



**COMSATS University Islamabad, Abbottabad
Campus**

Department of Computer Science

Project Proposal
Student Management System

Group Members

Nimra Jadoon (FA22-BSE-011)

Zoya Kayani (FA22-BSE-042)

Saud ur Rehman (FA22-BSE-048)

CHAPTER 1 PROJECT PROPOSAL

Introduction

The Student Management System (SMS) is a simple yet efficient application designed to manage student data through a graphical user interface (GUI). It allows users to add, view, update, and delete student records seamlessly. The system supports the entry of basic student details, such as ID, name, and age, and organizes this data in a list that is easy to navigate.

The system is divided into four main sections: **Add Student**, **View Students**, **Update Student**, and **Delete Student**. Each section is represented by a tab in the interface, making it easy to switch between different functionalities. The interface is designed with user-friendliness in mind, with clear labels, buttons, and input fields.

The background color for each panel is thoughtfully chosen to provide a visually appealing experience while the buttons are styled to enhance user interaction. The **Add Student** panel lets you input a student's details and adds the data to the system, while the **View Students** tab shows a list of all students in the system. The **Update Student** and **Delete Student** panels allow modification or removal of student data based on the student ID.

The application also ensures data persistence within the runtime, maintaining a list of students, and allows for efficient updates or deletions. Overall, this Student Management System is a simple, yet effective tool for organizing and managing student records in any educational institution or setting.

Vision:

The vision of the Student Management System (SMS) is to provide an intuitive, efficient, and user-friendly platform for managing student information. The goal is to simplify administrative tasks, such as student record creation, updates, and deletions, while maintaining accurate, organized, and easily accessible data. By enabling seamless data entry and retrieval, the system aims to reduce manual workload, enhance productivity, and improve overall management efficiency in educational institutions.

The SMS aspires to be a reliable tool that supports the growth of educational institutions by allowing staff and administrators to focus more on student development and less on administrative overhead. It envisions offering a centralized hub for managing student information, helping to streamline processes and improve decision-making based on easily accessible data. Ultimately, the vision is to empower educational institutions with a digital solution that fosters transparency, collaboration, and continuous improvement in student management practices.

Functional Requirements of the Student Management System (SMS):

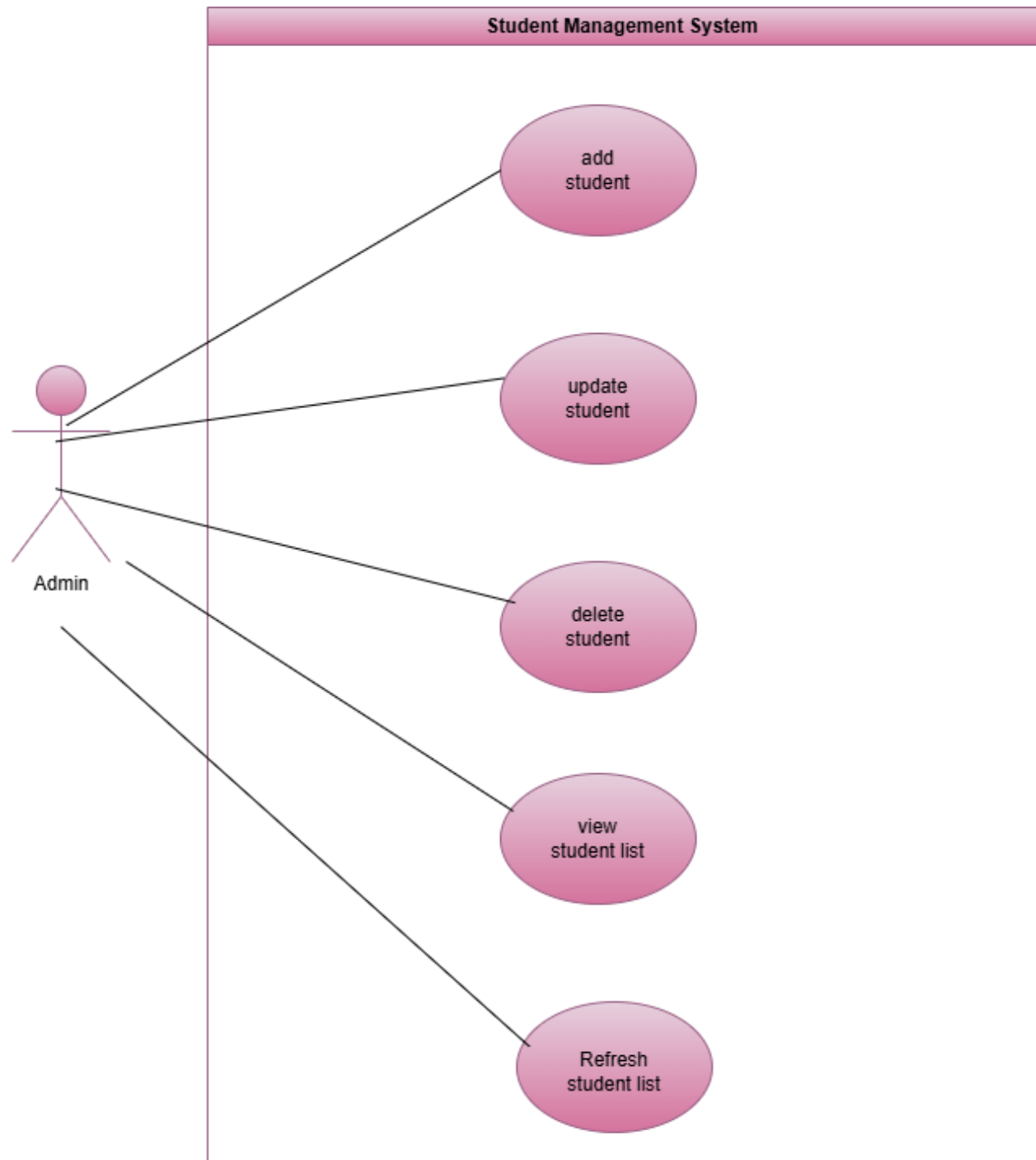
1. **Student Registration:** The system must allow administrators to add new student records, including details such as student ID, name, and age.
2. **Student Record Update:** The system must allow administrators to update the information of existing students (e.g., changing the student's name or age).
3. **Student Record Deletion:** The system must allow administrators to delete a student record by entering the student's unique ID.
4. **View Student Information:** The system must allow administrators to view a list of all students, displaying relevant details such as ID, name, and age.
5. **Refresh Student List:** The system should allow administrators to refresh the student list after adding, updating, or deleting a student record.
6. **Data Validation:** The system must validate user inputs to ensure all fields are filled out correctly (e.g., numeric values for age, non-empty name, and ID).

Non-Functional Requirements of the Student Management System (SMS):

1. **Usability:** The system should have a simple, easy-to-use interface, ensuring administrators can efficiently perform tasks with minimal effort.
2. **Performance:** The system should be responsive, handling tasks such as adding, updating, and deleting student records quickly.
3. **Reliability:** The system should function consistently without crashing or losing data when operations are performed (adding, updating, deleting, or viewing).
4. **Maintainability:** The system should be easy to maintain, with clear code structure and minimal dependencies, making future updates and bug fixes straightforward.

CHAPTER 2 USE CASES

2.1 Use Case Diagram



2.2 Use Cases Distribution

S#.	Group Member	Assigned Use Cases
1	<Nimra Jadoon> <FA22-BSE-011>	UC 1 : Add Student UC 2 : View Student List
2	<Zoya Kayani> <FA22-BSE-042>	UC 3 : Update Student UC 4 : Refresh Student List
3	<Saud ur Rehman> <FA22-BSE-048>	UC 5: Delete Student

2.3 Brief Level Use Cases

Nimra Jadoon:

Use case: Add Student

The **admin** can add a new student to the system by entering the student's information, including their ID, name, and age. After the admin fills in the details and submits the form, the system registers the new student in the database. This action updates the list of students, allowing the admin to manage them effectively.

Use case: View Student List

The **admin** can view the list of all students in the system. This list displays the ID, name, and age of each student. It allows users to quickly get an overview of the students enrolled in the system. Admins can access this list to manage students, while students can view their own details.

Zoya Kayani:

Use case: Update Student

The **admin** is responsible for updating the details of an existing student. If any information like the student's name or age needs to be changed, the admin can search for the student by their ID,

make the necessary changes, and save the updated data. This ensures that the system maintains accurate and current student records.

Use case: Refresh Student List

The **admin** can refresh the student list to ensure the displayed information is up to date. This is particularly important after adding, updating, or deleting students, as it provides a real-time view of the system's current state. This keeps the system dynamic and ensures that the admin works with the latest data.

Saud ur Rehman:

Use case: Delete Student

The **Admin** can remove a student from the system by entering their unique student ID. This action deletes the student's information from the system and updates the student list to reflect the removal. It helps maintain an accurate database by removing students who are no longer part of the institution.

2.4 Fully Dressed Use Cases

Use Case 1: Add Student

- **Use Case Name:** Add Student
- **Scope:** Student Management System
- **Level:** User Goal
- **Primary Actor:** Admin
- **Stakeholders and Interest:**
 - **Admin:** Wants to add new students to the system for managing their data.
 - **Student:** Will have their information stored in the system.
- **Preconditions:** Admin must be logged in to the system.
- **Success Guarantee:** The student is successfully added to the system, and the student data is saved.
- **Main Success Scenario:**
 1. Admin navigates to the "Add Student" page.
 2. Admin enters the student's details (e.g., ID, name, age).

3. Admin clicks the "Add" button.
 4. The system validates the input and saves the student details.
 5. The system confirms successful addition and updates the student list.
- **Exceptions:**
 1. If required fields are left empty, the system prompts the admin to fill them.
 2. If the student ID already exists, the system displays an error message.
 - **Special Requirements:**
 - The system should ensure that duplicate student IDs are not allowed.
 - **Frequency of Occurrence:** Occurs whenever a new student is added to the system.
 - **Miscellaneous:** The system should allow quick validation and addition of students.
-

Use Case 2: Update Student

- **Use Case Name:** Update Student
- **Scope:** Student Management System
- **Level:** User Goal
- **Primary Actor:** Admin
- **Stakeholders and Interest:**
 - **Admin:** Wants to update existing student information in the system.
 - **Student:** Their information is updated in the system.
- **Preconditions:** Admin must be logged in, and the student must already exist in the system.
- **Success Guarantee:** The student data is updated, and the changes are reflected in the system.
- **Main Success Scenario:**
 1. Admin navigates to the "Update Student" page.
 2. Admin enters the student ID to search for the student.

3. The system displays the student's current details.
 4. Admin updates the necessary details (e.g., name, age).
 5. Admin clicks the "Update" button, and the system saves the changes.
 6. The system confirms successful update.
- **Exceptions:**
 1. If the student ID does not exist, the system displays an error message.
 2. If fields are incorrectly filled, the system prompts the admin to correct the input.
 - **Special Requirements:**
 - The system should ensure that only valid data (e.g., valid age) is accepted.
 - **Frequency of Occurrence:** Occurs whenever a student's details need to be modified.
 - **Miscellaneous:** The system should provide an option to revert changes before confirming the update.
-

Use Case 3: Delete Student

- **Use Case Name:** Delete Student
- **Scope:** Student Management System
- **Level:** User Goal
- **Primary Actor:** Admin
- **Stakeholders and Interest:**
 - **Admin:** Wants to remove students from the system.
 - **Student:** Their data will be deleted from the system.
- **Preconditions:** Admin must be logged in, and the student must exist in the system.
- **Success Guarantee:** The student is removed from the system.
- **Main Success Scenario:**
 1. Admin navigates to the "Delete Student" page.

2. Admin enters the student ID.
3. The system verifies the student ID exists.
4. Admin clicks the "Delete" button.
5. The system removes the student from the database.
6. The system confirms the deletion.

- **Exceptions:**

1. If the student ID is not found, the system displays an error message.

- **Special Requirements:**

- The system should ask for confirmation before deleting a student to prevent accidental deletion.

- **Frequency of Occurrence:** Occurs when a student is no longer part of the system.

- **Miscellaneous:** The system should have a backup feature to prevent permanent loss of student data.
-

Use Case 4: View Student List

- **Use Case Name:** View Student List
- **Scope:** Student Management System
- **Level:** User Goal
- **Primary Actor:** Admin, Student

- **Stakeholders and Interest:**

- **Admin:** Wants to view all the students in the system.
 - **Student:** Can view their own details, but not others.

- **Preconditions:** User (Admin or Student) must be logged in.

- **Success Guarantee:** The system displays the list of students or specific student details.

- **Main Success Scenario:**

1. User navigates to the "View Student List" page.

2. The system displays the list of all students or the student's own data (if logged in as a student).
 3. User can scroll through the list or search for specific students.
 - **Exceptions:**
 1. If no students are in the system, the system displays a message saying "No students found."
 - **Special Requirements:**
 - The system should display students' details in an easy-to-read list format.
 - **Frequency of Occurrence:** Occurs regularly when users want to see student data.
 - **Miscellaneous:** The system should provide sorting and filtering options to easily navigate through the list.
-

Use Case 5: Refresh Student List

- **Use Case Name:** Refresh Student List
- **Scope:** Student Management System
- **Level:** User Goal
- **Primary Actor:** Admin
- **Stakeholders and Interest:**
 - **Admin:** Wants to refresh the student list to see the most up-to-date data.
 - **Student:** No direct involvement in refreshing the list.
- **Preconditions:** Admin must be logged in and viewing the student list.
- **Success Guarantee:** The student list is updated and shows the most current data.
- **Main Success Scenario:**
 1. Admin clicks the "Refresh" button on the student list page.
 2. The system retrieves the latest student data.
 3. The system updates and displays the refreshed list of students.
- **Exceptions:**

1. If there is an issue with the data retrieval, the system displays an error message.

- **Special Requirements:**

- The refresh process should be quick and efficient, with no delays.

- **Frequency of Occurrence:** Occurs whenever the admin wants to view the most up-to-date student information.

- **Miscellaneous:** The system should auto-refresh periodically in case of new data additions

2.4 Prototypes

Student Management System

— □ ×

Add Student View Students Update Student Delete Student

ID:

Name:

Age:

Add Student

Student Management System

Add Student

View Students

Update Student

Delete Student

123: nimra, Age: 20

333: zoya, Age: 20

44: saud, Age: 20

Refresh List

Student Management System

Add Student

View Students

Update Student

Delete Student

Enter ID to Update:

New Name:

New Age:

Update Student

Student Management System

Add Student


View Students

Update Student

Delete Student

Enter ID to Delete:

Message

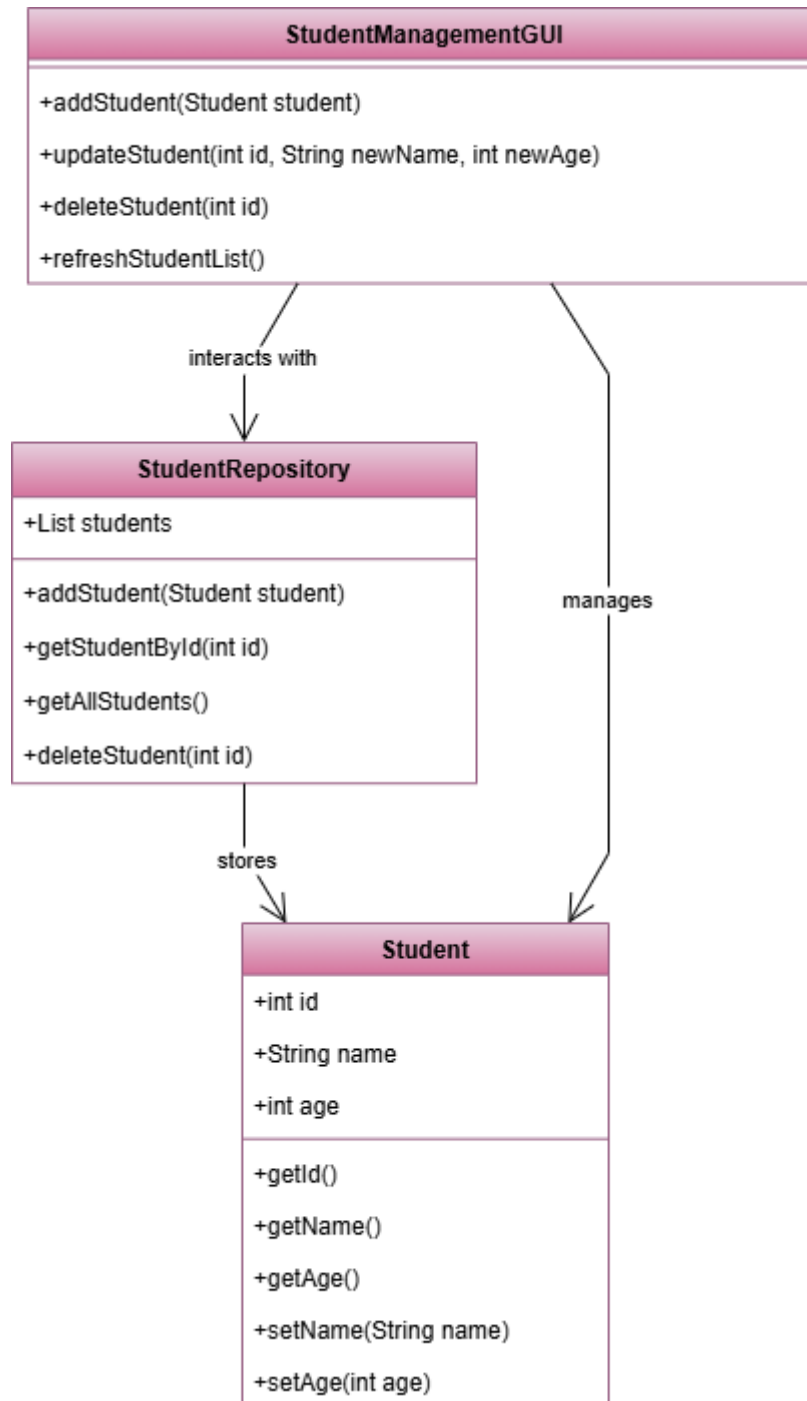


Student deleted successfully!

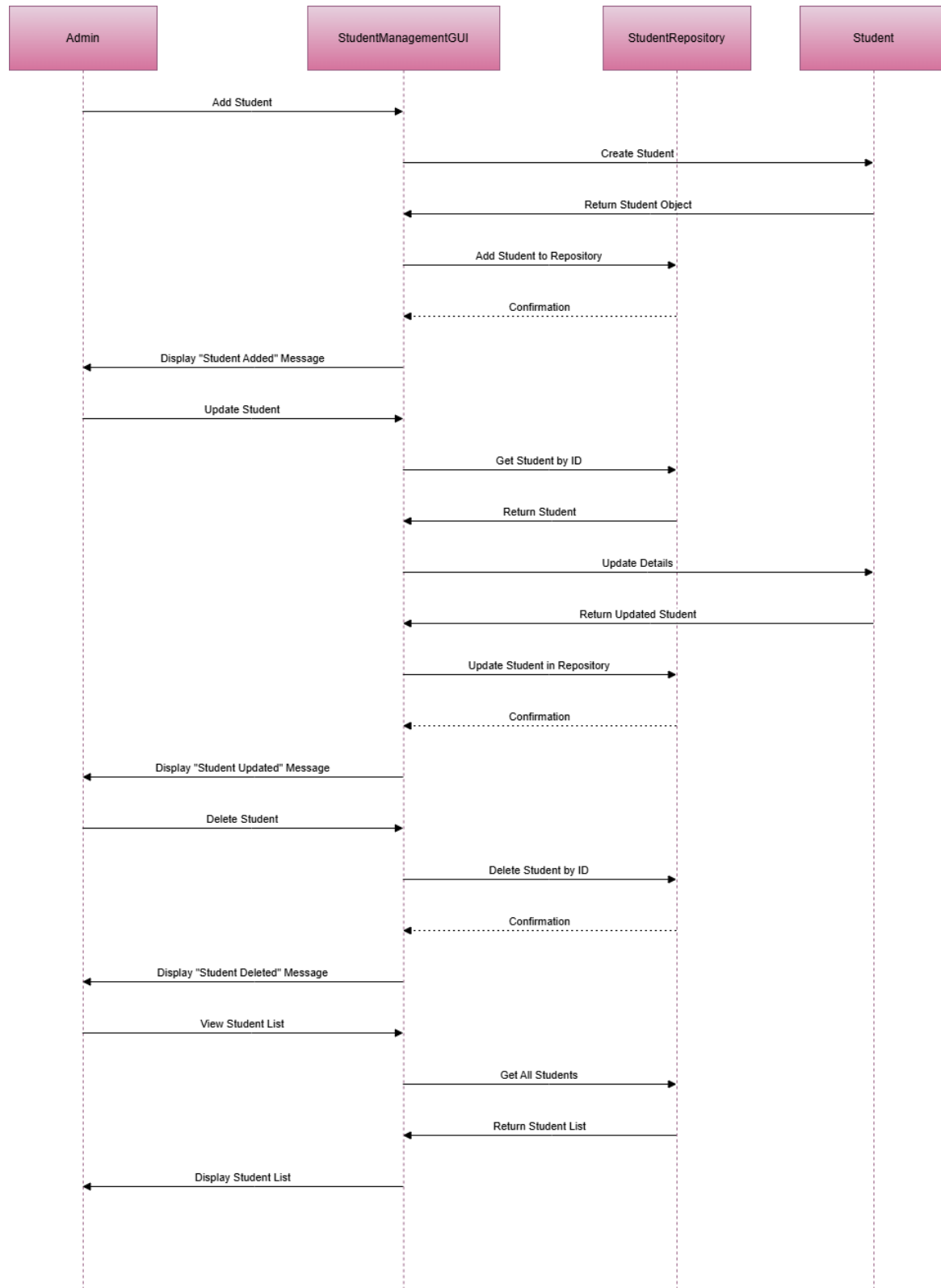
OK

Delete Student

CHAPTER 3: CLASS DIAGRAM



CHAPTER 4: SEQUENCE DIAGRAM



CHAPTER 5: PACKAGE DIAGRAM

