Laboratory Assignment AND Assessment Requirements Specification

Version 1.0

8 March 8, 2020

Developed by:

Student M, Student N

931

Version History

Version	Description of Change	Author	Date
V01	Initial/Modification of document	Student N	1 March 2020
V02	Completion of document	Student M	8 March 2020

Contents

Analysis and design Document

1 Functional Requirements

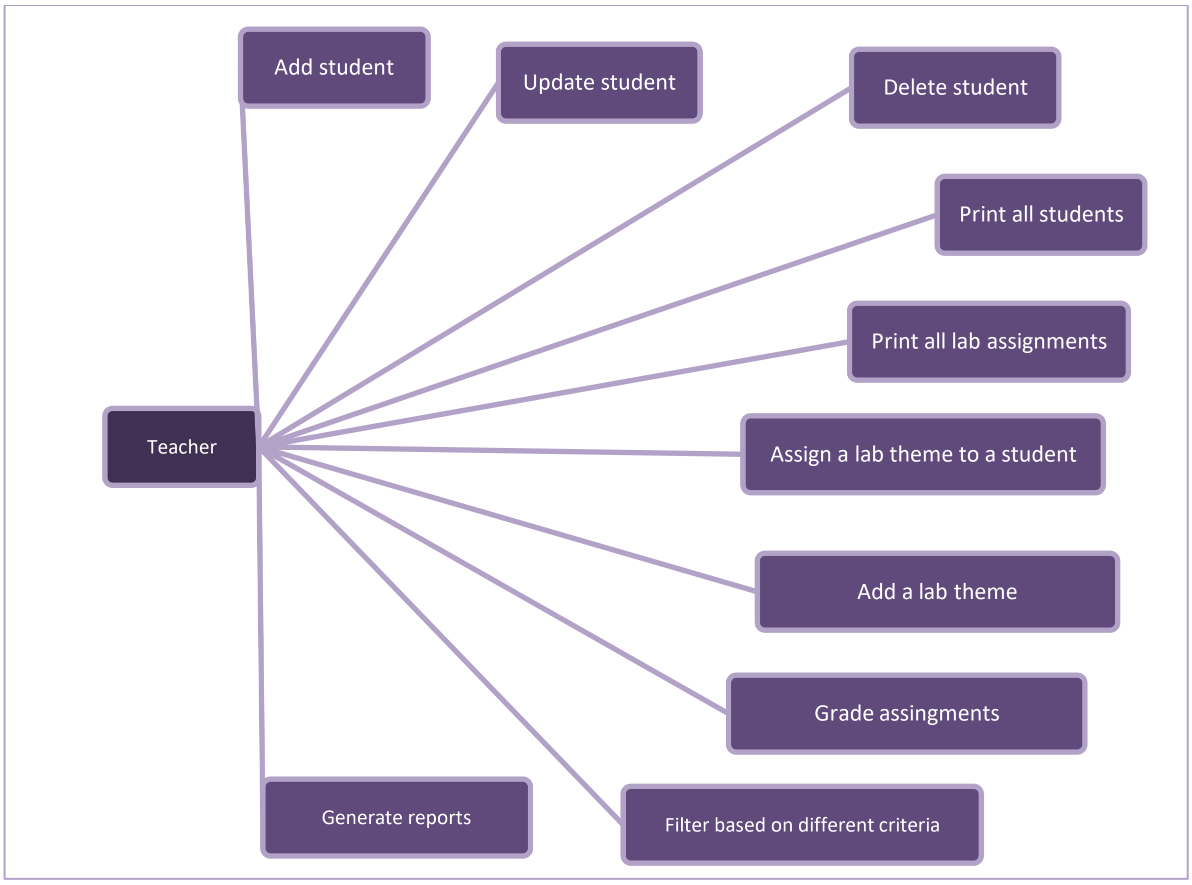
List the functional requirements (FR) of the system.

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
FR3.0	Filter students based on different criteria
FR4.0	Generate reports

2 Actors

- Teachers for MAP subject
- #The system should be designed for any kind of teacher from the requirements

3 Use cases – diagram



3.1 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.

3.2 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.

3.3 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.

3.4 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

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

Actors: Teacher

Description: Print all laboratory assignments

Precondition: -

Postcondition: -

User action	System response
1 #Request laboratory assignments	
	2 Prints all assignments, Then it returns to menu

3.6 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.

3.7 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.

3.8 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.

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

Actors: Teacher

Description: Filter the students, assignments, themes and grades based on different criteria
# Precondition: Some valid criterias for the etntities are provided and are also valid for the entities
Postcondition: -

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

#Exception: Criteria fields are not provided or are invalid for a certain entity

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

Actors: Teacher
Description: Generate reports based on student grades, laboratories, assignments
Precondition: Fields for the report are completed
Postcondition: -

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

#Exception : Fields for the report are completed in a wrong way

4 Analysis

4.1 Entities

- Student
- Laboratory assignments
- Grades
- #Themes

4.2 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.

4.3 Attributes

- Student
  - id: String
  - name: String
  - group: Int
  - email: String
  - teacher: String
- Grade

  - id: Map<String, Int> # ambiguous, should provide some explanation
  - st: Student #ambiguous
  - assign: Assignment #ambiguous
  - value: Float
  - date: Int #may be refactored as datetime type for safety and clarity
- Assignment
  - id: Int
  - description: String
  - deadline: Int
  - delivery_week: Int #doesn't respect Java naming conventions

4.4 System behavior

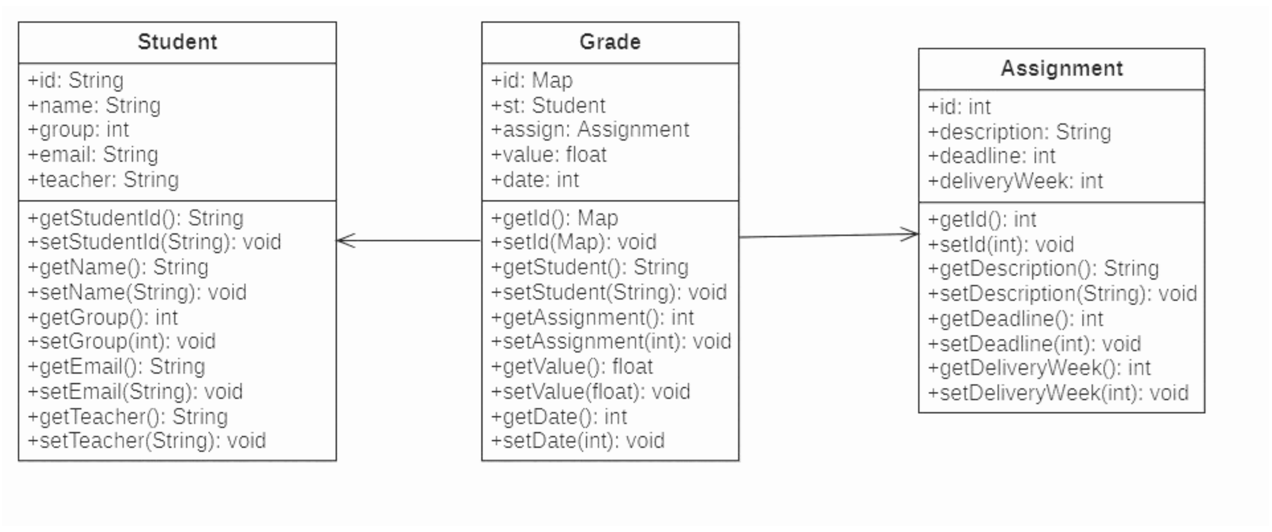
4.4.1 Use case 1-2-3

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.

#Probably word should not appear

5 Design

5.1 Class diagram


Student
+id: String
+name: String
+group: int
+email: String
+teacher: String
+getStudentId(): String
+setStudentId(String): void
+getName(): String
+setName(String): void
+getGroup(): int
+setGroup(int): void
+getEmail(): String
+setEmail(String): void
+getTeacher(): String
+setTeacher(String): void
Grade
+id: Map
+st: Student
+assign: Assignment
+value: float
+date: int
+getId(): Map
+setId(Map): void
+getStudent(): String
+setStudent(String): void
+getAssignment(): int
+setAssignment(int): void
+getValue(): float
+setValue(float): void
+getDate(): int
+setDate(int): void
Assignment
+id: int
+description: String
+deadline: int
+deliveryWeek: int
+getId(): int
+setId(int): void
+getDescription(): String
+setDescription(String): void
+getDeadline(): int
+setDeadline(int): void
+getDeliveryWeek(): int
+setDeliveryWeek(int): void