Laboratory Assignment AND Assessment Requirements Specification

Version 1.0

March, 2020

Developed by:

Hriscu Octavia, Hutupasu Stefana, Ianos Raluca

933

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description of Change | Author | Date |
| V01 | Initial | Hriscu Octavia, Hutupasu Stefana,  Ianos Raluca | 14.03.2023 |
|  |  |  |  |

**Contents**

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

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

[March, 2020 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 |
| FR1.0 | Implement CRUD operations for the Student entity |
| FR1.1 | Adding a laboratory theme |
| FR2.0 | Adding a grade for a particular student to a laboratory topic; |
| FR3.0 | Extending the term of delivery for an existing subject |
| FR4.0 | When adding a new laboratory theme, as well as modifying the delivery date of a theme, all students will be notified by email |
| FR5.0 | The NameStudent.txt file (or its content) will be emailed to the student, weekly, with the subject "Feedback laboratory MAP". |
| FR6.0 | The delays will not be considered if the student has motivation. Also, if the teacher did not enter the notes in time, it will be possible to specify the week in which the subject was delivered. |
| FR7.0 | Filtering entities based on criteria. |

# Actors

Teacher

# Use cases – diagram



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

Actors: teacher

Description: create 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.

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

Actors: teacher

Description: delete 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 |

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

Actors: teacher

Description: update 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: create a new assignment

Precondition: - all fields are specified

Postcondition: - a new assignment 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 |

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

Actors: teacher

Description: create a new grade for student

Precondition: - all fields are specified

Postcondition: - a new grade 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 |

# 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 as id, a pair consisting of studentId and assignmentId and it is the association class between the Student and Assignment classes.

## Attributes

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

Assignment: id, description, deadline, assignation date

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

## System behavior

## Use case 1-2-3-4-5

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 succesfully 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**

****

* 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