Student Management System

GitHub: https://github.com/WCSU-CS350-F24-Group-1

Project Team: Nikita Karim, Jakub Olsovsky, Daniel Espin, Vadim Cheremukhin, Ronan Banton, Jason Siecienski, Lucas Vas, and Marcos Santiago

Professor Coffman

Monday, 18 November 2024

Western Connecticut State University

CS350 - Object Oriented Software Engineering

Table of Contents

I. <u>Introduction</u>

The **Student Management System (SMS) Report** provides a comprehensive overview of the proposed SMS, detailing its objectives, system requirements, design structure, and essential functionalities. This document is organized to present a clear and thorough understanding of the system, from its purpose to the technical requirements and implementation strategy. Each section includes specific functionalities and requirements that are essential for efficient and accurate student data management within educational institutions. Diagrams are included to visually represent the system structure and workflow. This document aims to guide the design, development, and future maintenance of the SMS.

II. <u>Problem Statement</u>

The **Student Management System (SMS)** aims to streamline student data management by providing an intuitive interface that enables administrators and faculty to add, retrieve, update, search, and delete student records. The SMS addresses the limitations of manual record-keeping, which often lead to inefficient processes and increased potential for errors. By offering an integrated platform, the SMS enhances data integrity and performance, ensuring reliable access to student information.

Educational institutions commonly face challenges with outdated or inefficient systems that complicate student information management. Such systems can result in time-consuming operations, which detract from the institution's ability to respond promptly to student needs. The SMS minimizes time spent on record management through advanced search capabilities, allowing administrators and faculty to focus on other priorities.

The primary goal of this system is to create a reliable platform for managing student information, with essential functionalities such as quick search access, performance tracking (including GPA and declared majors), and efficient record updates. The SMS is designed to support large numbers of records, ensuring scalability and flexibility as student populations grow.

III. Core Objectives

The SMS is built with the following core objectives in mind:

- **Improve Speed and Accuracy:** Enhance the speed and accuracy of student record retrieval, ensuring users can access information efficiently.
- **Provide Secure Data Management:** Organize and secure student data to reduce errors and maintain integrity.
- **Ensure Scalability:** Design the system to handle growing student populations, with flexibility for additional records and potential future enhancements.

IV. System Requirements

Functional Requirements

1. Add Student Records

- Provide a form with the following UI components for input:
    - **Fields:** Student ID (unique), Name, Address, Phone, Email, DOB, Major, GPA.
    - **Buttons:** Submit, Reset, Cancel.
    - Validate inputs:
- No empty fields.
- Email must include "@" symbol.
- GPA must be between 0.0 and 4.0.
- Unique Student ID (show error for duplicates).
- Save valid records to the database.

2. Add Student Records

- Offer a **search bar** to locate records by Student ID or Name (partial/full match).
- Display matching results in a table or grid layout.
- Allow selection of a record for a detailed view.
- If no matching record is found, display a “No records found” message

3. Search for Student Records

- Display full details for a student: ID, Name, Address, Phone, Email, DOB, Major, GPA.
- Ensure real-time database retrieval for accuracy.

4. Retrieve Student Records

- Provide an **Edit button** to enable modification of Address, Phone, Email, Major, and GPA.

- Validate updates before saving changes, with the option to cancel and return to view mode.

5. Delete Student Records

- Allow authorized users to delete records.
- Confirm deletion with a pop-up dialog:
    - If confirmed, delete the record and display a success message.
    - If canceled, retain the record.

Non-Functional Requirements

1. Performance

- The system shall handle up to 100 concurrent users without performance degradation.
- Student record retrieval must occur within 2 seconds.

2. Security

- All sensitive data (e.g., login credentials) must be encrypted in transit and at rest.
- Implement role-based access to restrict features based on user permissions.

3. Scalability

- The system must support future expansion to handle a 50% increase in student records without significant changes.

4. Reliability

- Maintain 99.9% uptime, ensuring the system is operational and available for users.
- Implement automatic backups to protect data integrity in case of failure.

6. Maintainability

- Ensure modular code to simplify updates and troubleshooting.
- Provide documentation for developers to assist in future enhancements.

7. Compatibility

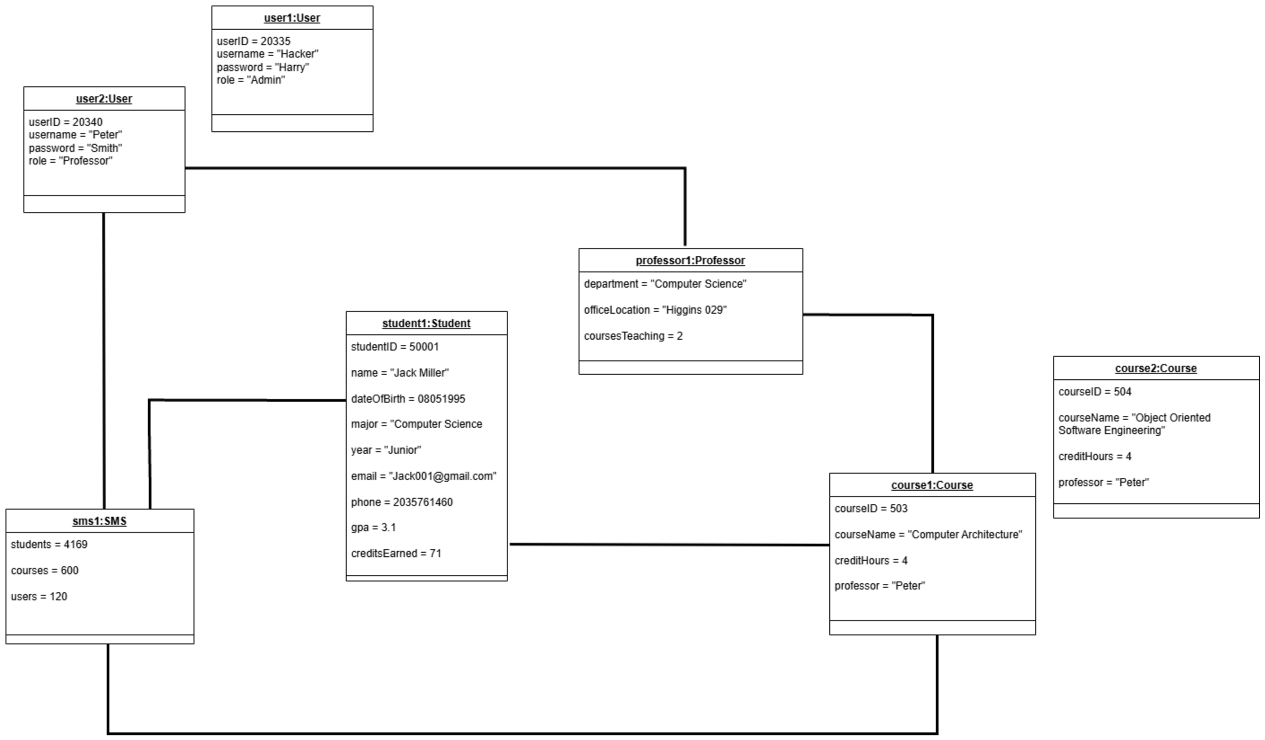
- The system will support only **Windows** as the operating platform.

8. Compliance

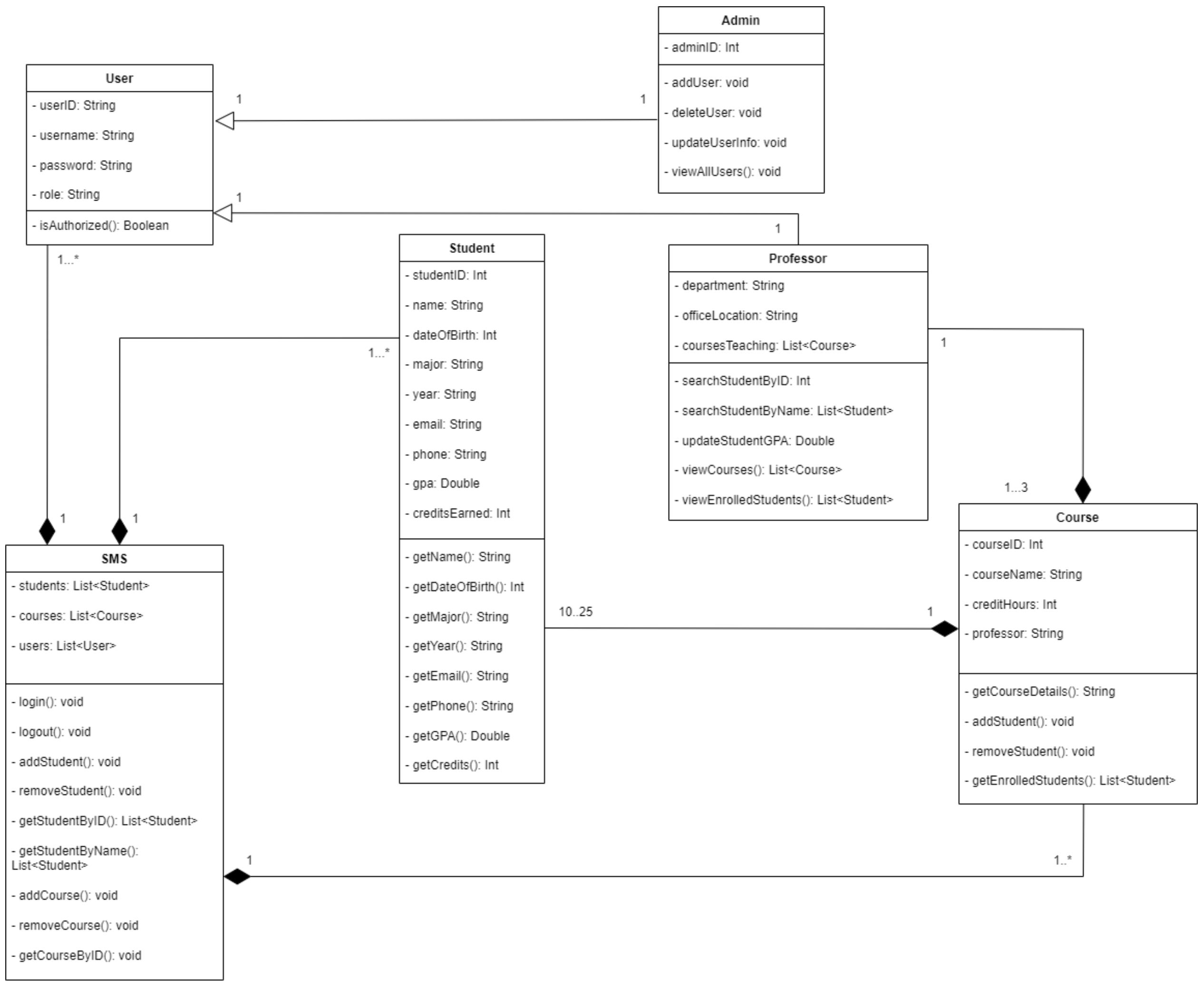
- Adhere to data privacy laws, such as GDPR or FERPA, ensuring student information is protected and used appropriately.

V. <u>Diagrams</u>

A. Object Diagram: The Object Diagram represents the static structure of the system, showing the system objects and their relationships at a specific time.

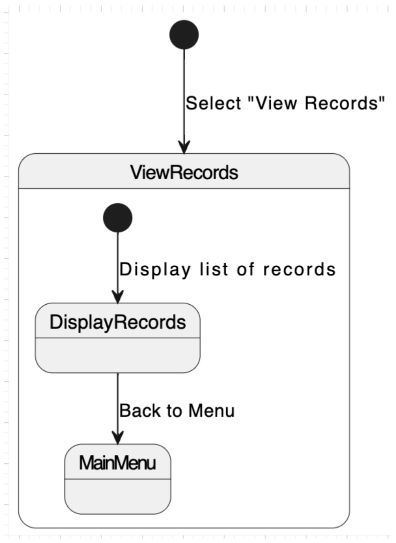


B. Class Diagram: The Class Diagram illustrates the structure of the system by showing classes, attributes, operations, and the relationships between classes.

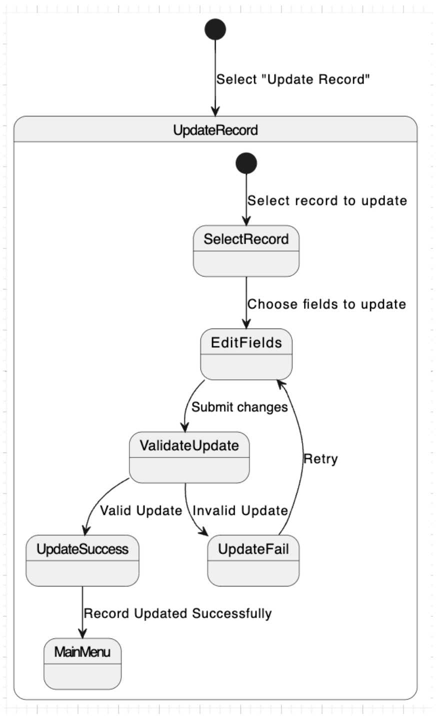


C. State Diagram

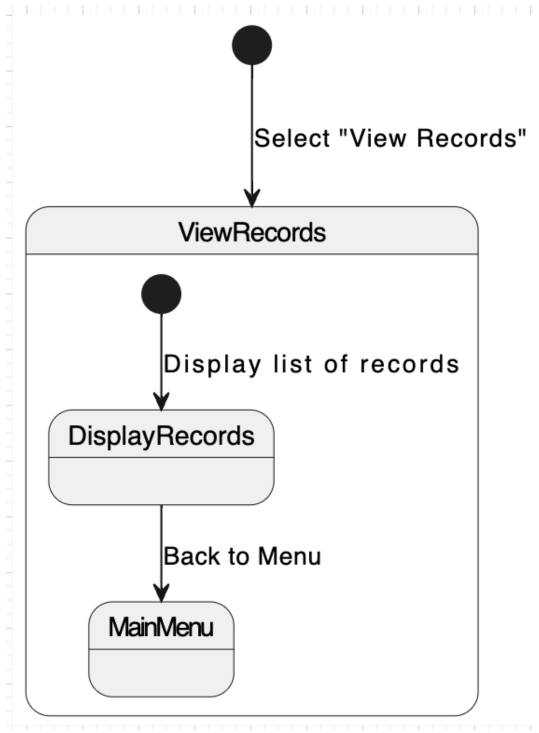
View Student This state diagram visualizes the flow of states when a user is viewing a student's record, including transitions like loading and displaying detailed information.



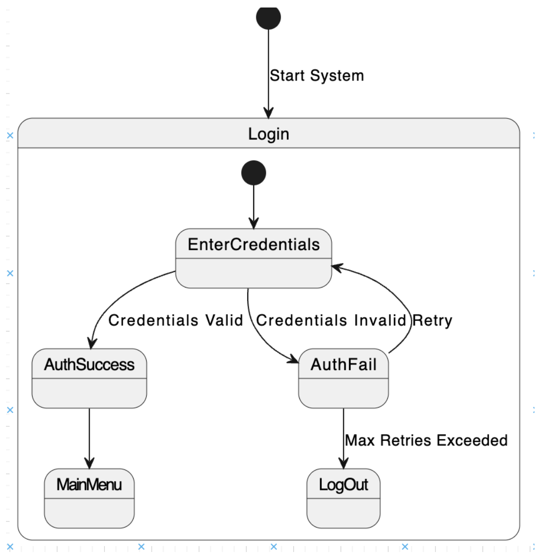
Update Student This state diagram shows the states involved when a user edits and updates a student's record, covering transitions like opening, editing, saving, or canceling updates.



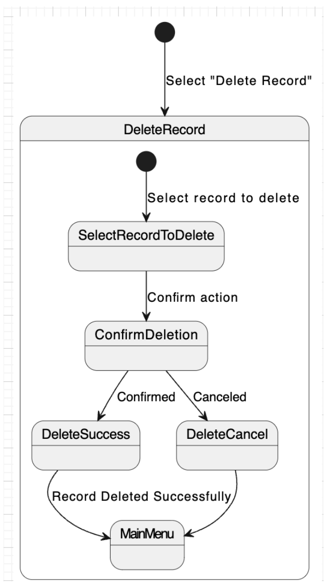
Search Student This state diagram represents the process of searching for a student's record, covering transitions such as entering search criteria, displaying search results, selecting a record for detailed view, or returning to the main menu.



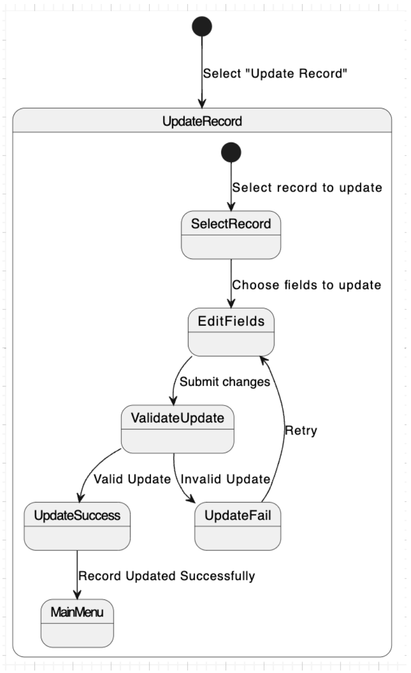
Login This state diagram represents the process of user authentication during login, covering transitions such as entering credentials, validating them, handling invalid attempts, granting access to the main menu, or logging out after exceeding retry limits.



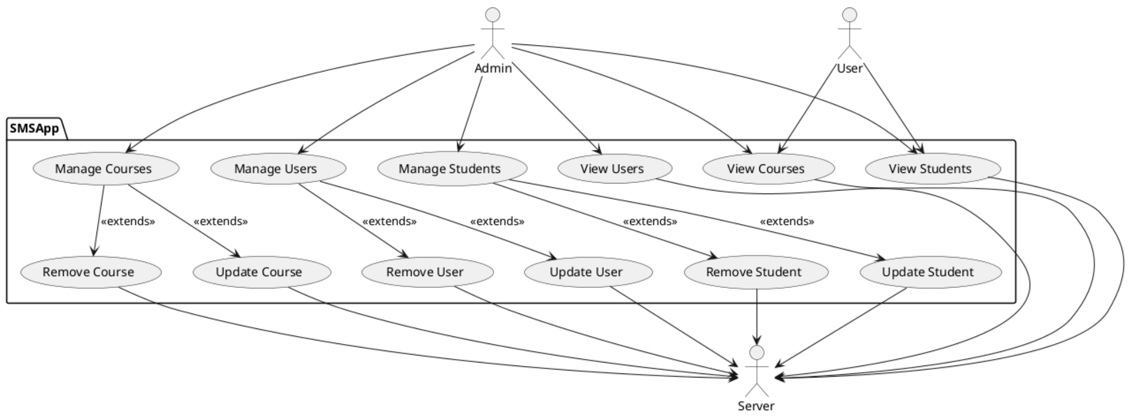
Delete Student This state diagram represents the process of deleting a student's record, covering transitions such as selecting a record to delete, confirming the deletion, successfully removing the record, canceling the action, and returning to the main menu.



Add Student This state diagram represents the process of adding a new student record, covering transitions such as selecting the "Add Student" option, entering required details, validating the input, handling validation errors, successfully saving the record, and returning to the main menu.



E. SMSApp Use Cases and Sequence Diagram



1. **Use Case: Remove Course**

   Actor: User

   Use Case Overview:
   A User initiates the process to remove a user account from the system. The system validates the removal request, checks for dependencies, and processes the removal through the server and database.

   Subject Area: User Management
   Trigger: User selects the option to remove a course.
   Preconditions:
   1. User is logged into the SMS with appropriate permissions.
   2. The user account to be removed exists in the system.

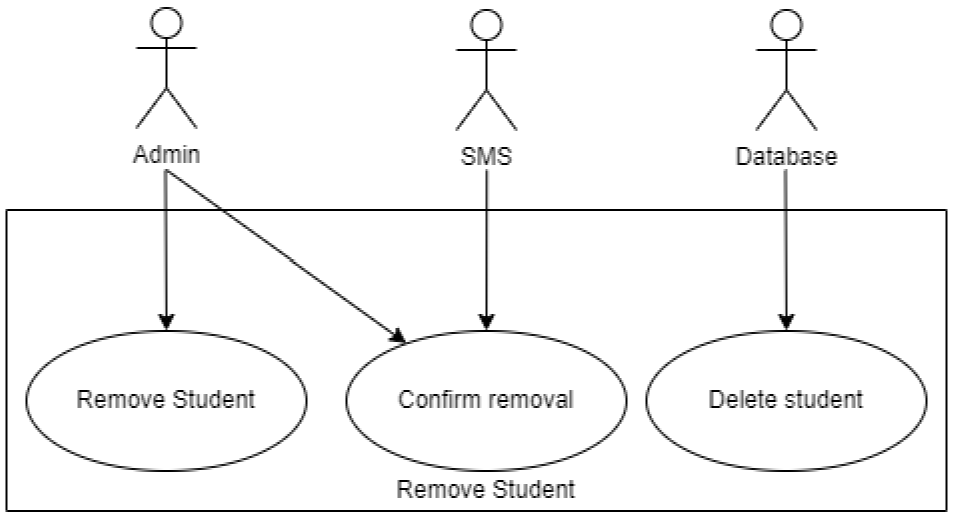
   Description:
   1. User selects the "Remove Course" option in the system.
   2. SMS prompts the User to confirm the removal request.
   3. User confirms the removal request.
   4. SMS validates the removal request and checks for dependencies (assigned roles or tasks).
   5. SMS sends the removal request to the server for validation.
   6. Server processes the request and validates the user’s existence.
   7. Server sends the deletion request to the database.
   8. The database confirms successful deletion of the user account.
   9. Server notifies SMS of the successful removal.
   10. SMS notifies the User of the successful removal.
   11. Termination: The user account is successfully removed from the system.
   Alternate Scenarios:

4A1: If the validation fails (user account has dependencies):
1. SMS notifies the User of the issue.
2. User resolves dependencies and retries the removal process.

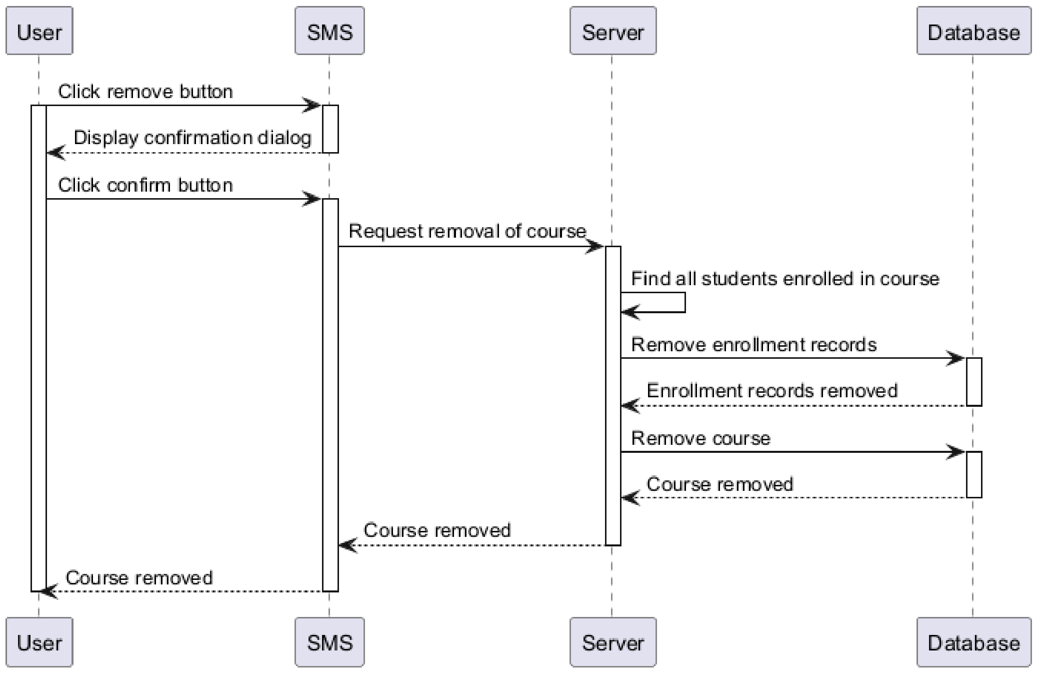
8A1: If the database deletion fails:
1. Server notifies SMS of the failure.
2. SMS notifies the User of the failure.
3. User retries the removal process.

Postconditions:
1. The user account is successfully removed from the system.
2. The updated user list is reflected in the database.

Quality Rules:
- Only authorized users can initiate user account removal.
- The system must validate and resolve all dependencies before processing the removal.
- The system must notify the User of success or failure promptly.



Sequence Diagram: Remove Course


User
SMS
Server
Database
Click remove button
Display confirmation dialog
Click confirm button
Request removal of course
Find all students enrolled in course
Remove enrollment records
Enrollment records removed
Remove course
Course removed
Course removed
Course removed
User
SMS
Server
Database

2. **Use Case: Update Course**

Use Case: Update Course
Actor: Admin

Use Case Overview:
An Admin initiates the process to update an existing course in the Student Management System (SMS). The Admin provides the updated details for the course, which are validated. Based on the outcome, the system either confirms or rejects the update.

Subject Area: Course Management
Trigger: Admin selects the option to update a course.
Preconditions:

1. Admin is logged into the SMS with appropriate permissions.
2. The SMS is operational and capable of processing course updates.

Description:

1. Admin selects the "Update Course" option in the system.
2. SMS displays the list of courses available for update.
3. Admin selects a course and enters the updated details (e.g., course name, description, credits, or prerequisites).
4. SMS validates the entered details.
5. SMS checks if the course already exists in the system.
6. If the course exists, SMS processes the update request.
7. SMS confirms the successful update of the course and reflects the changes in the database.
8. Termination: The course is successfully updated in the system.

Alternate Scenarios:
6A1: If the validation fails:

1. SMS displays an error message and highlights the fields requiring correction.
2. Admin corrects the details and resubmits.

7A1: If the course does not exist:

1. SMS notifies the Admin that the course could not be found and rejects the update.

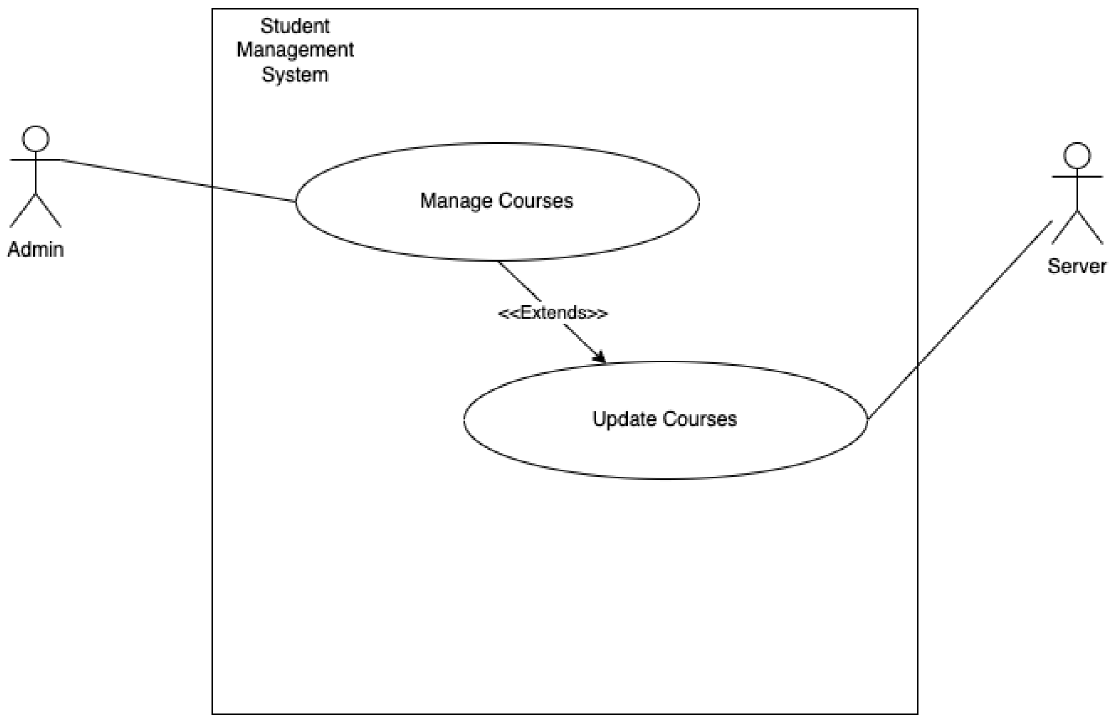
Postconditions:

1. The updated course information is successfully stored in the database.

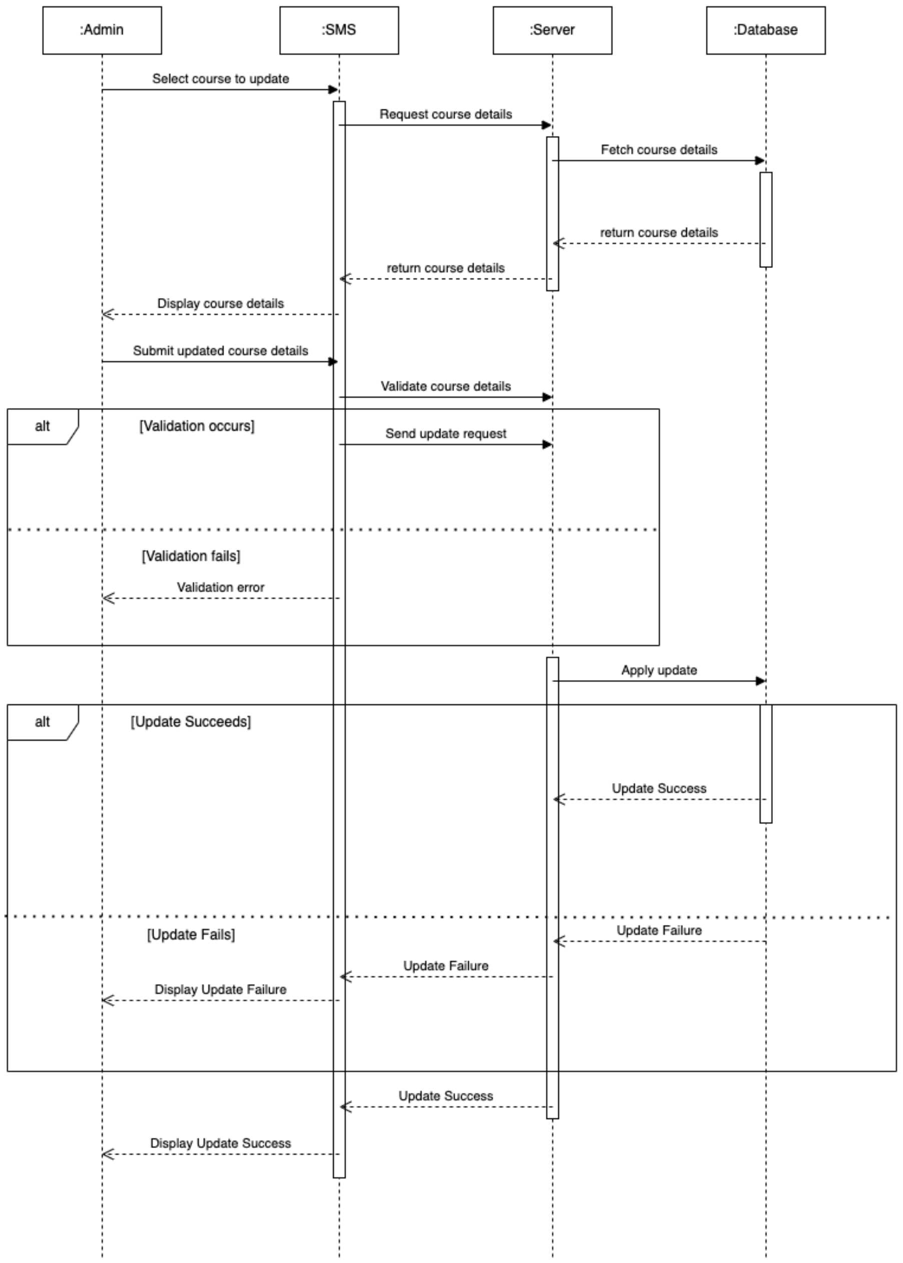
2. The system reflects the updated course details in all relevant views.

Quality Rules:

- The system must validate all required fields before updating a course.
- Duplicate course entries are not allowed.
- Only authorized admins can update course details.



Sequence Diagram: Update Course



3. Use Case: Remove User

Actor: User

Use Case Overview:
A User initiates the process to remove an existing user from the Student Management System (SMS). The system validates the user's existence, processes the removal, and confirms whether the deletion was successful or not.

Subject Area: User Management
Trigger: User selects the option to remove a user.

Preconditions:

1. The User is logged into the SMS with appropriate permissions to remove other users.
2. The SMS is operational and connected to the database.

Description:

1. The User selects the "Remove User" option from the system menu.
2. SMS prompts the User to confirm the removal action.
3. User confirms the removal request.
4. SMS sends a removal request to the server.
5. The server validates if the specified user exists in the database.
6. If the user exists, the server processes the deletion and removes the user's record from the database.
7. The server sends a success or failure response to the SMS.
8. SMS displays the final deletion status to the User.

Alternate Scenarios:
6A1: If the user does not exist:

1. SMS notifies the User that the specified user cannot be found and cancels the removal process.

8A1: If the deletion fails due to a database error:

1. SMS displays an error message indicating the failure.

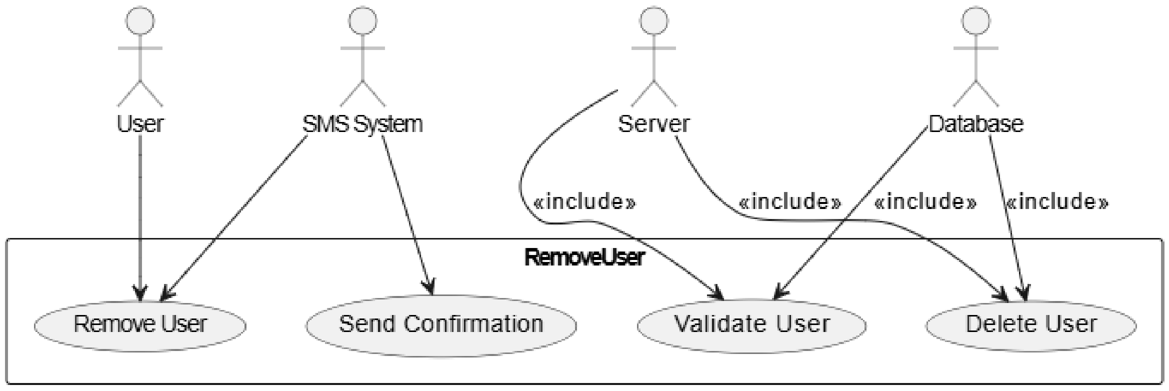
Postconditions:

1. If successful, the user’s record is removed from the database, and the system reflects the updated user list.

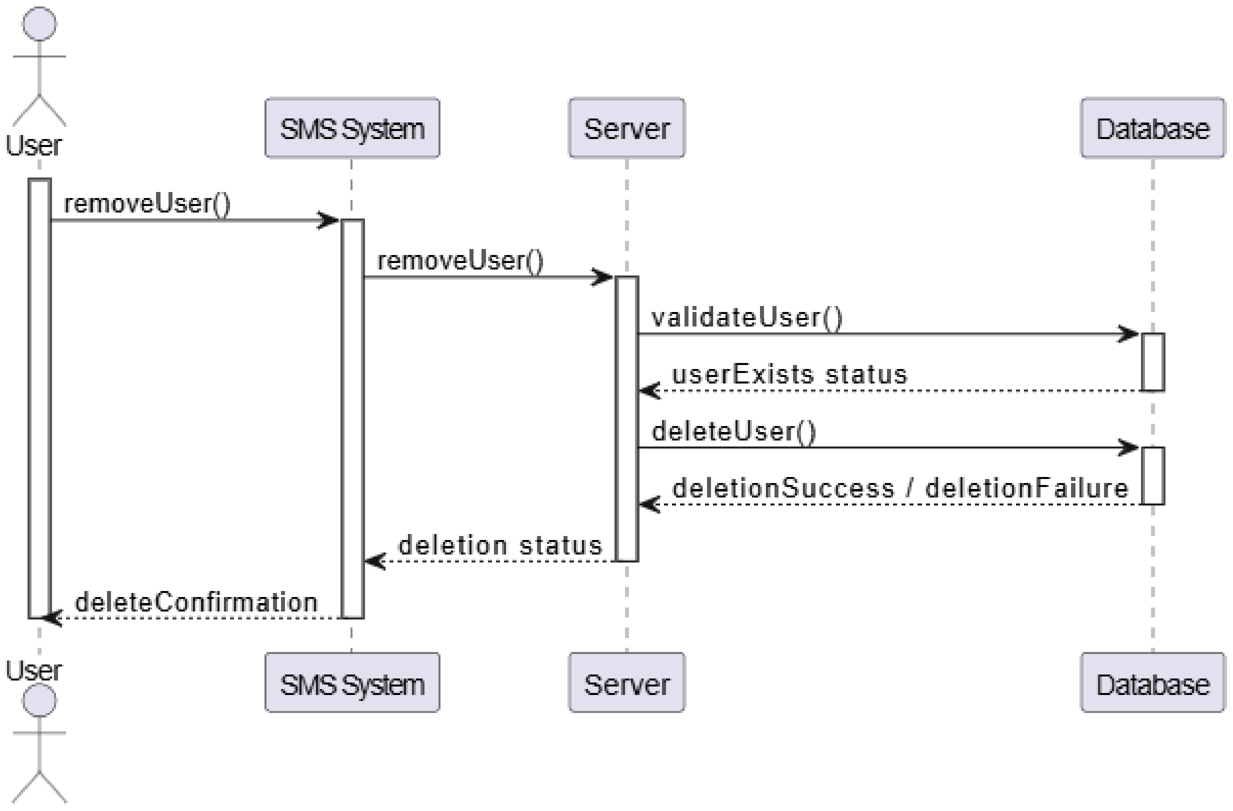
2. If unsuccessful, the database remains unchanged, and an error message is displayed to the User.

Quality Rules:

- Only authorized users can initiate the "Remove User" process.
- The system must validate the existence of the user before attempting deletion.
- Error messages should clearly indicate the cause of failure without exposing sensitive information.



Sequence Diagram: Remove User



4. Use Case: Update User

Actor: User

Use Case Overview:
A User initiates the process to add a new user to the Student Management System (SMS). The User provides details for the new user, which are validated and checked for existence. Based on the outcome, the system either confirms or rejects the addition.

Subject Area: User Management
Trigger: User selects the option to update a user.
Preconditions:
1. User is logged into the SMS with appropriate permissions.
2. The SMS is operational and able to process new user additions.

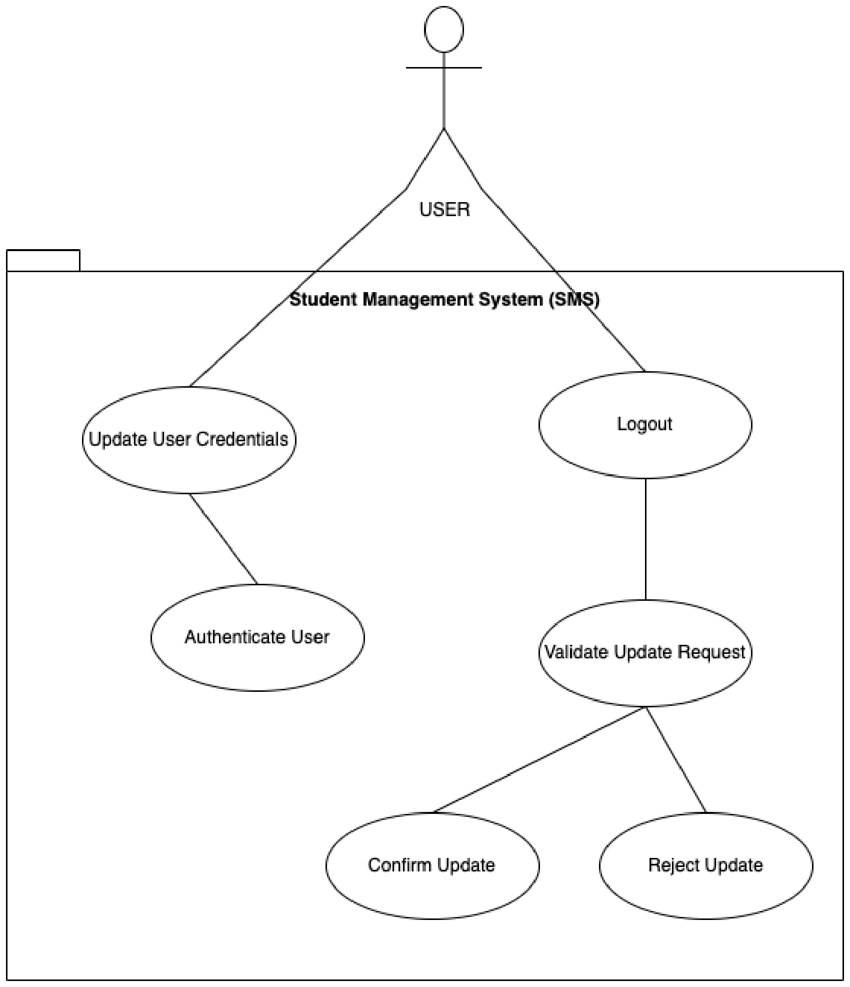
Description:
1. User selects the "Update User" option in the system.
2. SMS displays a form to enter details for the new user.
3. User fills in the required details (name, role, contact information).
4. SMS validates the entered details.
5. SMS checks if a user with the same information already exists.
6. If the user does not exist, SMS processes the addition request.
7. SMS confirms the successful addition of the user and updates the database.
8. Termination: The user is successfully added to the system.

Alternate Scenarios:
6A1: If the validation fails:
   1. SMS displays an error message and highlights the fields requiring correction.
   2. User corrects the details and resubmits.

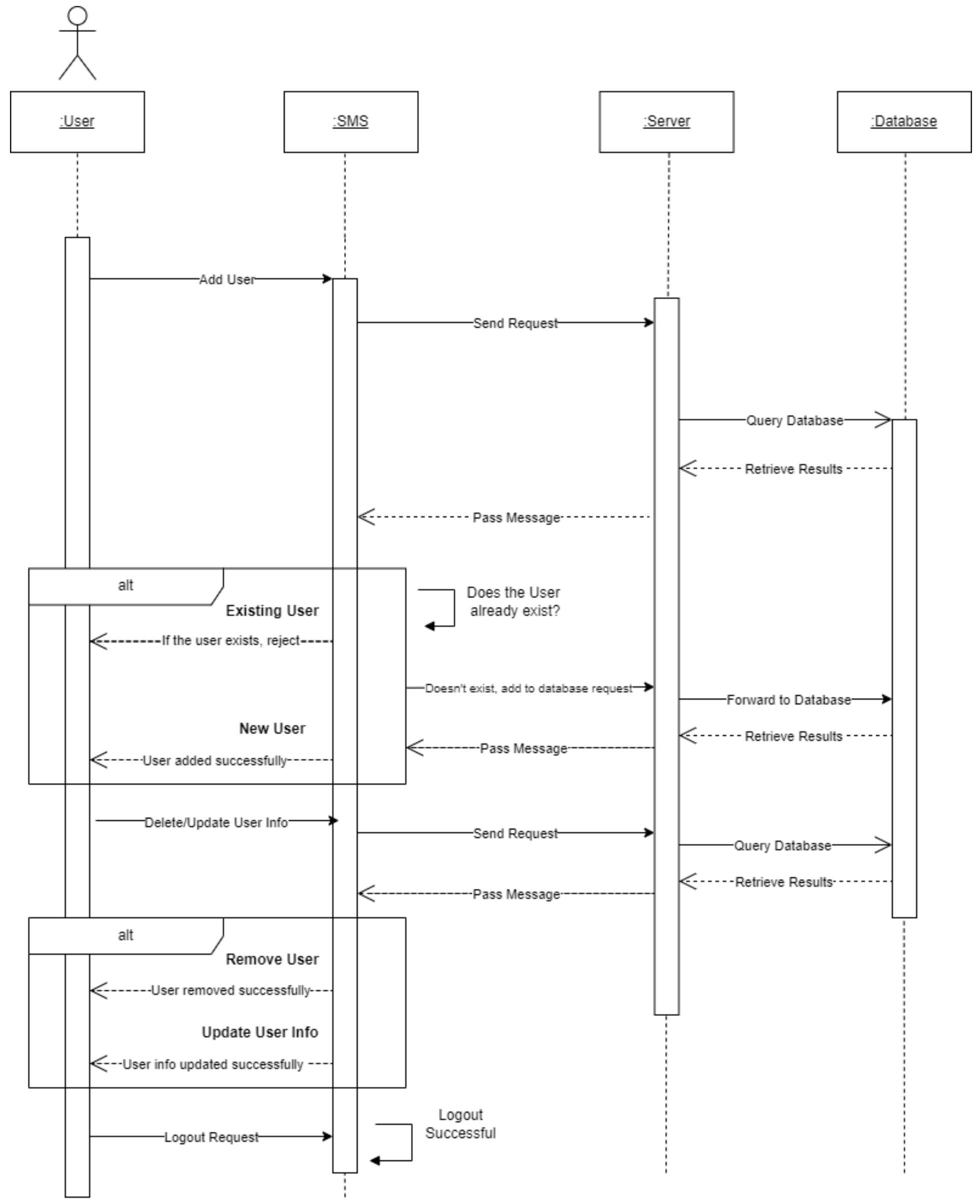
7A1: If the user already exists:
   1. SMS notifies the User of a duplicate entry and rejects the addition.

Postconditions:
1. The new user is successfully added to the system.
2. The database reflects the updated user list.

Quality Rules:
- The system must validate all required fields before adding a user.
- Duplicate entries are not allowed.
- Only authorized users can add new users.


USER
Student Management System (SMS)
Update User Credentials
Logout
Authenticate User
Validate Update Request
Confirm Update
Reject Update

Sequence Diagram: Update User



5. Use Case: Remove Student

Actor: Admin

Use Case Overview:
An Admin initiates the process to remove a student from the system. The system validates the request and ensures the student's records are deleted from the database.

Subject Area: Student Management
Trigger: Admin selects the option to remove a student.
Preconditions:
1. Admin is logged into the SMS with appropriate permissions.
2. The student record to be removed exists in the system.

Description:
1. Admin selects the "Remove Student" option in the system.
2. SMS displays the list of students or a search interface.
3. Admin selects the specific student to remove.
4. SMS validates the removal request.
5. SMS sends a removal request to the database server.
6. The database confirms successful deletion of the student record.
7. SMS notifies the Admin of the successful removal.
8. Termination: The student record is removed from the system.

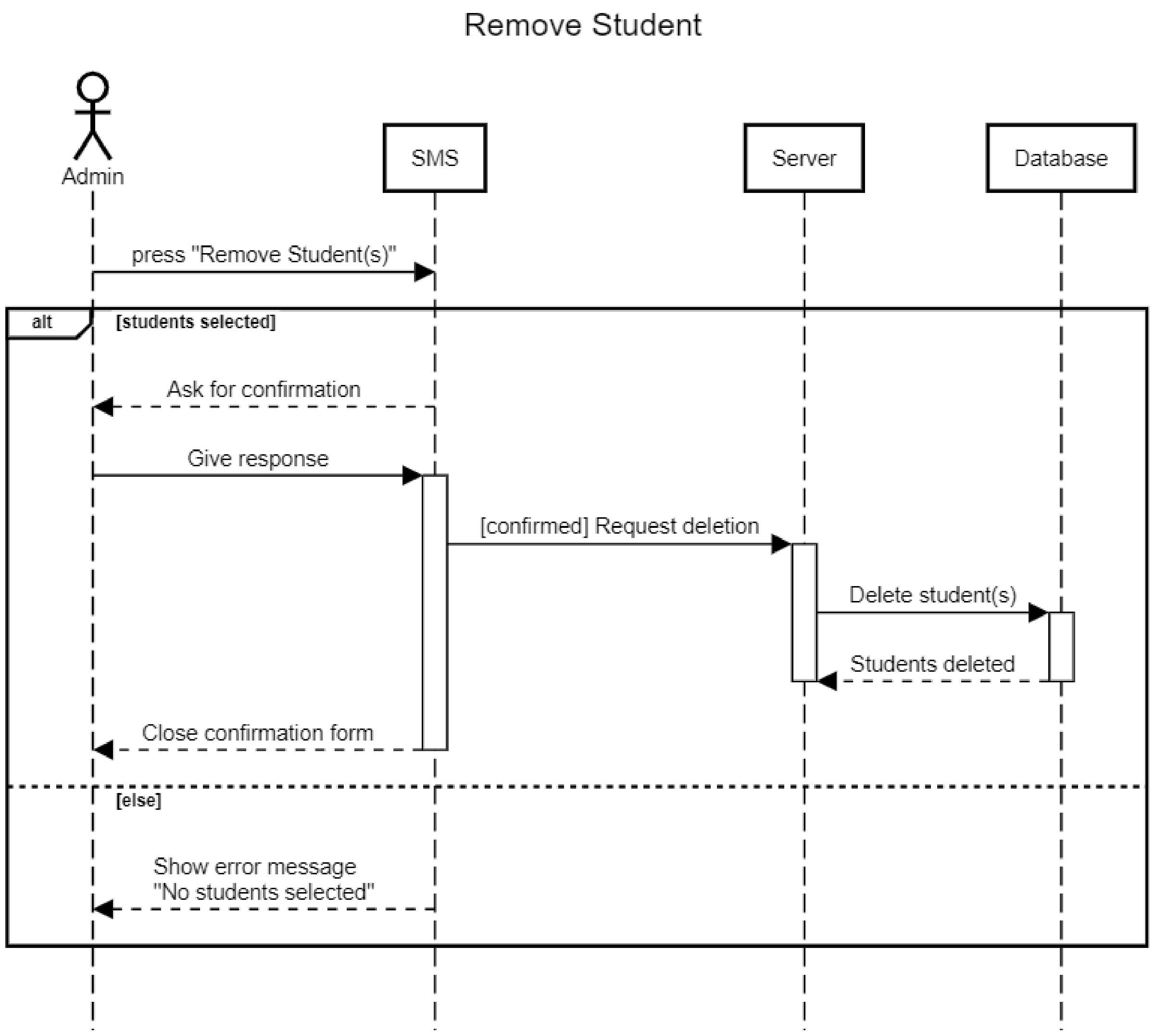
Alternate Scenarios:
4A1: If the validation fails (student ID not found):
   1. SMS notifies the Admin of the validation failure.
   2. Admin reviews the input and resubmits the request.

6A1: If the database deletion fails:
   1. SMS notifies the Admin of the failure.
   2. Admin retries the removal process.

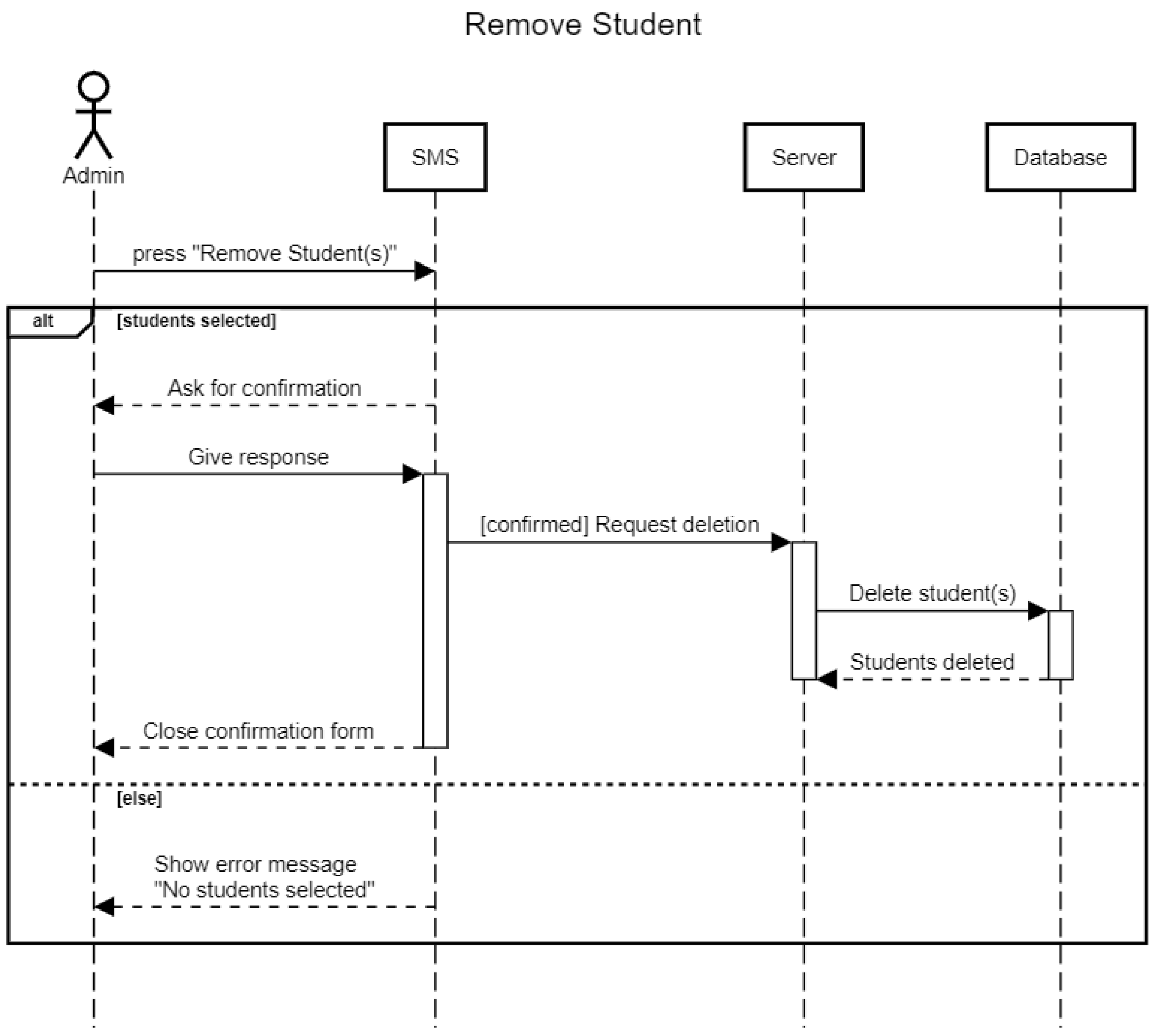
Postconditions:
1. The student record is successfully deleted from the database.
2. The updated student list is reflected in the system.

Quality Rules:
- The system must ensure that no dependent data (course enrollments) is left orphaned.

- The system must notify the Admin of success or failure promptly.
- Only Admins with the appropriate permissions can remove students.



Sequence Diagram: Remove Student



6. Use Case: Update Student

Actor: User

Use Case Overview:
A User initiates the process to update their information in the system. The system validates the provided updates and ensures the user is authenticated before applying the changes.

Subject Area: User Management
Trigger: User selects the option to update their information.
Preconditions:
1. User is logged into the SMS with valid credentials.
2. The user’s account exists in the system.

Description:
1. User selects the "Update User Info" option in the system.
2. SMS displays the current details of the user.
3. User edits the required fields (email, password, contact details).
4. SMS validates the updated details.
5. SMS sends an update request to the system server.
6. The server authenticates the user and validates the update request.
7. The system updates the database with the new user details.
8. SMS notifies the User of the successful update.
9. Termination: The user's information is updated in the system.

Alternate Scenarios:
4A1: If the updated details do not pass validation:
   1. SMS displays an error message indicating the invalid fields.
   2. User corrects the details and resubmits.

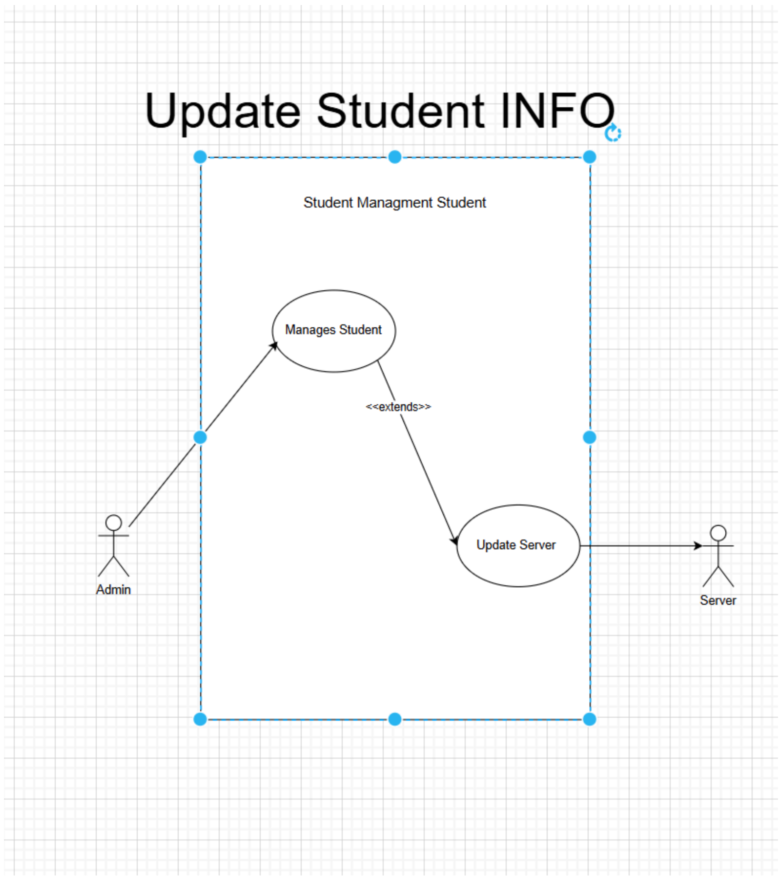
6A1: If the server authentication fails:
   1. SMS notifies the User of authentication failure.
   2. User re-authenticates and retries the update process.

7A1: If the database update fails:
   1. SMS notifies the User of the failure.
   2. The update process is terminated.

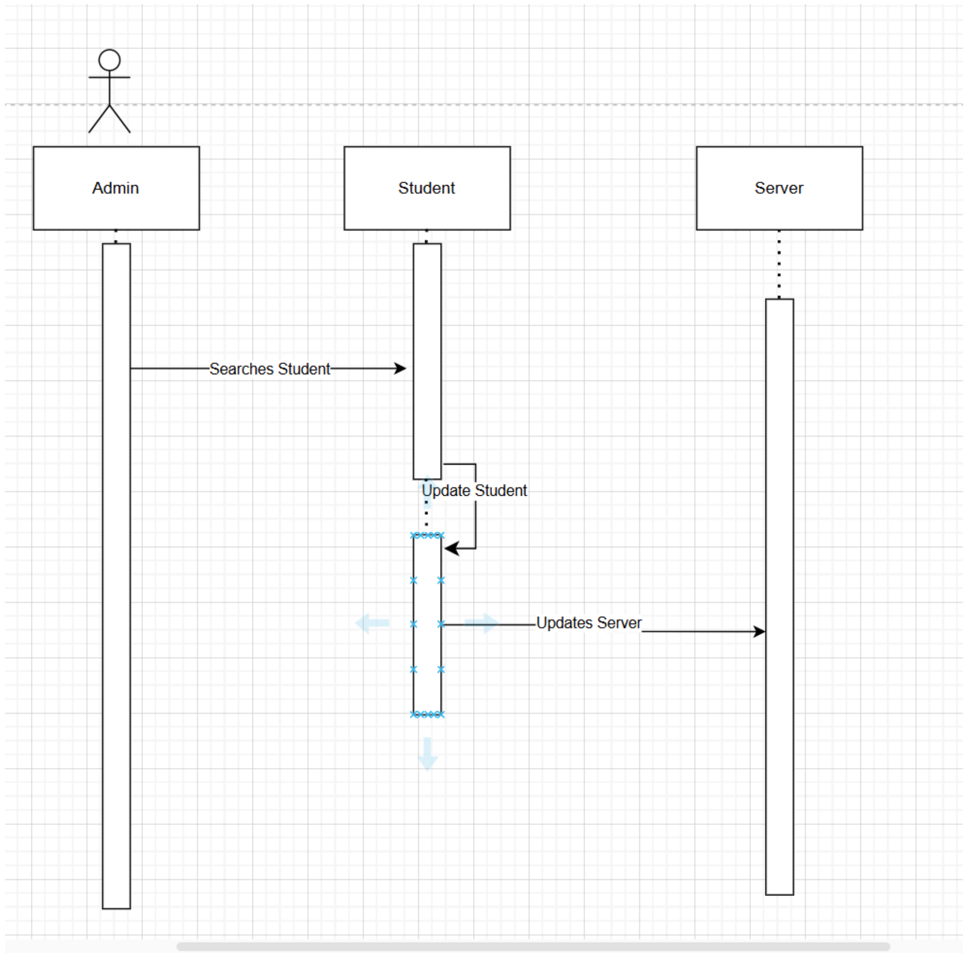
Postconditions:
1. The user's details are successfully updated in the database.

2. The updated information is reflected across the system.
Quality Rules:

- Only authenticated users can update their information.
- All required fields must be validated before submitting updates.
- The system must notify the User of success or failure promptly.



Sequence Diagram: Update Student



7. Use Case: Login

Actor: Administrator

Use Case Overview:
An Administrator logs into the system to gain access to its features. The system authenticates the provided credentials and verifies the Administrator's role.

Subject Area: User Authentication
Trigger: Administrator initiates the login process.
Preconditions:
1. Administrator has valid credentials (username and password).
2. The SMS is operational.

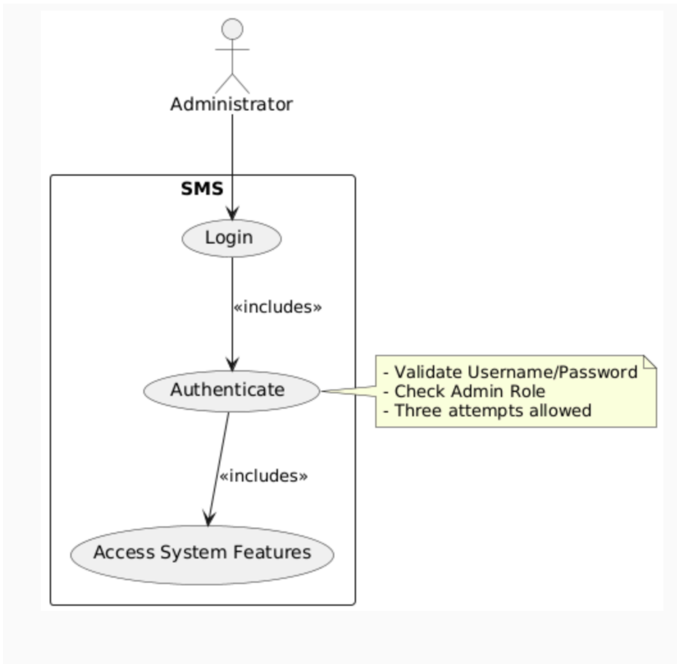
Description:
1. Administrator enters their username and password in the login form.
2. SMS validates the entered credentials.
3. SMS checks the Administrator's role and access permissions.
4. If authentication is successful, SMS grants access to the system features.
5. Termination: Administrator is successfully logged into the system.

Alternate Scenarios:
2A1: If the credentials are invalid:
  1. SMS notifies the Administrator of invalid login credentials.
  2. Administrator is prompted to re-enter the correct details.

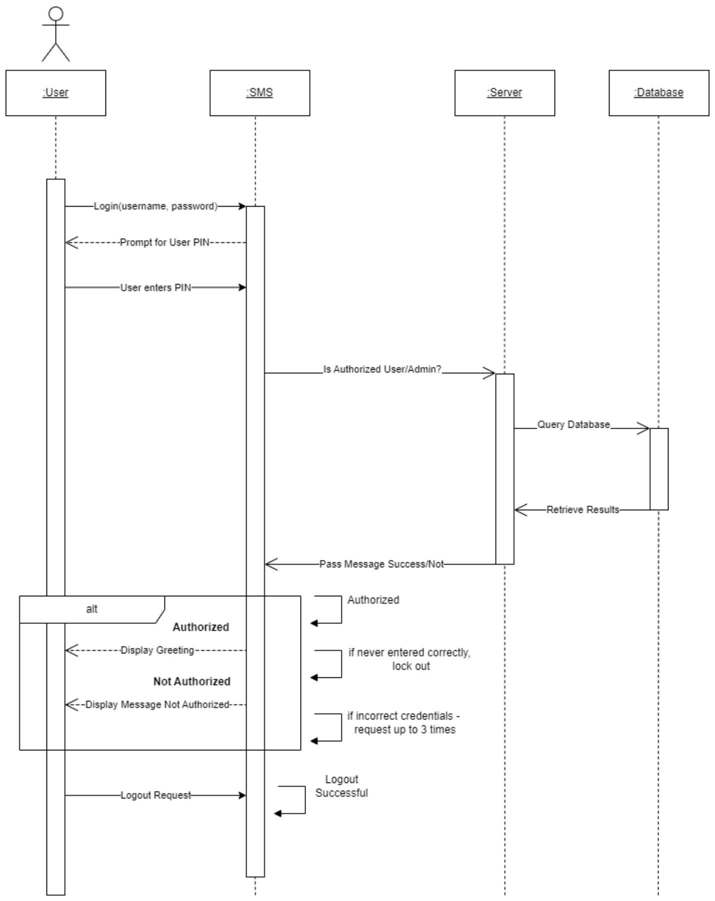
3A1: If the Administrator exceeds the maximum allowed login attempts:
  1. SMS locks the account temporarily.
  2. Administrator must reset the password to regain access.

Postconditions:
1. Administrator gains access to the system's features.
2. The session is active and ready for further actions.

Quality Rules:
- Only users with valid credentials can log into the system.
- A maximum of three failed login attempts is allowed before locking the account.
- The system must securely handle user credentials to prevent unauthorized access.



Sequence Diagram: Login



8. Use Case: Logout

Actor: Administrator

Use Case Overview:
An Administrator logs out of the system to terminate their session and ensure no unauthorized access. The system securely ends the session and clears any active data.

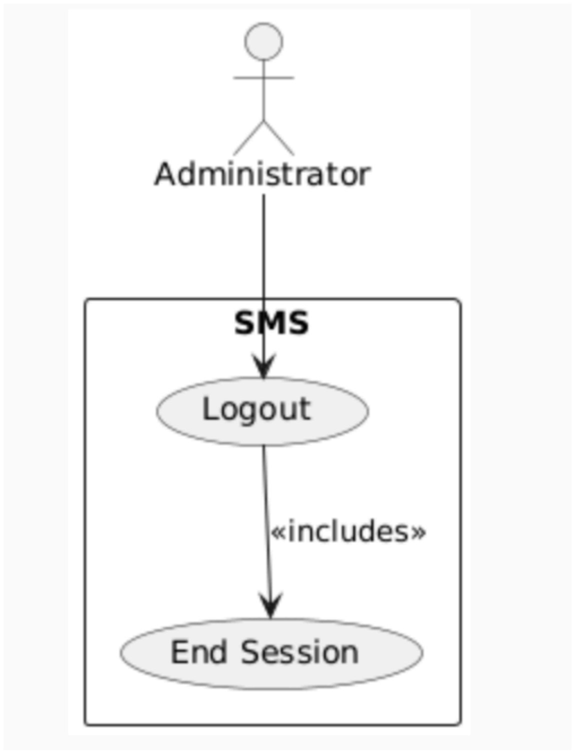
Subject Area: Session Management
Trigger: Administrator selects the "Logout" option.
Preconditions:
1. Administrator is logged into the system.
2. An active session exists for the Administrator.

Description:
1. Administrator selects the "Logout" option in the system.
2. SMS validates the logout request.
3. SMS ends the active session for the Administrator.
4. SMS clears any cached data related to the session.
5. SMS notifies the Administrator of a successful logout.
6. Termination: The session is securely ended, and the Administrator is returned to the login screen.

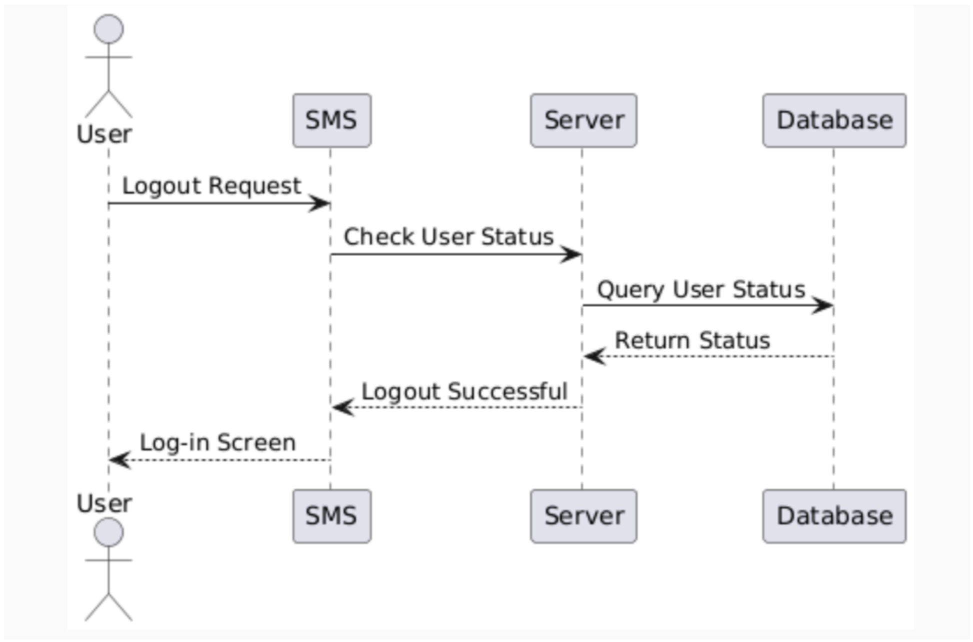
Alternate Scenarios:
2A1: If the logout request fails due to a system error:
   1. SMS notifies the Administrator of the failure.
   2. Administrator retries the logout process.

Postconditions:
1. The Administrator's session is securely terminated.
2. The system is ready for the next login attempt.

Quality Rules:
- The system must ensure all session data is cleared upon logout.
- Administrators must not be able to perform any actions after logging out.
- The logout process must notify the user of success or failure promptly.



Sequence Diagram: Logout



9. Use Case: View Student

Actor: User

Use Case Overview:
A User initiates the process to search and view a student's information by providing the student's name or ID. The system processes the request, retrieves the relevant data, and displays the results based on the search criteria.

Subject Area: Student Information Management
Trigger: User selects the option to search for a student.

Preconditions:

1. The User is logged into the Student Management System (SMS) with appropriate permissions.
2. The SMS is operational and connected to the database.

Description:

1. User selects the "Search Student" option in the system.
2. SMS displays a search form for entering a student's name or ID.
3. User enters the search criteria (name or ID) and submits the search request.
4. SMS sends the search request to the server.
5. The server queries the database using the provided search criteria.
6. The database retrieves the results and sends them back to the server.
7. The server forwards the results to the SMS.
8. SMS displays the retrieved student information to the User.

Alternate Scenarios:
8A1: If the student is not found:

1. SMS displays a "Student Not Found" message to the User.

Postconditions:

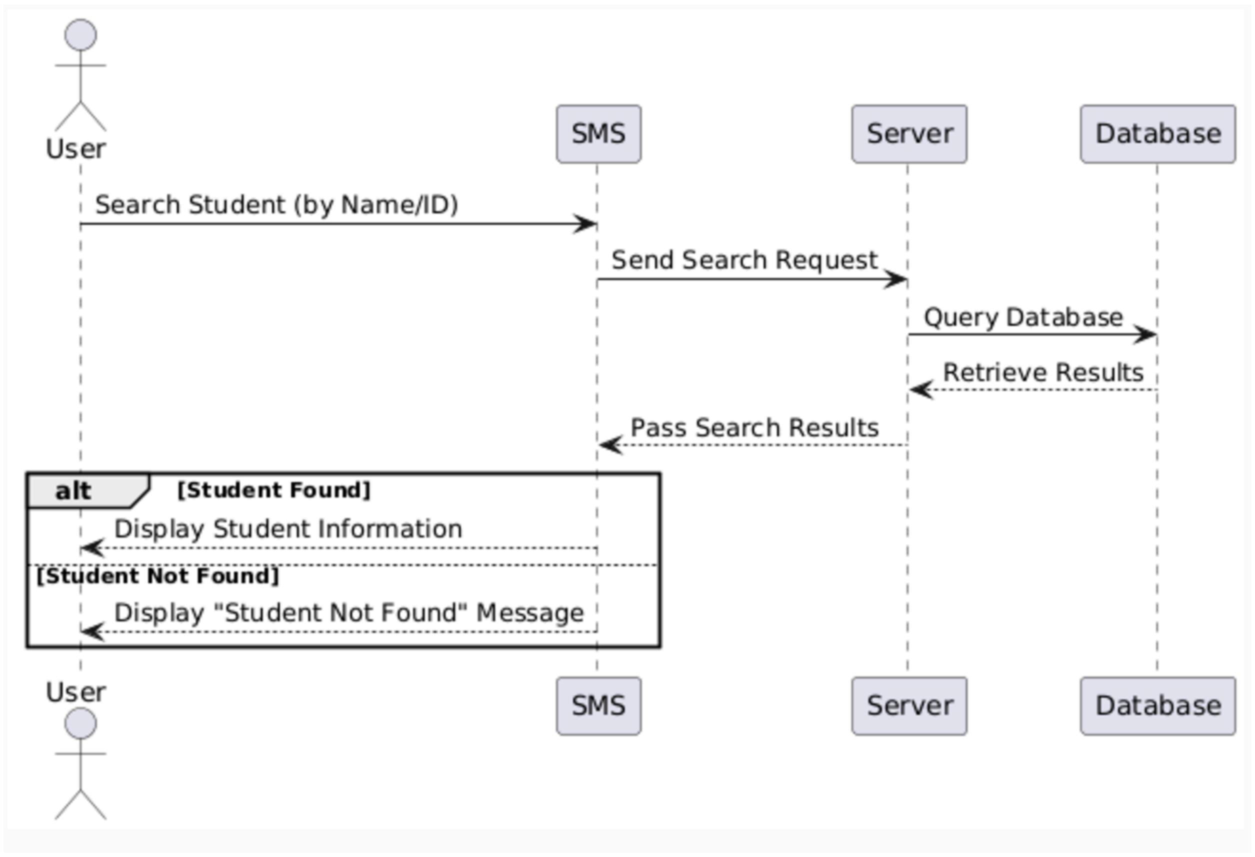
1. The retrieved student information is displayed to the User if the student exists.
2. If no matching student is found, the system provides feedback indicating that the search was unsuccessful.

Quality Rules:

- The system must validate the search input (e.g., ensure the ID or name is in the correct format).

- The system must handle cases where no data is found and provide appropriate feedback.
- Only authorized users can view student information.

Sequence Diagram



VI. Glossary

- *Administrator*: A user with elevated permissions who manages the Student Management System (SMS), including adding, updating, and deleting records.
- *Authentication*: The process of verifying a user's identity through login credentials (username and password).
- *Database*: A structured collection of data used to store student records and system information securely.
- *Dependency*: A relationship between system components, where one component relies on another for functionality.
- *Entity*: A distinct object in the system, such as a student record, user, or course.
- *Field Validation*: Ensuring that the data entered into a form meets specified criteria (e.g., required fields, valid email format).
- *Functional Requirements*: The specific features and capabilities that the system must include to meet its objectives.
- *Non-Functional Requirements*: Attributes of the system, such as performance, security, scalability, and usability, that affect its quality and operation.
- *Object Diagram*: A diagram showing the static structure of system objects and their relationships at a specific time.
- *Role-Based Access Control (RBAC)*: A security feature that restricts system access based on user roles and permissions.
- *Scalability*: The ability of the system to handle increased workload or data without performance degradation.
- *Sequence Diagram*: A diagram that visualizes the sequence of interactions between system components for a specific use case.
- *State Diagram:* A diagram that illustrates the various states a system or object can be in and the transitions between those states.
- *Student ID*: A unique identifier assigned to each student in the system to ensure distinct records.
- *System Requirements*: The conditions and capabilities needed for the successful operation of the SMS, including functional and non-functional aspects.
- *Trigger*: An event or action that initiates a use case or process within the system.
- *Use Case*: A description of how a user interacts with the system to achieve a specific goal.
- *User*: Any individual who interacts with the SMS, including administrators, faculty, and students.
- *Validation*: The process of checking the correctness and completeness of data entered into the system.