Laboratory Assignment AND Assessment Requirements Specification

Version 1.0

March, 2023

Developed by:

Astăluș Adrian, Apăvăloaei Teodor

931

Version History

| Version | Description of Change | Author | Date |
| --- | --- | --- | --- |
| V01 | Initial | Astăluș Adrian  Apăvăloaei Teodor | 07.03.2023 |

**Contents**

[Laboratory Assignment AND Assessment Requirements Specification 1](#_heading=h.gjdgxs)

[Version 1.0 1](#_heading=h.30j0zll)

[March, 2023 1](#_heading=h.1fob9te)

[1 Functional Requirements 3](#_heading=h.2et92p0)

[2 Actors 3](#_heading=h.tyjcwt)

[3 Use cases – diagram 3](#_heading=h.3dy6vkm)

[3.1 Use case number 1 (Description of the use case) 3](#_heading=h.1t3h5sf)

[4 Analysis 3](#_heading=h.4d34og8)

[4.1 Entities 3](#_heading=h.2s8eyo1)

[4.2 Relations between entities 3](#_heading=h.17dp8vu)

[4.3 Attributes 3](#_heading=h.3rdcrjn)

[4.4 System behavior 4](#_heading=h.26in1rg)

[4.4.1 Use case 1-2-3 4](#_heading=h.lnxbz9)

[4.5 System events 4](#_heading=h.35nkun2)

[5 Design 4](#_heading=h.1ksv4uv)

**Analysis and design Document**

# Functional Requirements

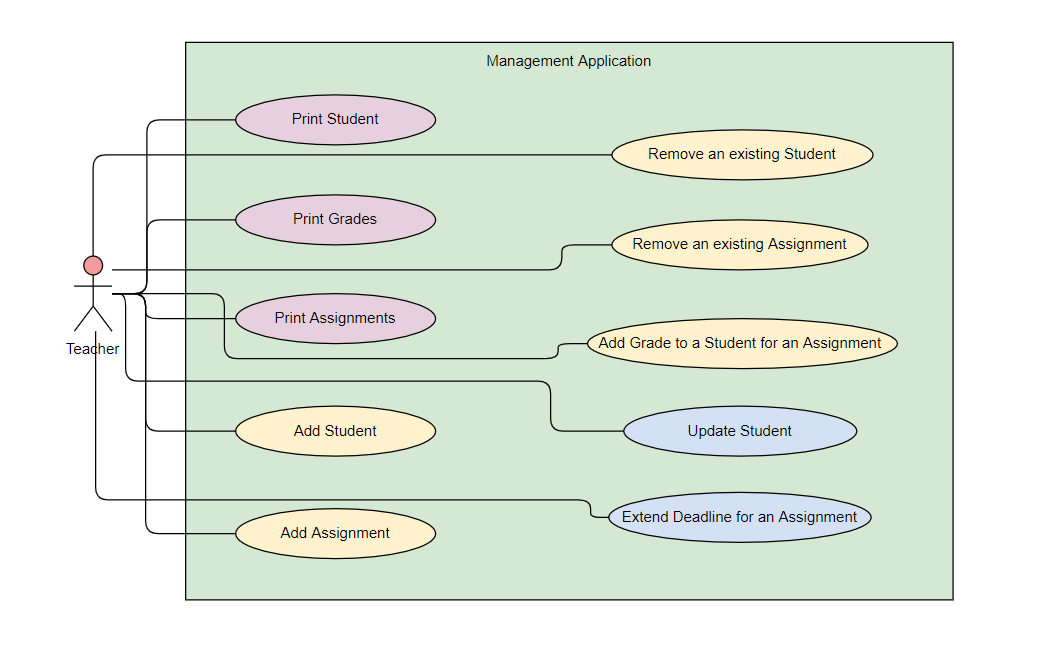
List the functional requirements (FR) of the system.

| Section/ Requirement ID | Requirement Definition |
| --- | --- |
| FR 1.0 | Add a new student |
| FR 1.1 | Remove a student |
| FR 1.2 | Update student |
| FR 1.3 | Print all students |
| FR 2.0 | Add a new assignment |
| FR 2.1 | Extend deadline for an assignment |
| FR 2.2 | Remove an existing assignment |
| FR 2.3 | Print all assignments |
| FR 3.0 | Add a grade to a student for an assignment |
| FR 3.1 | Print all grades |

# Actors

Teacher

# Use cases – diagram



## Use case number 1 (Description of the use case)

Actors: teacher

Description: add a new student

Precondition: - all fields are specified

Postcondition: - a new student was added in the list

| Action | System Response |
| --- | --- |
| 1 Completes the necessary fields for adding |  |
|  | 2 Checks if everything is alright, adds a new element in the list if so |
| 3 - | 3. If the input is invalid, throws an exception |

Exceptions: When the fields aren’t filled.

## Use case number 2 (Description of the use case)

Actors: teacher

Description: delete a student

Precondition: - valid id belonging to an existing student is specified

Postcondition: - the student with the specified id is removed from the list

| Action | System response |
| --- | --- |
| 1 Give an id as input |  |
|  | 2 Checks if it is a valid id and there is a student with that id and deletes the student |
| 3 - | 3. If the input is invalid, throws an exception |

## Use case number 3 (Description of the use case)

Actors: teacher

Description: update a student

Precondition: - valid id belonging to an existing student and all other fields for student are specified

Postcondition: - the student with the specified id has the data updated

| Action | System response |
| --- | --- |
| 1 Give an id and all other fields for the Student entity as input |  |
|  | 2 Checks if it is a valid id and there is a student with that id, than checks if the rest of the input is valid, and updates the data for that student |
| 3 - | 3. If the input is invalid, throws an exception |

## 

## 3.4 Use case number 4 (Description of the use case)

Actors: teacher

Description: add an assignment

Precondition: - all the required fields are specified

Postcondition: - a new assignment is added to the list of assignments

| Action | System response |
| --- | --- |
| 1 Give an id and all other fields for the Assignment entity as input |  |
|  | 2 Checks if all the fields are introduced correctly. If so, a new assignment is added |
| 3 - | 3. If the input is invalid, throws an exception |

## 3.5 Use case number 5 (Description of the use case)

Actors: teacher

Description: remove an existing assignment

Precondition: - a valid assignment id is introduced

Postcondition: - the assignment with the corresponding id is removed

| Action | System response |
| --- | --- |
| 1 Give an id as input |  |
|  | 2 Checks if the id is valid and if there exists an assignment with that id. If so, it deletes the assignment |
| 3 - | 3. If the input is invalid, throws an exception |

## 3.6 Use case number 6 (Description of the use case)

Actors: teacher

Description: extending the deadline of an assignment

Precondition: - a valid assignment id is introduced and the number of weeks to extend the assignment is specified

Postcondition: - the assignment’s deadline is extended by the specified number of weeks

| Action | System response |
| --- | --- |
| 1 Give an id as input and the number of weeks of extension |  |
|  | 2. Checks if there exists an assignment with the given id and adds the number of weeks given to the deadline if possible |
| 3 - | 3. If the input is invalid, throws an exception |

## 

## 

## 

## 

## 3.7 Use case number 7 (Description of the use case)

Actors: teacher

Description: add a grade to a student for an assignment

Precondition: - a valid student id and assignment id are introduced, as well as all the required field for a grade entity

Postcondition: - a new grade entity is added to the list of grades

| Action | System response |
| --- | --- |
| 1 Give an assignment id, a student id, and specify all the required fields for a grade entity |  |
|  | 2. Checks if both ids are correct, then a new grade is added |
| 3 - | 3. If the input is invalid, throws an exception |

## 3.8 Use case number 8 (Description of the use case)

Actors: teacher

Description: list all students

Precondition: -

Postcondition: all the students are displayed

| Action | System response |
| --- | --- |
| 1 The value for the functionality is introduced |  |
|  | 2. If the correct value is introduced, the list of students is printed |
| 3 - | 3. If the input is invalid, throws an exception |

## 3.9 Use case number 9 (Description of the use case)

Actors: teacher

Description: list all assignments

Precondition: -

Postcondition:

| Action | System response |
| --- | --- |
| 1 The value for the functionality is introduced |  |
|  | 2. If the correct value is introduced, the list of assignments is printed |
| 3 - | 3. If the input is invalid, throws an exception |

## 

## 

## 3.10 Use case number 10 (Description of the use case)

Actors: teacher

Description: list all grades

Precondition: -

Postcondition: -

| Action | System response |
| --- | --- |
| 1 The value for the functionality is introduced |  |
|  | 2. If the correct value is introduced, the list of grades is printed |
| 3 - | 3. If the input is invalid, throws an exception |

# Analysis

## Entities

Student, Assignment, Grade

## Relations between entities

One student can have multiple assignments and one assignment can be assigned to many students. It is a many-to-many relationship between the two classes. Class Grade has an id - a pair consisting of studentId and assignmentId and it is the association class between the Student and Assignment classes. A student can have only one grade for a specific assignment.

## Attributes

Student: id, name, group, email, professor name

Assignment: id, description, deadline, assignation date

Grade: id(studentId, assignmentId), value, delivery date, feedback

## System behavior

## Use case 1-2-3

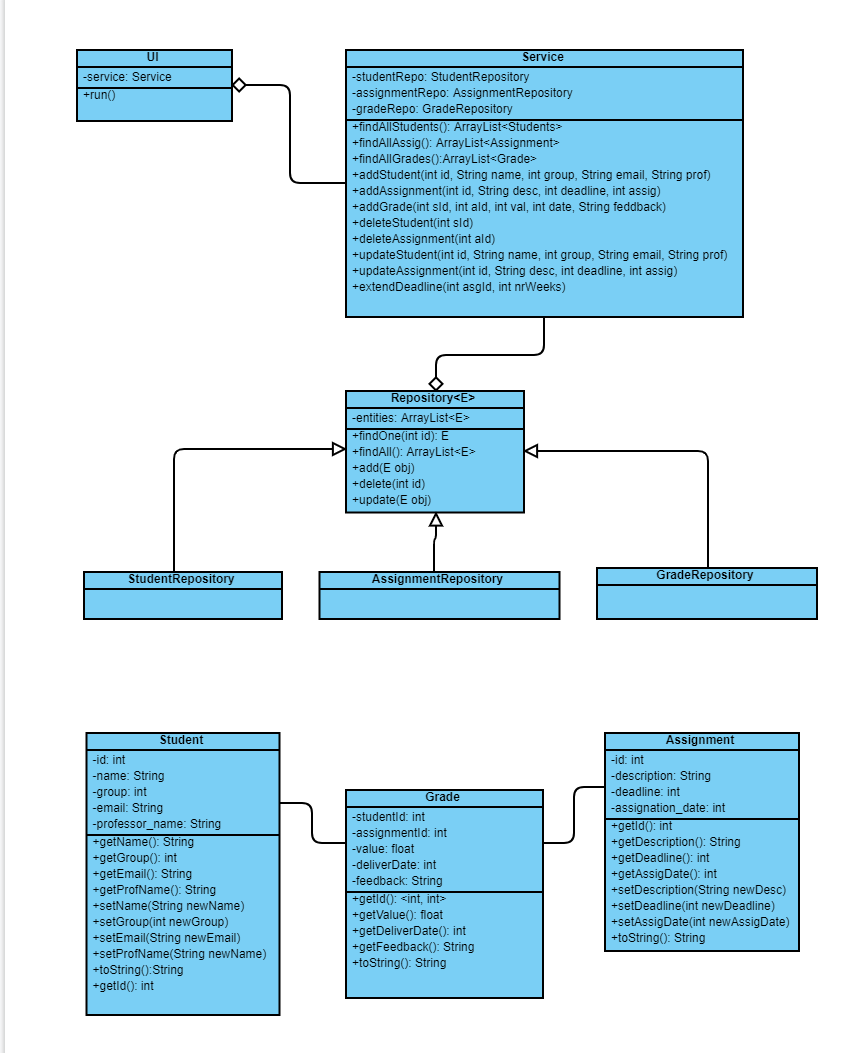
The system will act as a subsystem to a larger environment, in order to speed up a certain process in the company’s workflow.

## System events

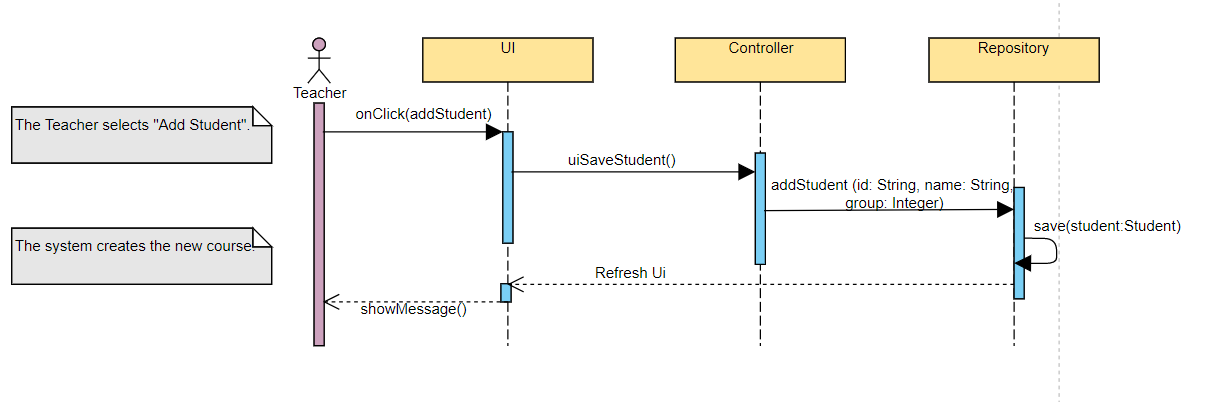
After each operation a message is shown to the user either if the command terminated successfully or with an error message.

# Design

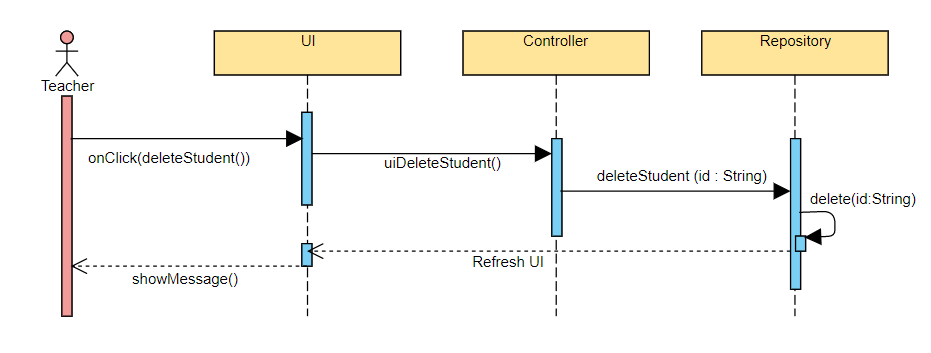
* 1. **Class diagram**

****

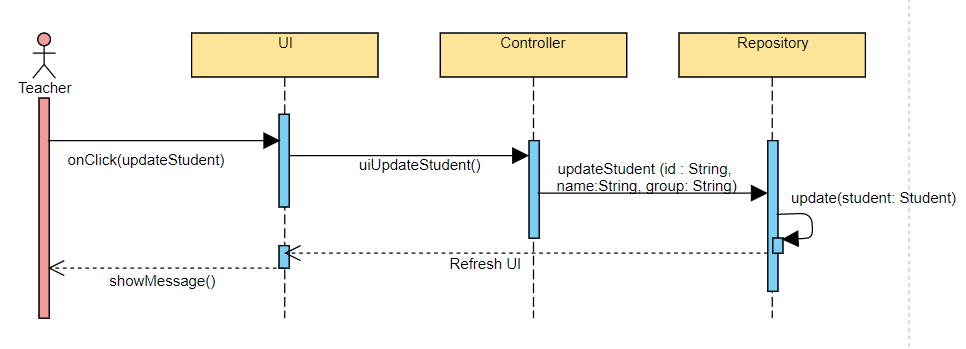
* 1. **Sequence diagrams (for each use case)**
* **Add Student Sequence Diagram**

****

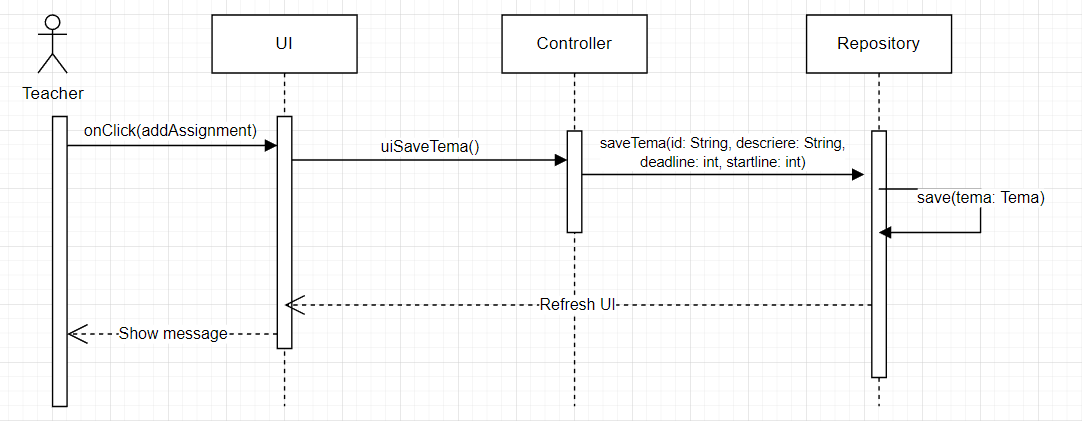
* **Delete Student Sequence Diagram**

****

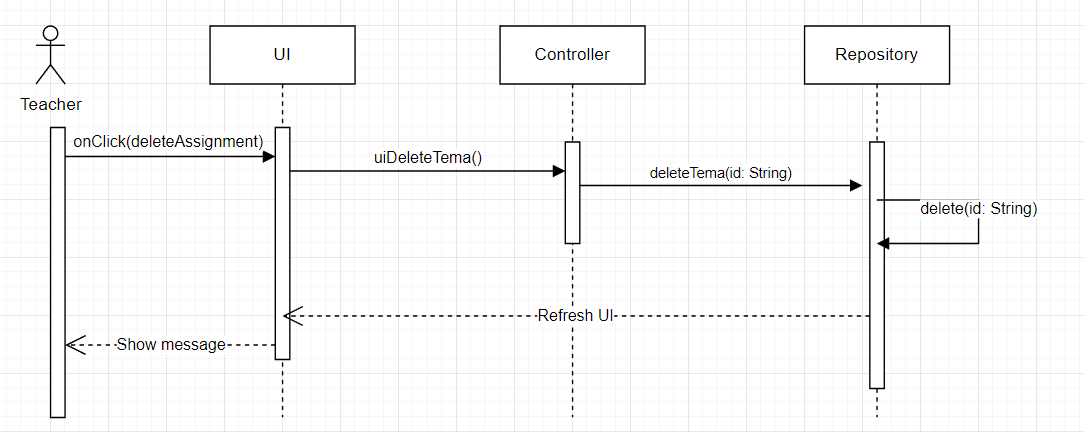
* **Update Student Sequence Diagram**

****

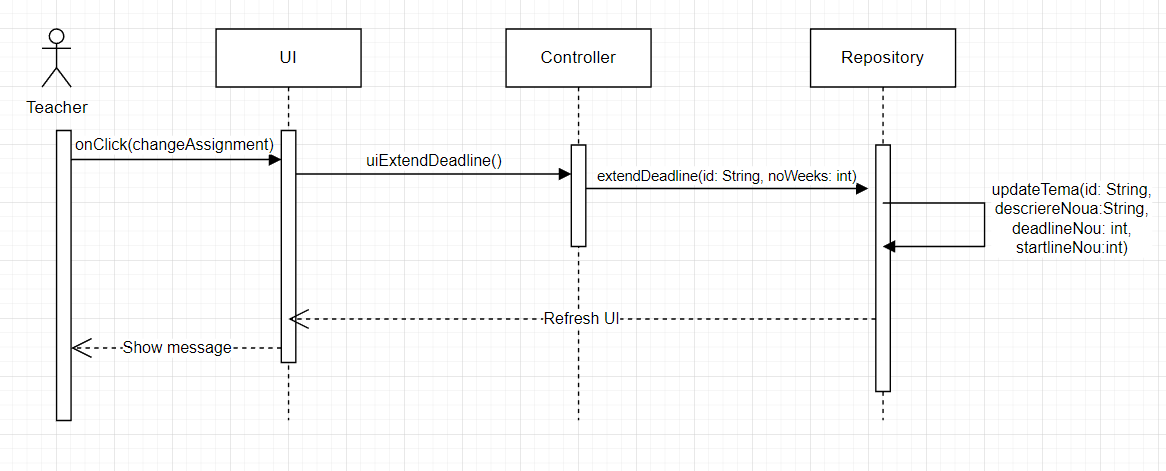
* **Add Assignment Sequence Diagram**

****

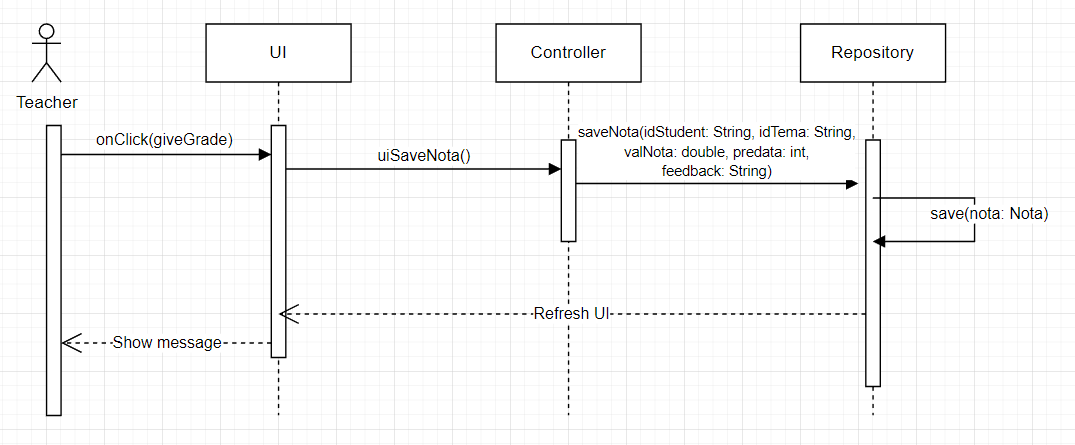
* **Delete Assignment Sequence Diagram**

****

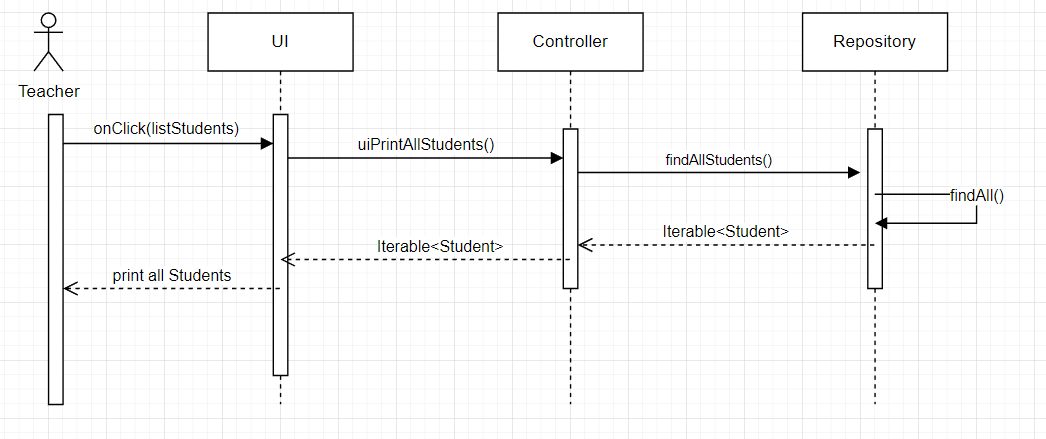
* **Change Assignment Deadline Sequence Diagram**

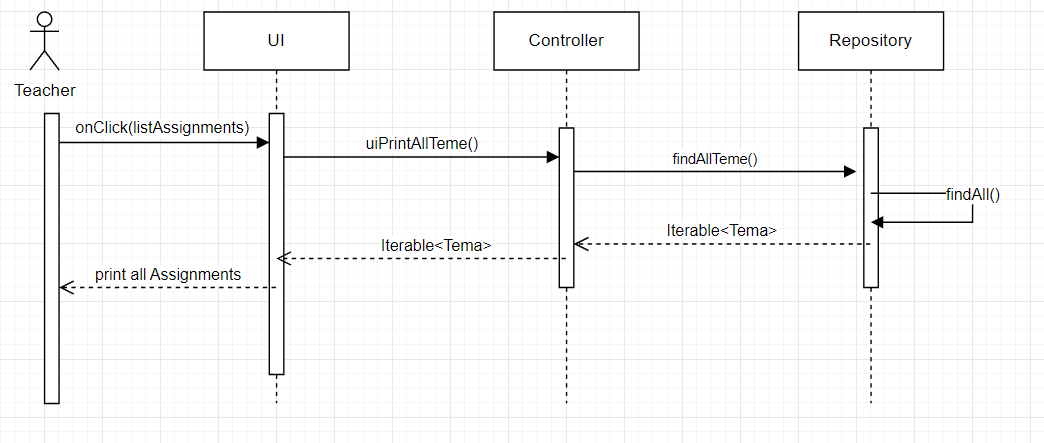
****

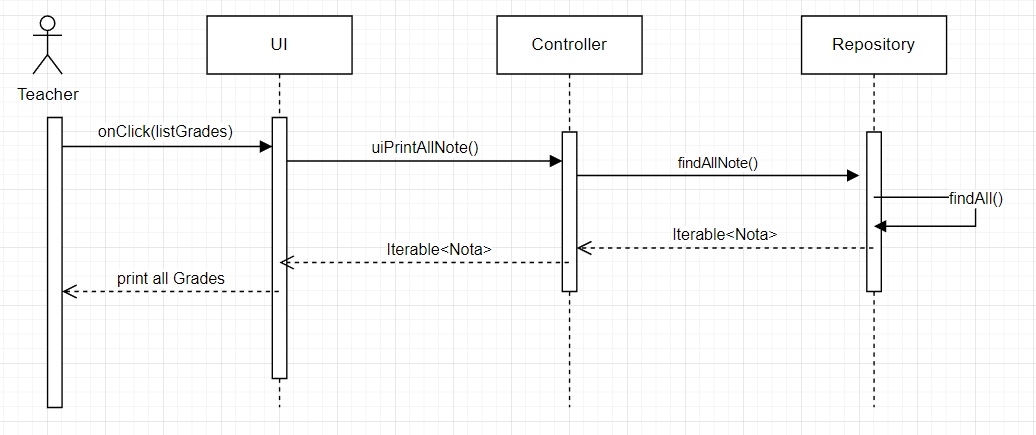
* **Give Grade to Student for Assignment Sequence Diagram**

****

* **List All Students Sequence Diagram**

****

* **List All Assignments Sequence Diagram**
* **List All Grades Sequence Diagram**

****

* 1. **GRASP**

GRASP is set of exactly 9 **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns:

1. Information Expert

2. Creator

3. Controller

4. Low Coupling

5. High Cohesion

6. Indirection

7. Polymorphism

8. Pure Fabrication

9. Protected Variations