Laboratory Assignment AND Assessment Requirements Specification

Version 1.0

March 2, 2024

Developed by:

Râpeanu George, Pricop Laurențiu

936

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description of Change** | **Author** | **Date** |
| V01 | Initial/Modification of document |  | 1 March 2020 |
| V02 | Completion of document | Râpeanu George, Pricop Laurențiu | 8 March 2020 |

Contents

[Laboratory Assignment AND Assessment Requirements Specification 1](#_Toc1859878)

[Version 1.0 1](#_Toc1859879)

[15 March, 2019 1](#_Toc1859880)

[1 Functional Requirements 3](#_Toc1859881)

[2 Actors 3](#_Toc1859882)

[3 Use cases – diagram 3](#_Toc1859883)

[3.1 Use case number 1 (Description of the use case) 3](#_Toc1859884)

[4 Analysis 3](#_Toc1859885)

[4.1 Entities 3](#_Toc1859886)

[4.2 Relations between entities 3](#_Toc1859887)

[4.3 Attributes 3](#_Toc1859888)

[4.4 System behavior 4](#_Toc1859889)

[4.4.1 Use case 1-2-3 4](#_Toc1859890)

[4.5 System events 4](#_Toc1859891)

[5 Design 4](#_Toc1859892)

**Analysis and design Document**

# Functional Requirements

List the functional requirements (FR) of the system.

| Section/ Requirement ID | Requirement Definition |
| --- | --- |
| FR1.0 | CRUD operations for the Student entity |
| FR2.0 | Manage laboratory homework |
| FR2.1 | Extend the deadline for an existing homework |
| FR2.2 | Add a new laboratory homework |
| FR2.3 | Notify students by email when adding a new laboratory homework or modifying the delivery date of an existing homework |
| FR2.4 | Add a grade to a particular laboratory homework to a particular student. Delays will be calculated automatically |
| FR3.0 | Filter students based on different criteria |
| FR4.0 | Generate reports   * Laboratory grade for each student * The hardest homework * Students who can enter the exam * Students who have delivered all their assignments on time |

# Actors

* Teachers for MAP subject

# Use cases – diagram

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

Actors: Teacher

Description: Add a student

Precondition: User gives data for a student

Postcondition: If student is valid, then it is added to student list.

|  |  |
| --- | --- |
| 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

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.

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

Actors: Teacher

Description: Delete a student

Precondition: User gives the id

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.

## 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

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.

## 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.

## 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 Select a report type to be generated:   1. Laboratory grade for each student 2. The hardest assignment 3. Students who can enter the exam 4. Students who have delivered all assignments on time |  |
|  | 2 Show the corresponding report:   1. The weighted average of grades from the lab topics; weight share = number of weeks allocated to the topic 2. The homework where the average of all grades is smallest. 3. Students with average greater than 4. 4. Students with no delays |

# 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 faculty’s workflow.

# Design

* 1. **Class diagram**

