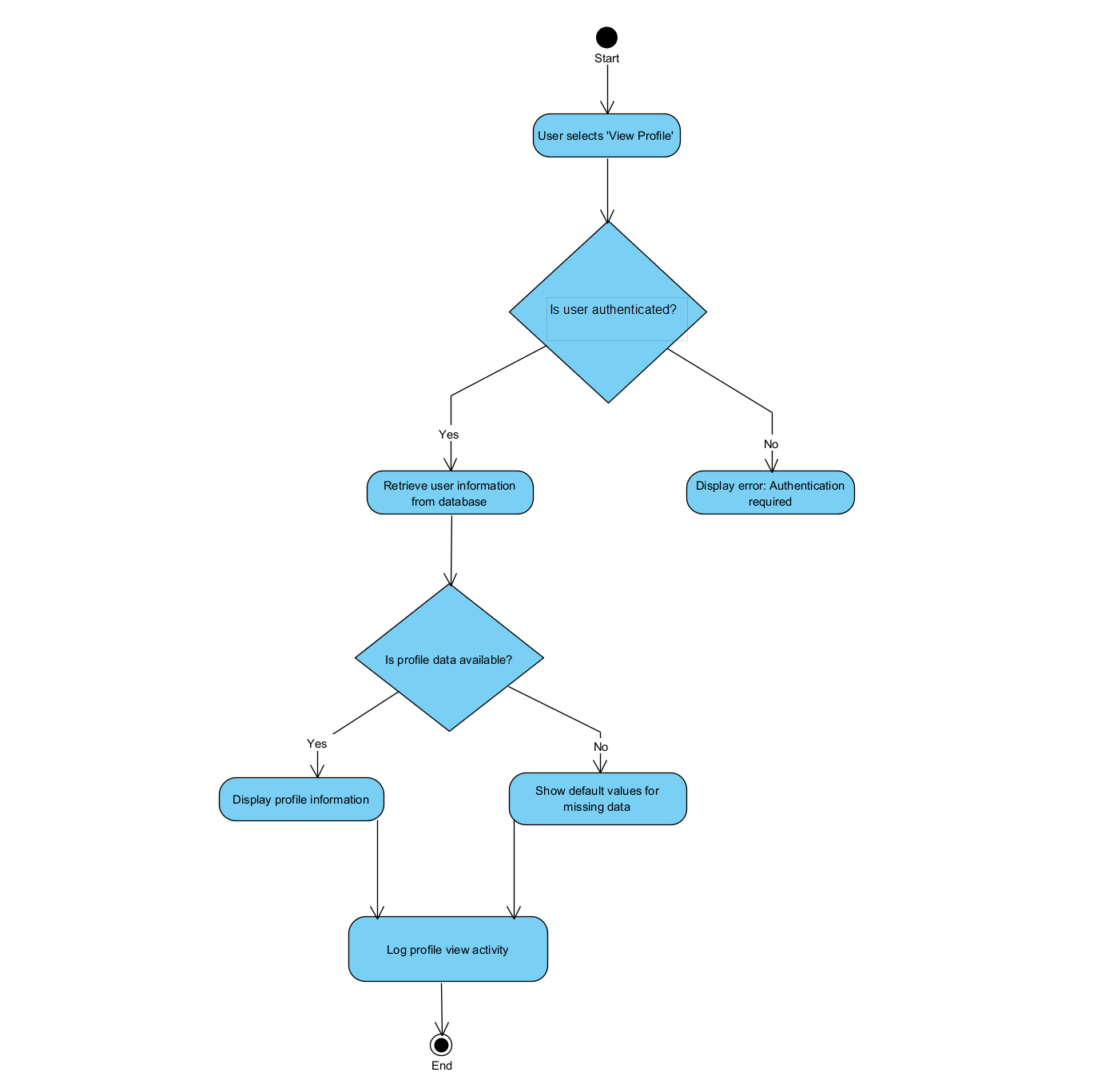
**3.3.2 Stimulus/Response Sequences**

|  |  |
| --- | --- |
| **Stimulus** | **Response** |
| User selects "View Profile" | System displays profile page with current user information |
| User clicks "Edit Profile" button | System switches to edit mode, editable fields become active |
| User modifies information | System validates data format in real-time |
| User submits update form | System validates all data |
| Data is valid | System saves new information and displays success message |
| Data is invalid | Displays detailed error messages |
| User uploads profile picture | System checks and processes image file |
| User cancels update | System returns to profile page |

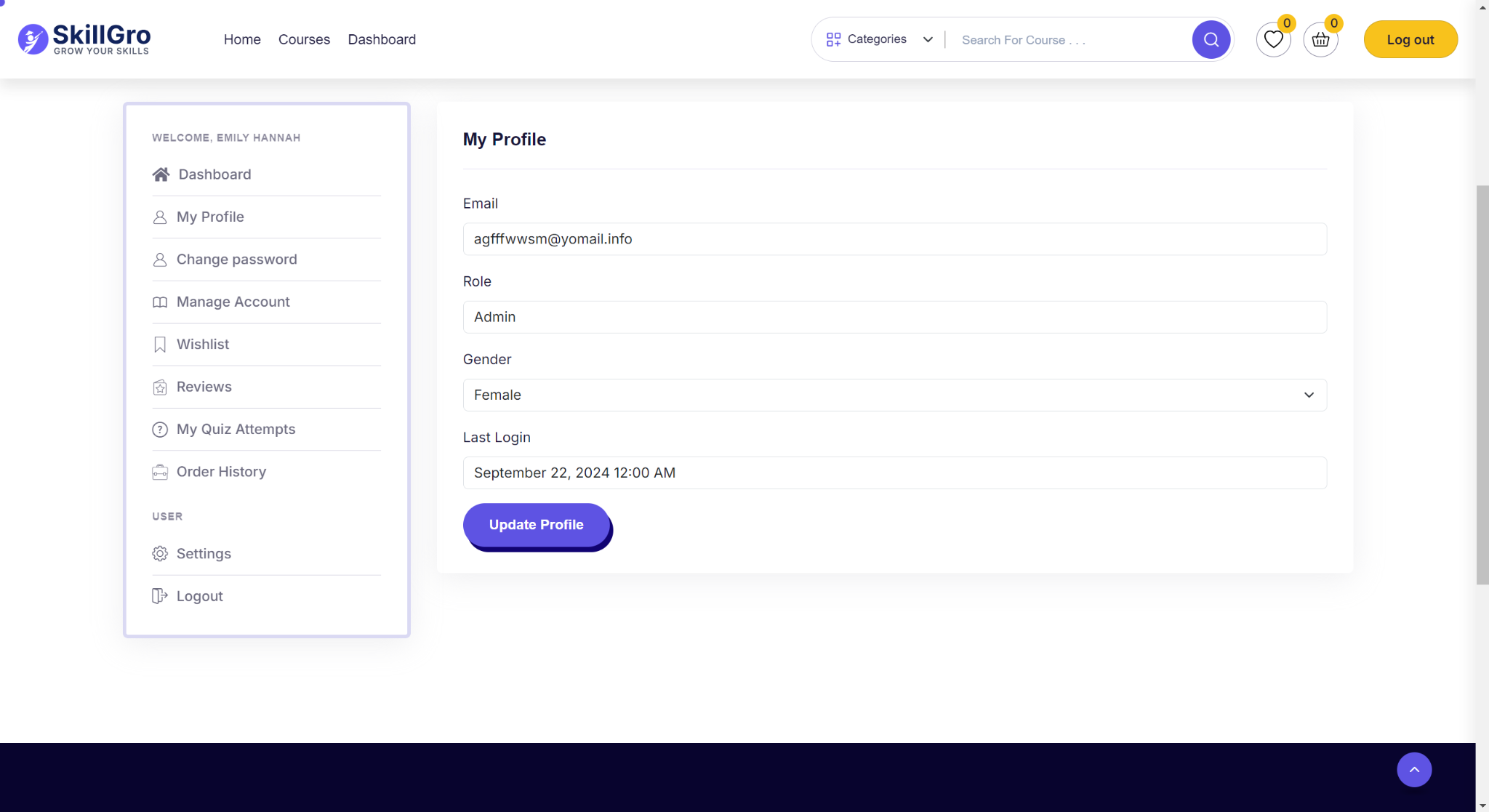
**3.3.3 Function View Profile**



|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | View Profile | | |
| Created By: | HuyPQ | Date Created: | 12/08/2024 |
| Primary Actor: | User (Student/Instructor/Admin) | Secondary Actors: | System Database |
| Trigger: | User selects "View Profile" option from the system menu | | |
| Description: | Allows users to view their personal information in the OCMS system | | |
| Preconditions: | 1. User must be logged into the OCMS system  2. User must have a valid account | | |
| Postconditions: | User profile information is displayed correctly on the screen | | |
| Normal Flow: | 1. User accesses their profile page  2. System retrieves user information from database  3. System displays profile information including:  - Email  - Role  - Gender  - Last login | | |
| Alternative Flows: | None | | |
| Exceptions: | 1. Database connection error:  - System displays error message  - Suggests refreshing the page  2. Missing profile data:  - System displays default values  - Shows "Not Available" for empty fields | | |
| Priority: | High | | |
| Frequency of Use: | High - Daily | | |
| Business Rules: | 1. Only authenticated users can view their own profile  2. All displayed information must be retrieved from secure database  3. Sensitive information must be properly masked | | |
| Other Information: | System should log profile view activities | | |
| Assumptions: | 1. User has stable internet connection  2. User profile exists in database | | |



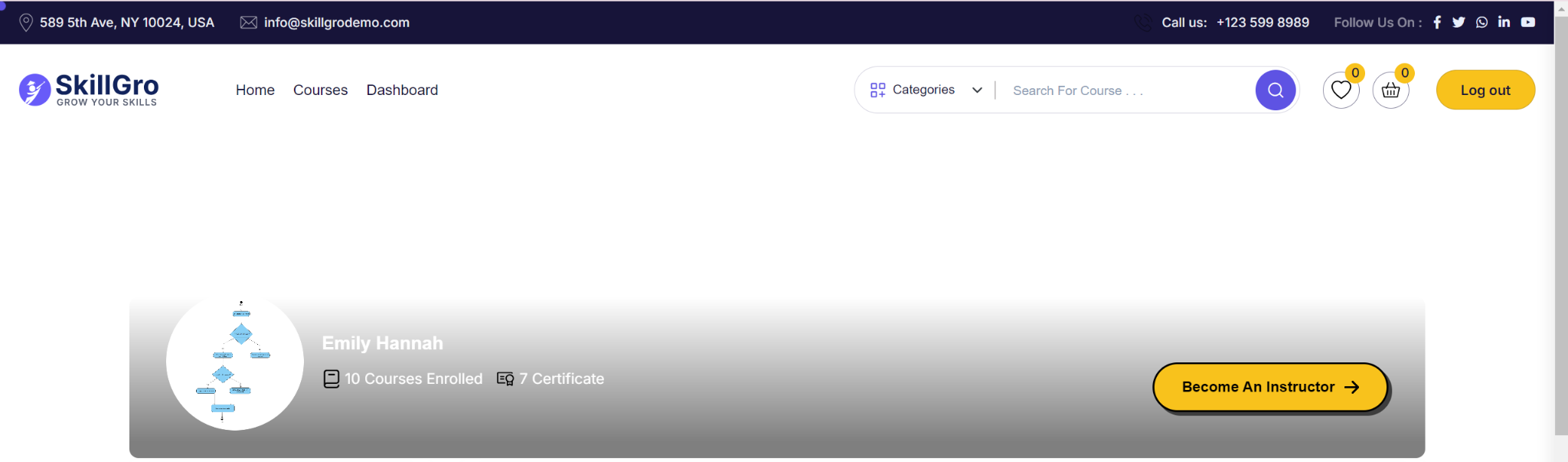
**3.3.3 Function Update Profile**

****

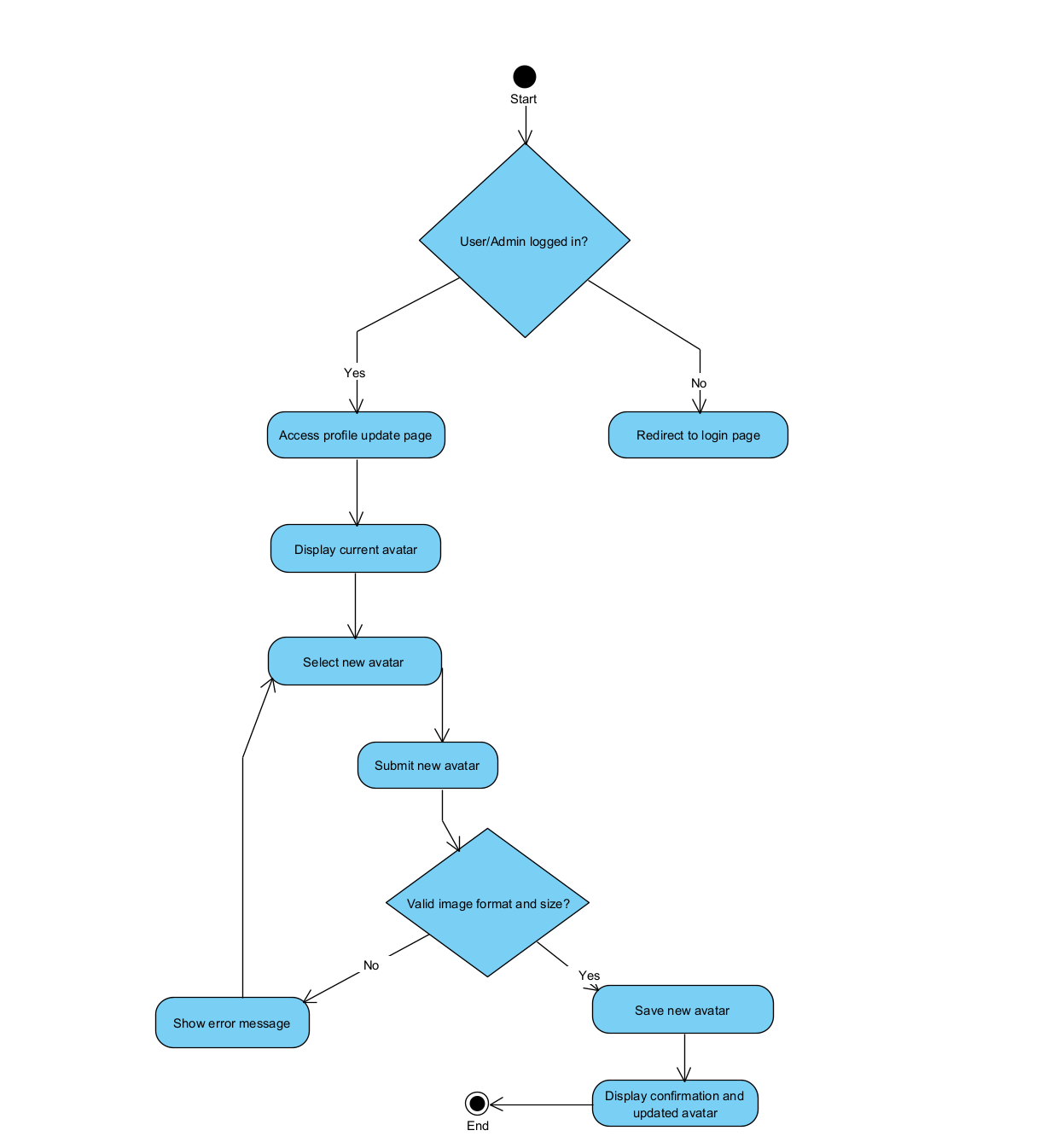
|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | View Profile | | |
| Created By: | HuyPQ | Date Created: | 12/08/2024 |
| Primary Actor: | User (Student/Instructor/Admin) | Secondary Actors: | System Database |
| Trigger: | The user requests to update their profile information. | | |
| Description: | This use case allows the user to modify and update their personal information stored in the system, such as email, phone number, or address. | | |
| Preconditions: | The user must be logged into the system and must have an active account. | | |
| Postconditions: | After the profile is updated, the system will reflect the changes and store the updated information in the database. | | |
| Normal Flow: | The user accesses the profile update section.  The system displays the current profile information.  The user makes changes to the desired fields.  The user submits the updated profile.  The system validates the changes and updates the profile in the database.  The system confirms that the profile has been updated successfully. | | |
| Alternative Flows: | 3a. If the user provides invalid data (e.g., incorrect email format), the system shows an error message and requests corrections.  5a. If the system encounters an issue while updating the profile, it prompts the user to retry. | | |
| Exceptions: | The system is down and cannot process updates.  User session has expired, requiring the user to log in again. | | |
| Priority: | Medium | | |
| Frequency of Use: | Whenever users need to update their profile information. | | |
| Business Rules: | Only authorized users can update their profiles.  Some fields, like the user ID, cannot be changed. | | |
| Other Information: | Changes to sensitive information, such as passwords, may require additional security steps like two-factor authentication. | | |
| Assumptions: | Users have valid information that needs to be updated.  The system ensures data integrity during the update process. | | |



**3.3.4 Function Update Avatar**

****

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | View Profile | | |
| Created By: | HuyPQ | Date Created: | 12/08/2024 |
| Primary Actor: | User (Student/Instructor/Admin) | Secondary Actors: | System Database |
| Trigger: | The user/admin requests to update the avatar. | | |
| Description: | This use case allows the user or admin to change their profile picture (avatar). | | |
| Preconditions: | The user/admin must be logged into the system and must have an existing account. | | |
| Postconditions: | After successfully updating the avatar, the new profile picture is saved in the system, and the user/admin can view the updated avatar. | | |
| Normal Flow: | 1. The user/admin accesses the profile update section.  2. The system displays the current avatar.  3. The user/admin selects a new avatar to upload.  4. The system validates the image (e.g., format, size).  5. The system saves the new avatar in the database.  6. The system confirms the avatar update and displays the new avatar. | | |
| Alternative Flows: | 3a. If the selected image is invalid (e.g., incorrect file type, size too large), the system shows an error message and requests a new file. | | |
| Exceptions: | The system is down and cannot process the update.  The image upload fails due to a network or system issue. | | |
| Priority: | Low to Medium | | |
| Frequency of Use: | Whenever users need to change their avatar. | | |
| Business Rules: | Only authorized users/admins can update the avatar.  The avatar must meet system-defined image requirements. | | |
| Other Information: | Updating the avatar may require refreshing the user profile page to reflect changes. | | |
| Assumptions: | Users have the appropriate image file ready to upload.  The system ensures data integrity during the update process. | | |

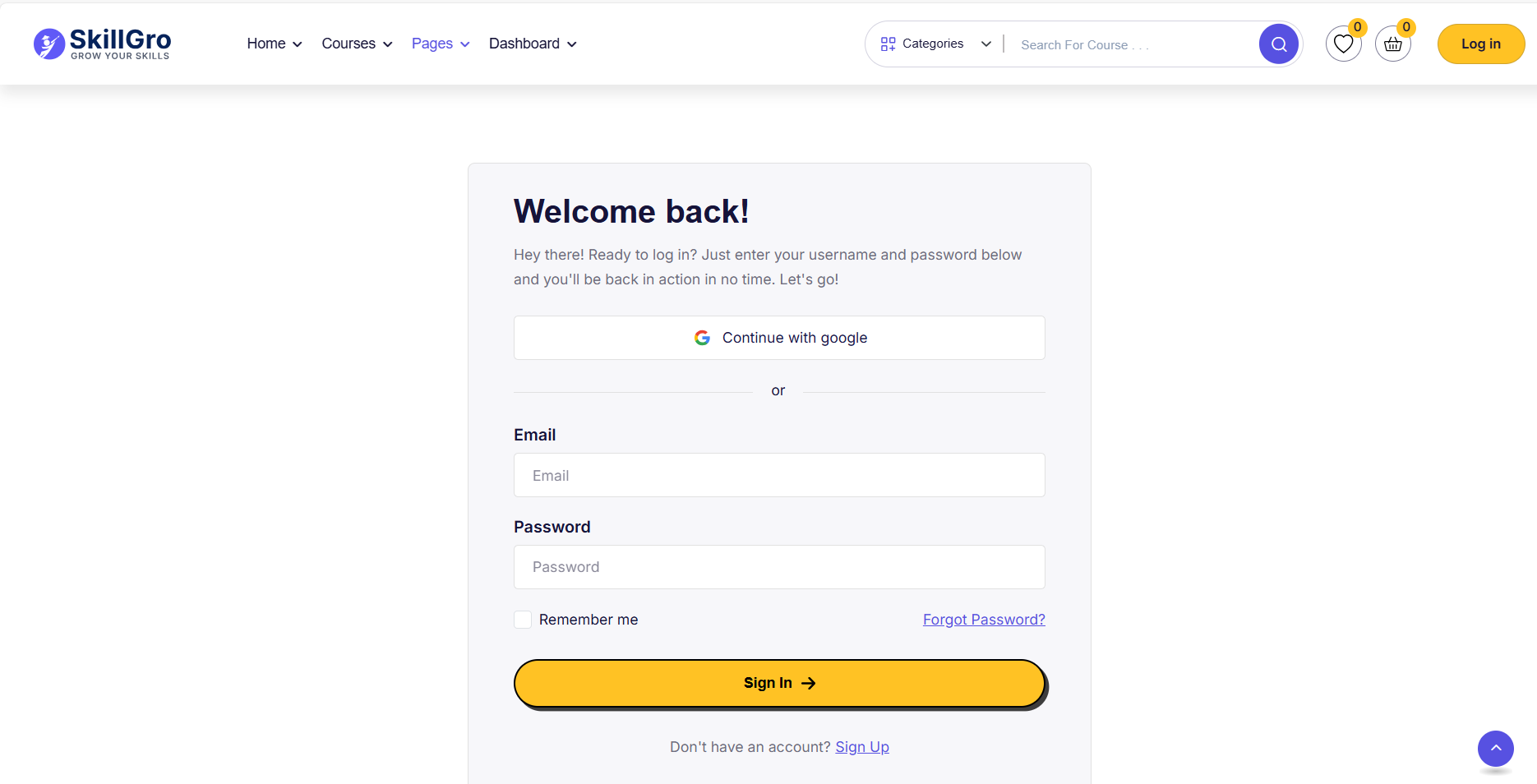
****

**3.4 System Feature Authentication**

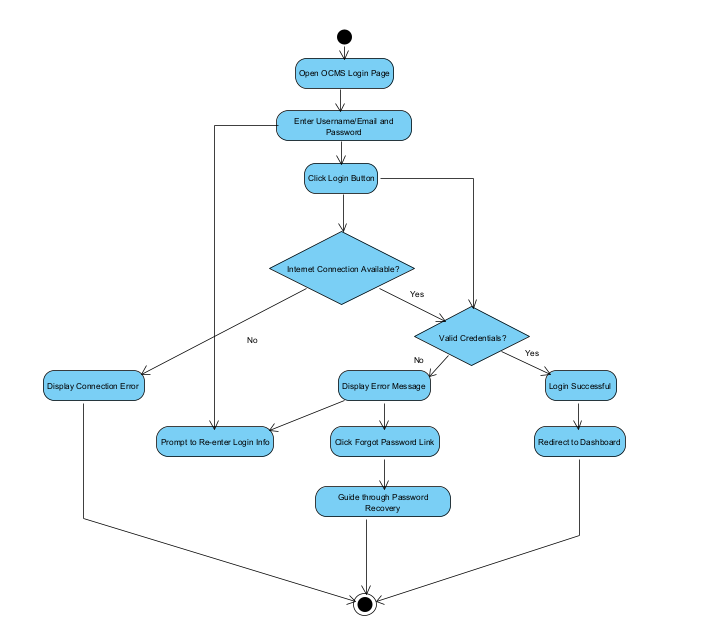
**3.4.1 Description**

**3.4.2 Stimulus/ Response**

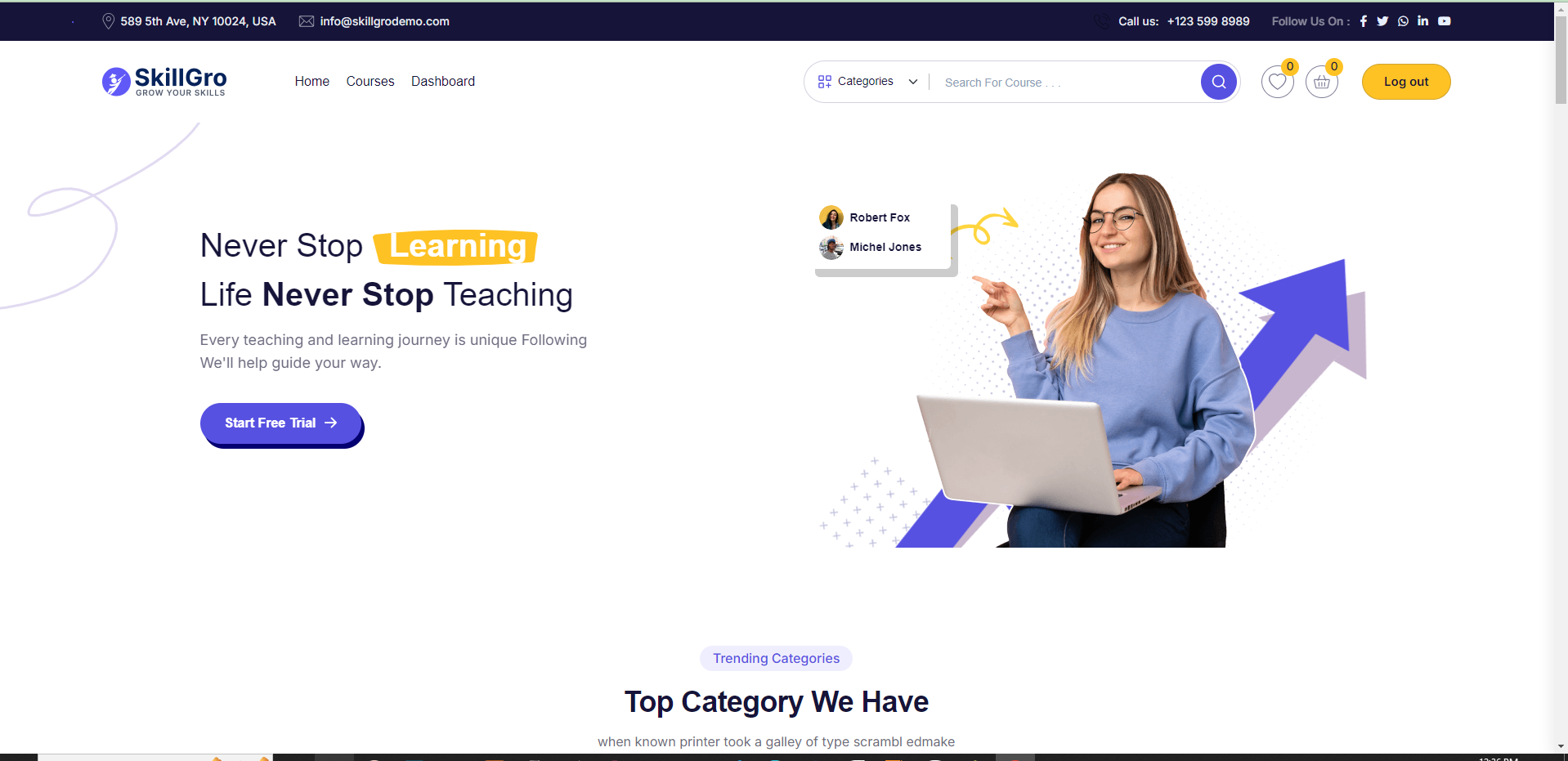
**3.4.3 Function Login**

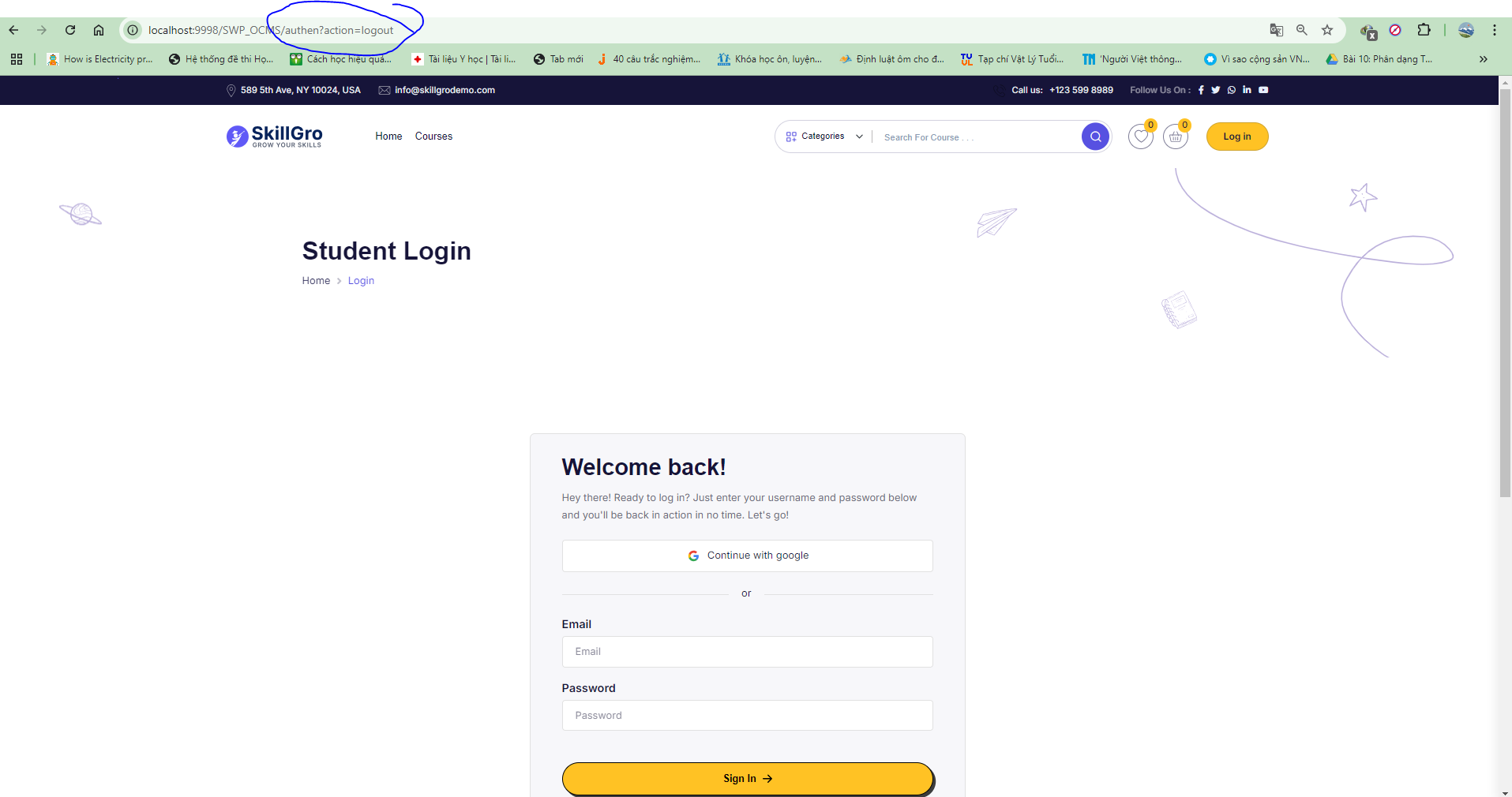
****

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Login | | |
| Created By: | Phan Tiến Mạnh | Date Created: | 10/10/2024 |
| Primary Actor: | Student, Instructor | Secondary Actors: |  |
| Trigger: | This use case describes how a student/instructor logs into the OCMS platform to access their personalized dashboard, courses, and related materials. | | |
| Description: | This use case describes how the student/instructor login to OCMS system | | |
| Preconditions: | The student has an active account on OCMS.  The student knows their login credentials (username/email and password). | | |
| Postconditions: | 1. The student successfully logs into the platform and is redirected to their dashboard. | | |
| Normal Flow: | 1. The student opens the OCMS login page.  2. The student enters their username/email and password.  3. The student clicks the "Login" button.  4. The system validates the credentials.  5. The system logs the student in and redirects them to their dashboard. | | |
| Alternative Flows: | 1. Invalid login credentials:   * If the student enters incorrect credentials, the system displays an error message. * The student is prompted to re-enter their login information. * Return to step 2 in normal flow.   2. Forgotten password:   * The student clicks the "Forgot Password" link. * The system guides the student through the password recovery process. | | |
| Exceptions: | Loss of internet connection during login attempt.  System failure or downtime. | | |
| Priority: | High | | |
| Frequency of Use: | Students log in multiple times per day, especially before and after class hours. | | |
| Business Rules: | Only registered students with valid accounts can log in.  Password must meet the security guidelines set by the institution. | | |
| Other Information: | This use case is part of the essential functionalities of the OCMS platform. | | |
| Assumptions: | Students have reliable internet access.  Students know their login credentials. | | |

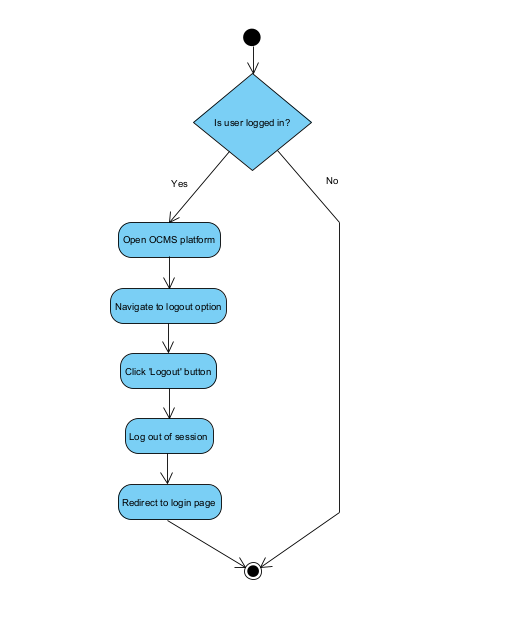
****

**3.4.3 Function Logout**

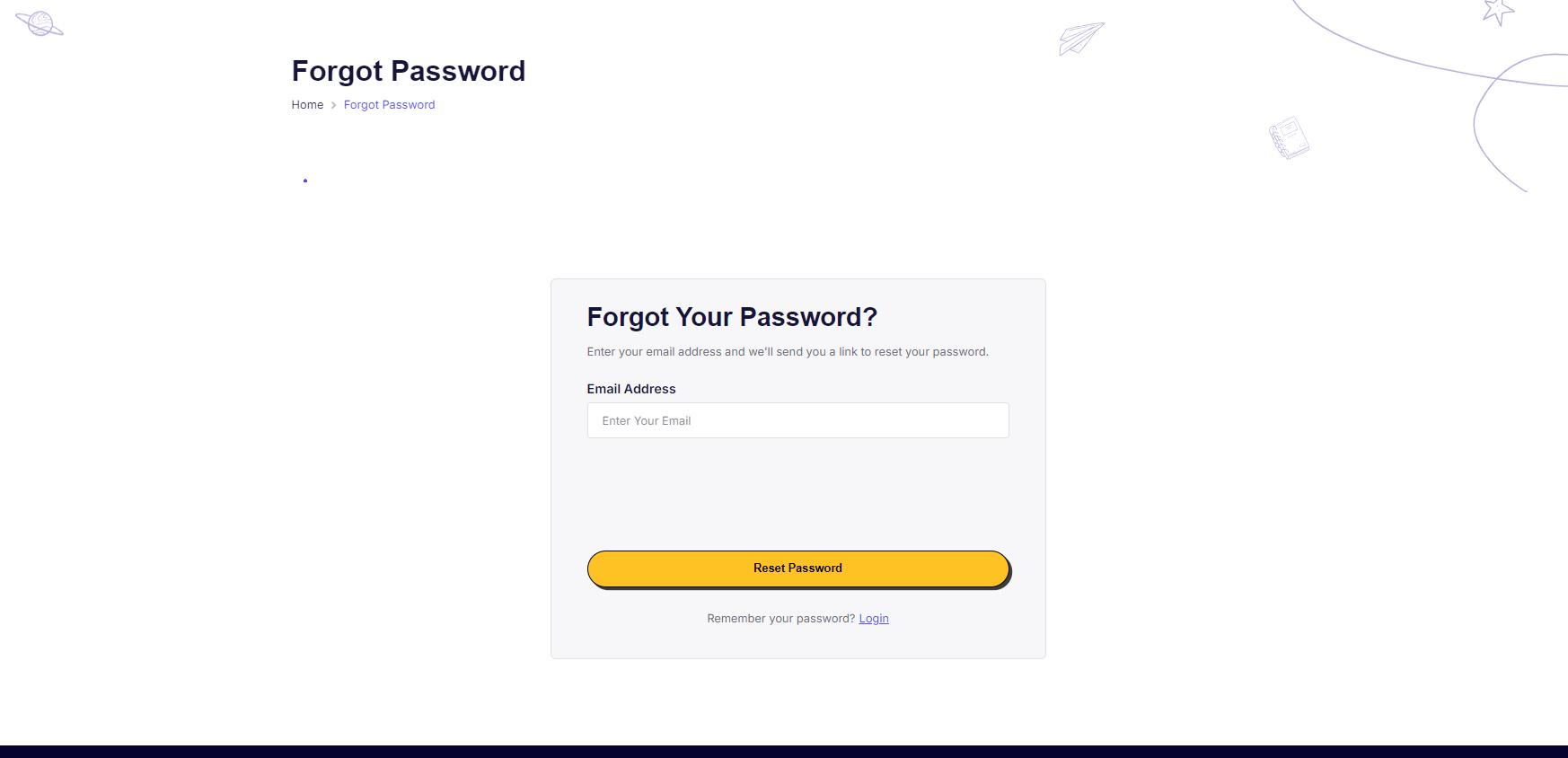
****

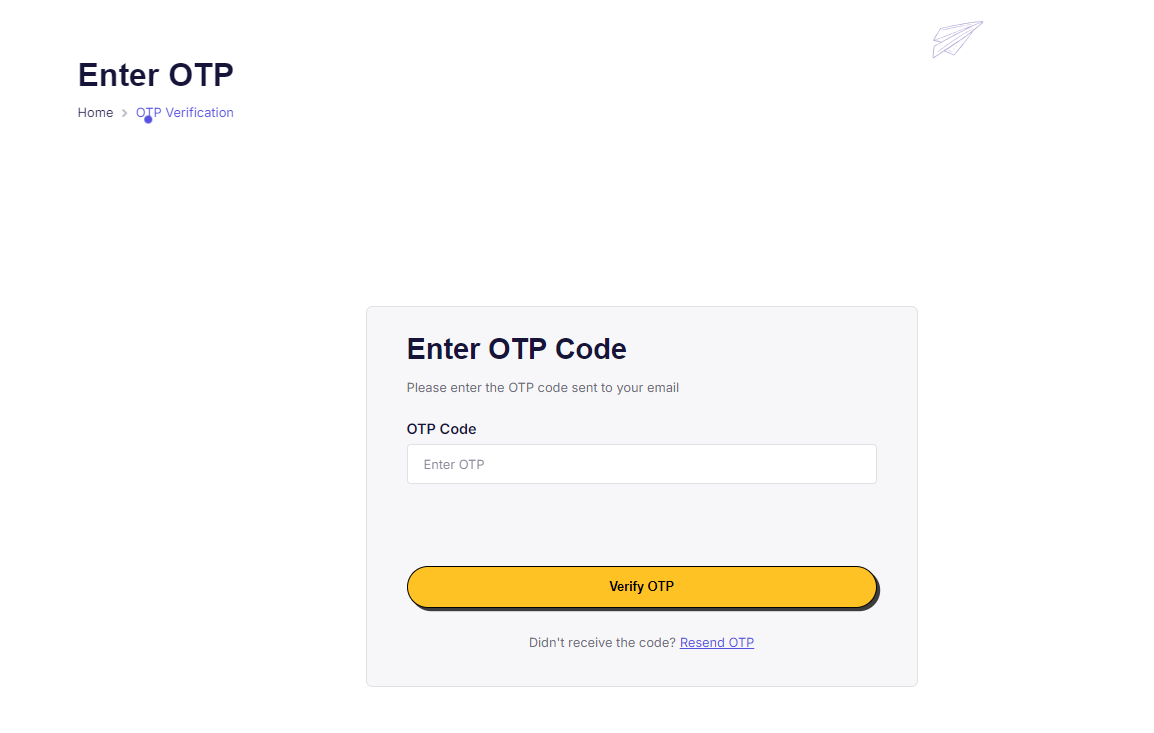
****

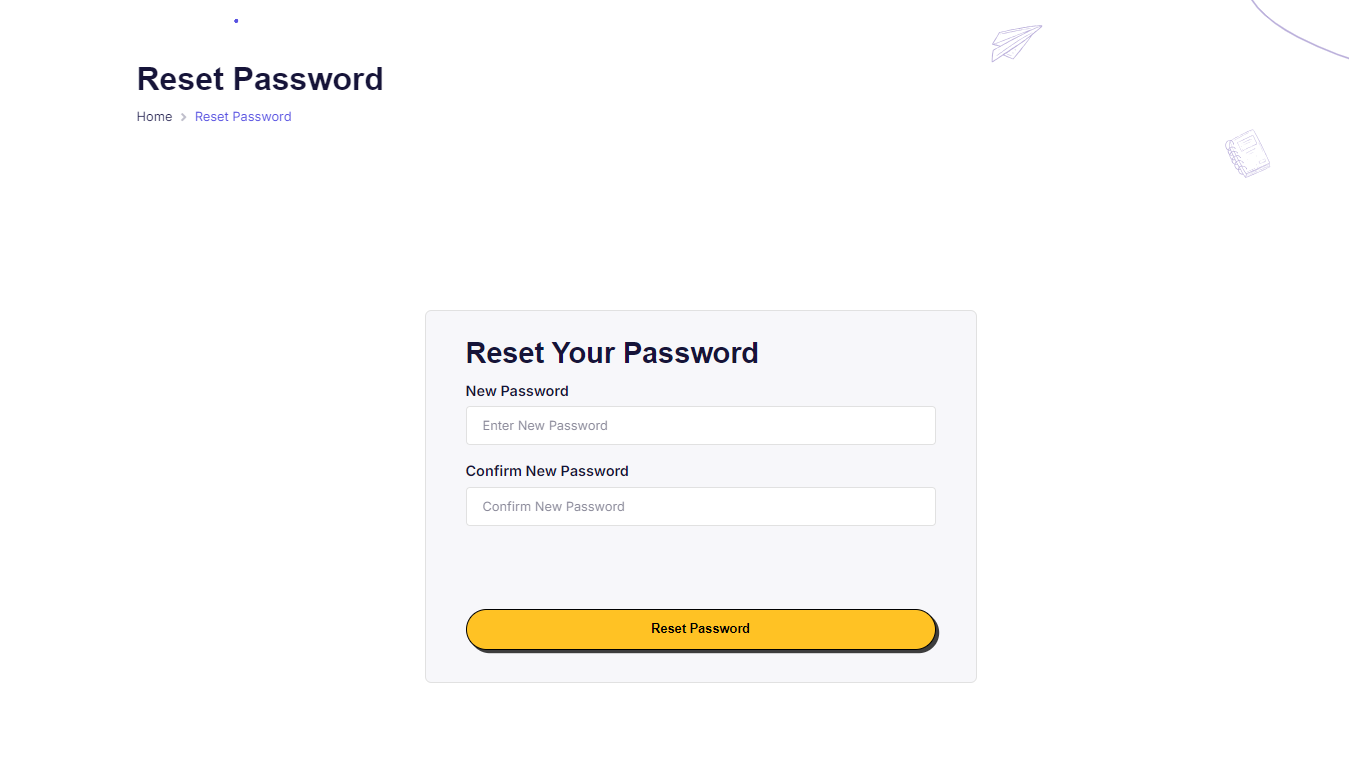
|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Logout | | |
| Created By: | Phan Tiến Mạnh | Date Created: | 10/10/2024 |
| Primary Actor: | Student, Instructor | Secondary Actors: |  |
| Trigger: | The student/instructor decides to log out of the OCMS platform. | | |
| Description: | This use case describes how the student/instructor logs out of the OCMS system. | | |
| Preconditions: | The student/instructor is already logged into the OCMS platform. | | |
| Postconditions: | 1. The student/instructor is successfully logged out of the platform.  2. The system displays the login page. | | |
| Normal Flow: | |  | | --- | |  |  |  | | --- | | 1. The student/instructor opens the OCMS platform and navigates to the logout option.  2. The student/instructor clicks the "Logout" button.  3. The system logs the student/instructor out of the session.  4. The system redirects the student/instructor to the login page. | | | |
| Alternative Flows: | None | | |
| Exceptions: | E1. **Session timeout**: If the session expires before the user clicks "Logout," the system will automatically log the student/instructor out and redirect them to the login page.  E2. **System failure**: If the system fails during the logout process, an error message will be displayed, but the session may be forcibly terminated. | | |
| Priority: | High | | |
| Frequency of Use: | Students/instructors log out 1-2 times per day, typically after completing tasks. | | |
| Business Rules: | 1. A session should automatically log out after a predefined period of inactivity for security reasons. | | |
| Other Information: | This use case is part of the essential security features of the OCMS platform. | | |
| Assumptions: | Students/instructors have successfully logged in before initiating the logout. | | |

****

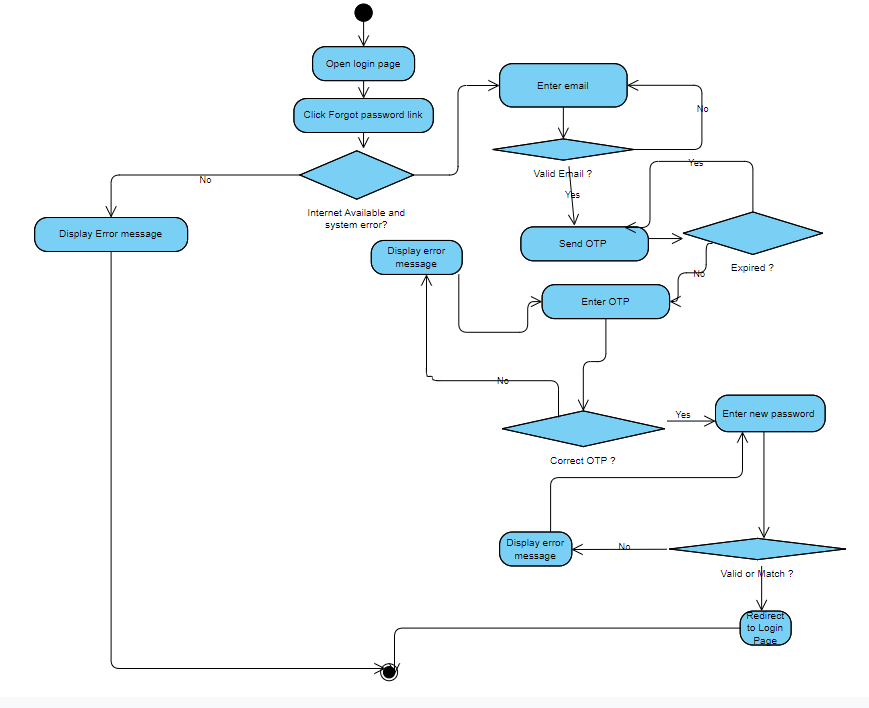
**3.4.3 Function Forget Password**

****

****

****

|  |  |  |  |
| --- | --- | --- | --- |
| 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

*<This section describes various aspects of the data that the system will consume as inputs, process in some fashion, or create as outputs.>*

## Logical Data Model

*<A data model is a visual representation of the data objects and collections the system will process and the relationships between them. Include a data model for the business operations being addressed by the system, or a logical representation for the data that the system itself will manipulate. Data models are most commonly created as an entity-relationship diagram.>*

* *Entity-relationship diagram here.*
* *Class diagram here.*
* *State diagram here.*

## Data Dictionary

*<The data dictionary defines the composition of data structures and the meaning, data type, length, format, and allowed values for the data elements that make up those structures. In many cases, you're better off storing the data dictionary as a separate artifact, rather than embedding it in the middle of an SRS. That also increases its reusability potential in other projects.>*

## Reports

*<If your application will generate any reports, identify them here and describe their characteristics. If a report must conform to a specific predefined layout you can specify that here as a constraint, perhaps with an example. Otherwise, focus on the logical descriptions of the report content, sort sequence, totaling levels, and so forth, deferring the detailed report layout to the design stage.>*

## Data Acquisition, Integrity, Retention, and Disposal

*<If relevant, describe how data is acquired and maintained. State any requirements regarding the need to protect the integrity of the system's data. Identify any specific techniques that are necessary, such as backups, checkpointing, mirroring, or data accuracy verification. State policies the system must enforce for either retaining or disposing of data, including temporary data, metadata, residual data (such as deleted records), cached data, local copies, archives, and interim backups.>*

# External Interface Requirements

*<This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements.>*

## User Interfaces

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

## Software Interfaces

*<Describe the connections between this product and other software components (identified by name and version), including other applications, databases, operating systems, tools, libraries, websites, and integrated commercial components. State the purpose, formats, and contents of the messages, data, and control values exchanged between the software components. Specify the mappings of input and output data between the systems and any translations that need to be made for the data to get from one system to the other. Describe the services needed by or from external software components and the nature of the intercomponent communications. Identify data that will be exchanged between or shared across software components. Specify nonfunctional requirements affecting the interface, such as service levels for responses times and frequencies, or security controls and restrictions.>*

## Hardware Interfaces

*<Describe the characteristics of each interface between the software and hardware (if any) components of the system. This description might include the supported device types, the data and control interactions between the software and the hardware, and the communication protocols to be used. List the inputs and outputs, their formats, their valid values or ranges, and any timing issues developers need to be aware of. If this information is extensive, consider creating a separate interface specification document.>*

## Communications Interfaces

*<State the requirements for any communication functions the product will use, including e-mail, Web browser, network protocols, and electronic forms. Define any pertinent message formatting. Specify communication security or encryption issues, data transfer rates, handshaking, and synchronization mechanisms. State any constraints around these interfaces, such as whether e-mail attachments are acceptable or not.>*

# Quality Attributes

## Usability

*<Specify any requirements regarding characteristics that will make the software appear to be “user-friendly.” Usability encompasses ease of use, ease of learning; memorability; error avoidance, handling, and recovery; efficiency of interactions; accessibility; and ergonomics. Sometimes these can conflict with each other, as with ease of use over ease of learning. Indicate any user interface design standards or guidelines to which the application must conform.>*

## Performance

*<State specific performance requirements for various system operations. If different functional requirements or features have different performance requirements, it's appropriate to specify those performance goals right with the corresponding functional requirements, rather than collecting them in this section.>*

## Security

*<Specify any requirements regarding security or privacy issues that restrict access to or use of the product. These could refer to physical, data, or software security. Security requirements often originate in business rules, so identify any security or privacy policies or regulations to which the product must conform. If these are documented in a business rules repository, just refer to them.>*

## Safety

*<Specify requirements that are concerned with possible loss, damage, or harm that could result from use of the product. Define any safeguards or actions that must be taken, as well as potentially dangerous actions that must be prevented. Identify any safety certifications, policies, or regulations to which the product must conform.>*

## [Others as relevant]

*<Create a separate section in the SRS for each additional product quality attribute to describe characteristics that will be important to either customers or developers. Possibilities include availability, efficiency, installability, integrity, interoperability, modifiability, portability, reliability, reusability, robustness, scalability, and verifiability. Write these to be specific, quantitative, and verifiable. Clarify the relative priorities for various attributes, such as security over performance.>*

# Internationalization and Localization Requirements

*<Internationalization and localization requirements ensure that the product will be suitable for use in nations, cultures, and geographic locations other than those in which it was created. Such requirements might address differences in: currency; formatting of dates, numbers, addresses, and telephone numbers; language, including national spelling conventions within the same language (such as American versus British English), symbols used, and character sets; given name and family name order; time zones; international regulations and laws; cultural and political issues; paper sizes used; weights and measures; electrical voltages and plug shapes; and many others.>*

# Other Requirements

*<Examples are: legal, regulatory or financial compliance, and standards requirements; requirements for product installation, configuration, startup, and shutdown; and logging, monitoring and audit trail requirements. Instead of just combining these all under "Other," add any new sections to the template that are pertinent to your project. Omit this section if all your requirements are accommodated in other sections. >*

**Appendix A: Glossary**

*<Define any specialized terms that a reader needs to know to understand the SRS, including acronyms and abbreviations. Spell out each acronym and provide its definition. Consider building a reusable enterprise-level glossary that spans multiple projects and incorporating by reference any terms that pertain to this project.>*

**Appendix B: Analysis Models**

*<This optional section includes or points to pertinent analysis models such as data flow diagrams, feature trees, state-transition diagrams, or entity-relationship diagrams. You might prefer to insert certain models into the relevant sections of the specification instead of collecting them at the end.>*