Student:Iancu Daiana

**Group:30431**

**Assignment 3**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Design and implementation of an application for tracking the laboratory activity of the Software Design laboratory. The application must have two types of users: teacher and student. For the second part, create a desktop application to consume the APIs.

# Functional Requirements

The application has the following functional requirements, divided into two categories according to each type of user:

The teacher can perform the following operations:

* Login
* CRUD on students. When you create a student, a 128 characters token is created. Using that token student should be able to register. Teacher will send the token by email manually. For each student we should track: email address, full name, group (ex. 30431) and top 1 hobby.
* Can add/edit/delete Laboratory classes. For each class we should track: laboratory number (1-14), date, title, curricula for what are the topics presented in that lab and a long description with the laboratory text (should accept html).
* CRUD on attendance for each lab.
* CRUD on assignments. Some of the laboratory will have assignments: for each assignment we must track the name, deadline and a long description with the assignment text.
* Grade the submitted assignments. It should also be possible to regrade the assignment.
* Get the list of grades for all students for a given assignment.

The student can perform the following operations:

* Register using the token generated by the teacher.
* Login with the username and password.
* View a list of laboratory classes. Also view a filtered list: student inserts a keyword and that keyword is searched in the curricula and long description.
* View the assignments for a laboratory class.
* Create an assignment submission. Here, students should be able to insert a link to a git repository and a short remark for the teacher.

For the second part of the application, the operations performed are reduced to the ones presented in the use case diagram.

# 1.3 Non-functional Requirements

* The data will be stored in a database.
* Use the MVC architectural pattern to organize the application.
* API design should be RESTful.
* Use and ORM (Hibernate / Entity framework) to access the database
* Use dependency injection to inject Services in Controllers and Repositories in Services
* Install and use Swagger to call the APIs.
* Connection string should be stored in a separate config file

2. Use-Case Model

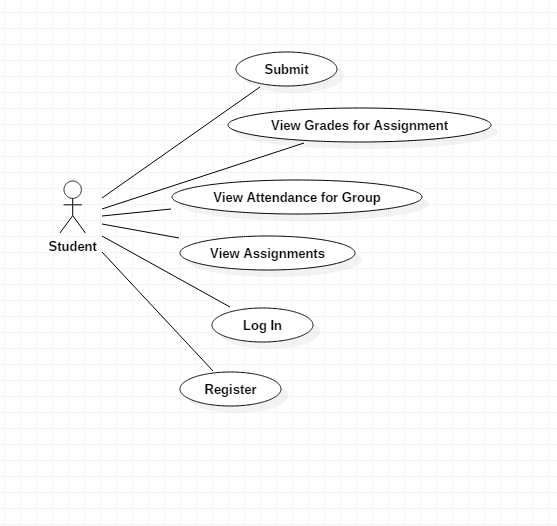
*Use case: create Submission*

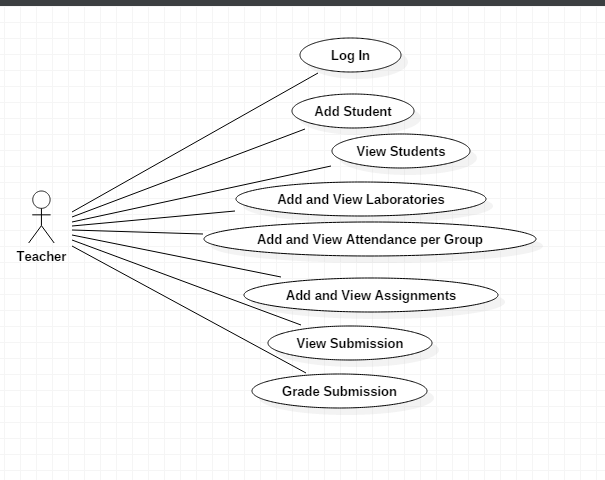
*Level: user-goal level*

*Primary actor: student*

*Main success scenario: successful submission*

*Extensions: unsuccessful submission*





3. System Architectural Design

**3.1 Architectural Pattern Description**

The used architectural pattern is Model-View-Controller.

The MVC pattern is used to divide an application into 3 parts, as the name suggests it:

* the model-containing the functionality and data
* the view-displays information to the user
* the controller-handles the input from the user

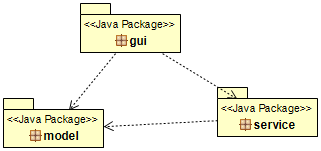
This pattern allows efficient code reuse.

In the package diagram, the split of the application in a MVC manner can be observed.

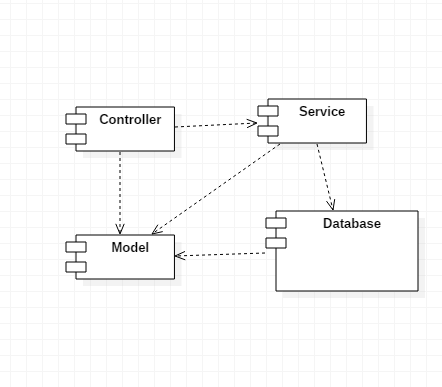
In this assignment, the view part is represented by the desktop application, which is separated as well into three parts, as it can be seen in the package diagram.

**3.2 Diagrams**

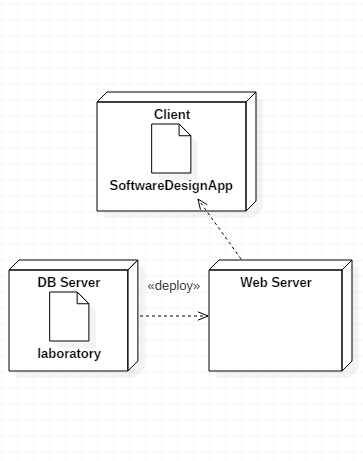
Package diagram



Component Diagram

The controller is represented by the desktop application itself.

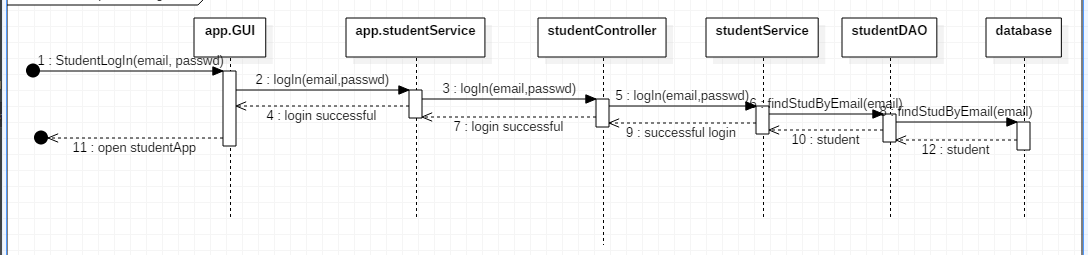
Deployment Diagram



4. UML Sequence Diagrams

The scenario for which the sequence diagram below is created is: student log in

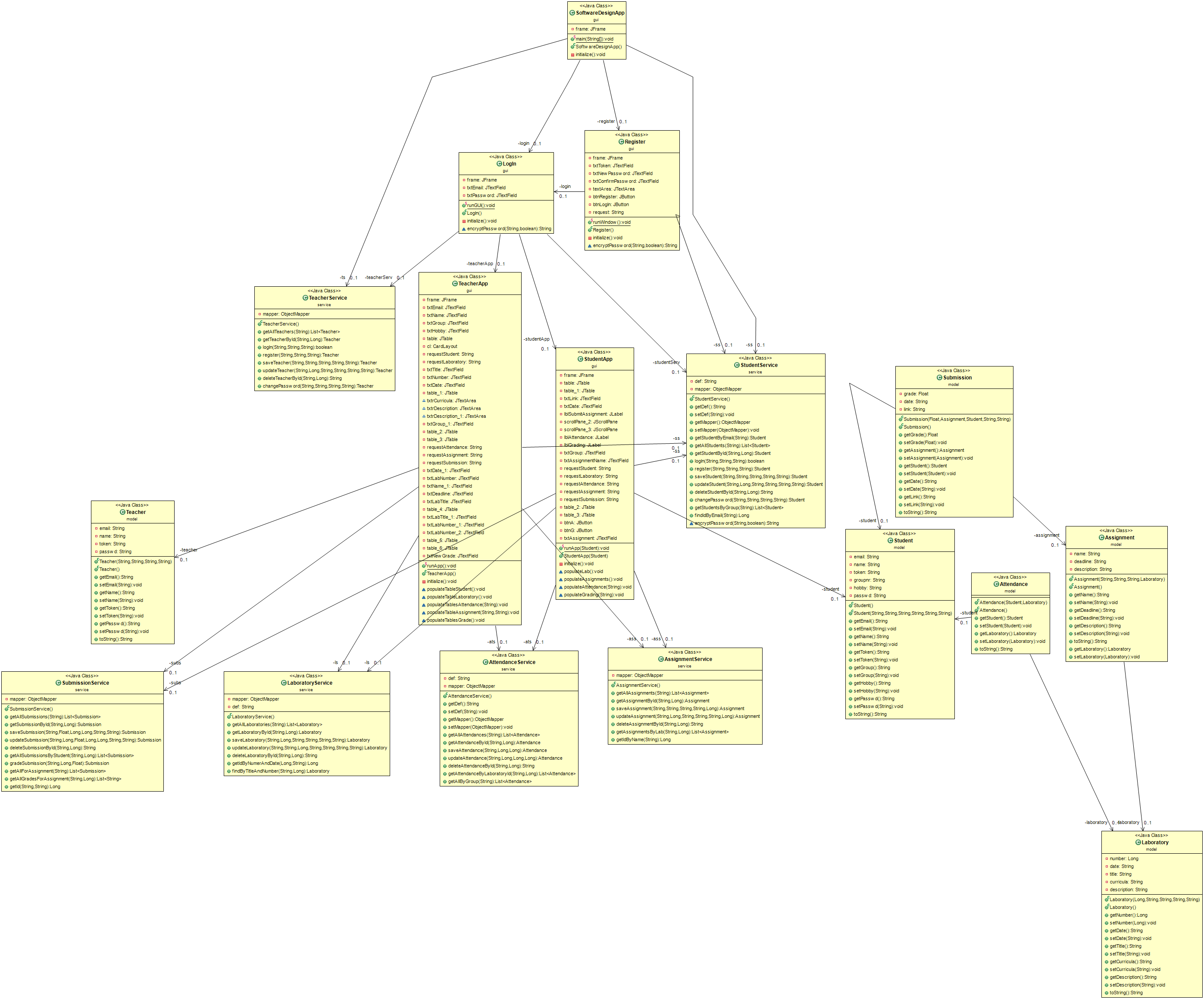
Student LogIn



5. Class Design

**5.1 Design Patterns Description**

**5.2 UML Class Diagram**

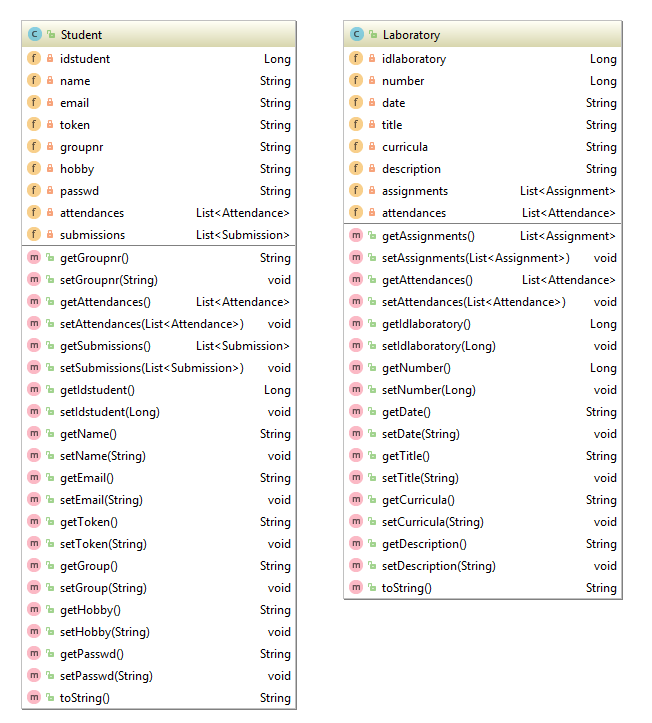
**

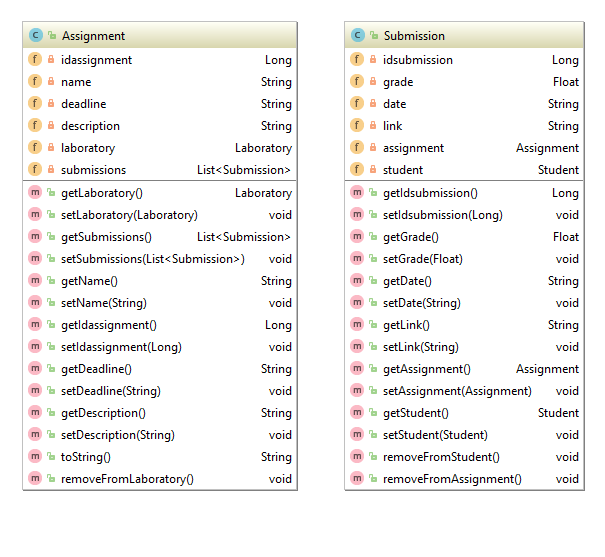
6. Data Model

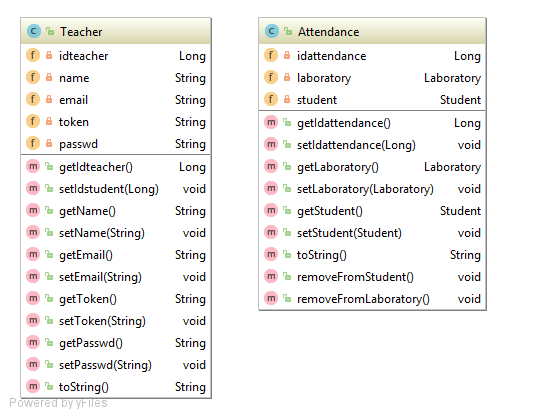
The following classes represent the model:

* Student
* Teacher
* Laboratory
* Attendance
* Assignment
* Submission

In the images below, we can observe the attributes that these classes have.







7. System Testing

The testing was done for each flow database-model-service-controller.

For example, for the StudentController, there were tests performed for each of the methods, using correct as well as incorrect examples. For instance, when trying to add a student with the same e-mail, nothing happens.

8. Bibliography