Laboratory Assignment AND Assessment Requirements Specification

Version 1.0

19 March, 2023

Developed by:

Brinza Alina, Hostiuc Robert

931

Version History

| **Version** | **Description of Change** | **Author** | **Date** |
| --- | --- | --- | --- |
| V01 | Initial/Modification of document | Brinza Alina  Hostiuc Robert | 19 March 2023 |
| V02 | Completion of document | Brinza Alina  Hostiuc Robert | March 2023 |

**Contents**

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

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

19 [March, 2023 1](#_heading=h.lnxbz9)

[1 Functional Requirements 3](#_heading=h.1fob9te)

[2 Actors 3](#_heading=h.3znysh7)

[3 Use cases – diagram 3](#_heading=h.2et92p0)

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

[4 Analysis 3](#_heading=h.3dy6vkm)

[4.1 Entities 3](#_heading=h.1t3h5sf)

[4.2 Relations between entities 3](#_heading=h.4d34og8)

[4.3 Attributes 3](#_heading=h.2s8eyo1)

[4.4 System behavior 4](#_heading=h.17dp8vu)

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

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

[5 Design 4](#_heading=h.26in1rg)

**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 |
| FR2.0 | Manage laboratory themes and subjects. |
| FR2.1 | Extend the deadline for an existing subject |
| FR2.2 | Add a new laboratory theme |
| FR2.3 | Notify students by email when adding a new laboratory theme or modifying the delivery date of an existing subject |
| FR2.4 | Add a grade to a particular laboratory theme to a particular student |
| FR2.5 | Save theme number, delivery week, deadline and feedback for each added grade |
| FR2.6 | Automatically compute the delay for a grade |
| FR2.7 | Unsubscribe from notifications |
| FR3.0 | Filter students based on different criteria |
| FR4.0 | Generate reports |
| FR4.1 | Send weekly reports by email |
| FR4.2 | Report: laboratory grade for each student |
| FR4.3 | Report: hardest theme (smallest average of grades) |
| FR4.4 | Report: students who can enter the exam |
| FR4.5 | Report: students who have delivered all the themes on time |
| FR5.0 | Give the possibility of having no delay if the student has motivation |

# Actors

* Teachers for MAP subject

# Use cases – diagram



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

Actors: Teacher

Description: Add a student

Precondition: valid student information (id, name, group)

Postcondition: The student is added to the repository

| User action | System response |
| --- | --- |
| 1 Completes the necessary fields for adding |  |
|  | 2 Checks if everything is all right. Adds the student if it is ok or displays an error message otherwise. Then it returns to menu |

Exceptions: When the fields aren’t filled, id already exists, one of the fields has an invalid data type.

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

Actors: Teacher

Description: Edit a student

Precondition: User gives the id and the fields he wants to modify and they are valid

Postcondition: If student with that id exists, then its data is updated

| User action | System response |
| --- | --- |
| 1 Completes the necessary fields for updating |  |
|  | 2 Checks if student exists. If so, it updates it, otherwise it displays an error. Then it returns to menu |

Exceptions: Student with that id doesn’t exist, one of the fields has an invalid data type, one of the fields has not been filled.

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

Actors: Teacher

Description: Delete a student

Precondition: User gives the id and it is valid

Postcondition: If student with that id exists, then it is removed from the students list

| User action | System response |
| --- | --- |
| 1 Inputs the id |  |
|  | 2 Checks if student exists. If so, it deletes it, otherwise it displays an error. Then it returns to menu |

Exceptions: Student with that id doesn’t exist or the id field is not filled.

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

Actors: Teacher

Description: Print all students

Precondition: -

Postcondition: -

| User action | System response |
| --- | --- |
| 1 |  |
|  | 2 Prints all students. Then it returns to menu |

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

Actors: Teacher

Description: Print all laboratory assignments

Precondition: -

Postcondition: -

| User action | System response |
| --- | --- |
| 1 |  |
|  | 2 Prints all assignments, Then it returns to menu |

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

Actors: Teacher

Description: Assign a lab theme to a student.

Precondition: User gives theme and student and both are valid

Postcondition: Theme is assigned to student

| User action | System response |
| --- | --- |
| 1 Completes info about student and theme |  |
|  | 2 If the user and the given theme exist, it assigns the theme to the student. Otherwise, it displays an error. Then it returns to menu |

Exceptions: When student or assignment doesn’t exist or the fields are not filled.

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

Actors: Teacher

Description: Add a lab theme

Precondition: User gives information about theme

Postcondition: Theme is added

| User action | System response |
| --- | --- |
| 1 Completes the necessary fields for adding |  |
|  | 2 Checks if everything is all right. Adds the theme if it is ok or displays an error message otherwise. Then it returns to menu |

Exceptions: When the fields aren’t filled, id already exists, one of the fields has an invalid data type.

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

Actors: Teacher

Description: Grade a student’s assignment

Precondition: User gives student, assignment and grade

Postcondition: Grade is added for the given student on the given theme

| User action | System response |
| --- | --- |
| 1 Completes the necessary fields |  |
|  | 2 Checks if everything is correct and adds the grade for the given student on the given theme. Then returns to menu |

Exceptions: When the fields aren’t filled, the assignment is invalid, the student already has a grade, the student is invalid.

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

Actors: Teacher

Description: Filter the students, assignments, themes and grades based on different criteria

Precondition: -

Postcondition: -

| User action | System response |
| --- | --- |
| 1 Completes the necessary fields |  |
|  | 2 Returns the result set of the selected filter. Then returns to menu |

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

Actors: Teacher

Description: Generate reports based on student grades, laboratories, assignments

Precondition: -

Postcondition: -

| User action | System response |
| --- | --- |
| 1 Completes the necessary fields |  |
|  | 2 Shows the corresponding report |

# Analysis

## Entities

* Student
* Laboratory assignments
* Grades

## Relations between entities

A student can have more assignments and an assignment can be assigned to more students.

A grade can be given for a specific student on an assignment.

## Attributes

* Student
  + id: String
  + name: String
  + group: Int
  + email: String
  + teacher: String
* Grade
  + id: Map<String, Int>
  + st: Student
  + assign: Assignment
  + value: Float
  + date: Int
* Assignment
  + id: Int
  + description: String
  + deadline: Int
  + delivery\_week: Int

## System behavior

## Use case 1-2-3

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

# Design

* 1. **Class diagram**

**A screenshot of a cell phone

Description automatically generated**