<Assignment 3>

Analysis and Design Document

Student: Marius Pop

**Group:30431**

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

# The objective of this assignment is to extend the second assignment by adding security on an API and consuming the API in a desktop application.

# Functional Requirements

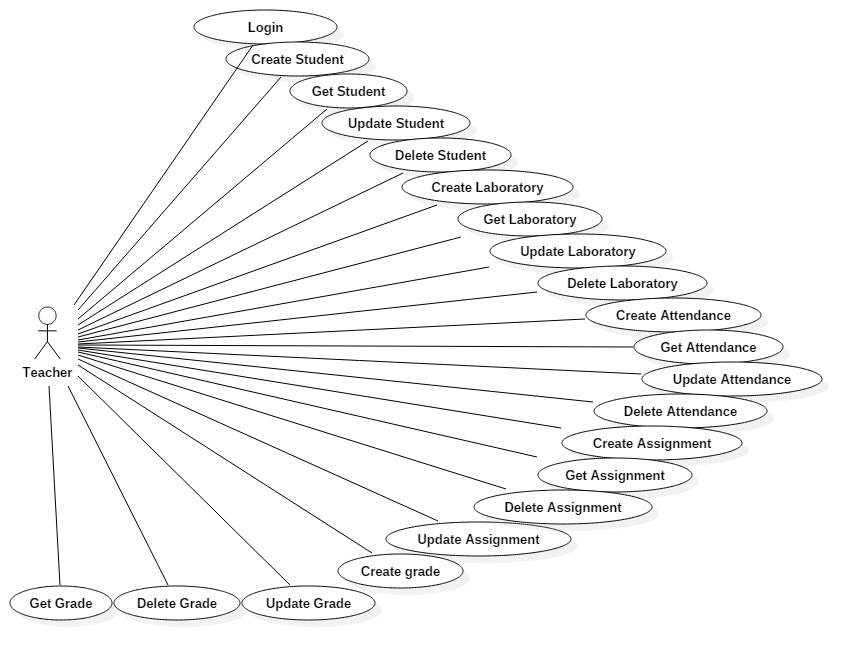
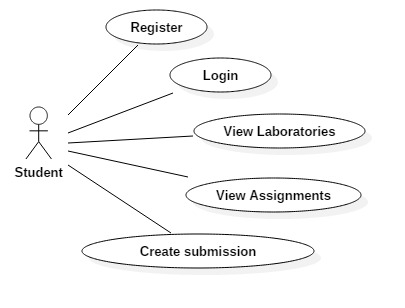
# Use JAVA Spring/C# Web API in combination with JAVA Swing/.NET WinForms to design and implement an application for tracking the laboratory activity for the Software Design laboratory. The requirements of the application are provided in Assignment 2. Additional requirements are: - Create a desktop application to consume the APIs. Have screens for login, teacher side: register students, laboratory, assignments, attendance and grading; student side: first time register, login, laboratory, assignments, post attendance and grading. Nice to have: view attendance for students. Any additional feature that you consider helpful is highly appreciated. [8 points]

# - Secure all APIs with an authorization header that contains an Email and a Password. First, you will have to check that the password is correct and then find out the role (Teacher/Student) for that given user. Do not do this logic in every controller method, this logic is common for all controllers, so it has to be at a higher level. Depending on the roles, some APIs might be accessed only by the teacher (for example: add student, add laboratory), only by the student (submit assignment) or by both (view laboratories). - Encrypt the password from the UI with a one direction algorithm and then work with the password encrypted. [1 point] Nice to have:

# - Create a system that notifies by email all students when a new assignment is posted by the teacher.

# - When a new user is added by the teacher, the token is sent by email. [extra points]

2. Use-Case Model



Use case: Login

Level: user-goal level

Actors: Teacher (admin), Student

Main success scenario: Successfully login for student

Extensions: Student failed to login

Use case: Create student

Level: user-goal level

Primary actor: Teacher (admin)

Main success scenario: Student successfully created

Extensions: Student creation failed

Use case: Retrieve a student

Level: user-goal level

Primary actor: Teacher (admin)

Main success scenario: Student retrieved by id successfully executed

Extensions: Student not found for the given id

Use case: Delete a student

Level: user-goal level

Primary actor: Teacher (admin)

Main success scenario: Student successfully deleted from the database

Extensions: Unable to find a student by id

Use case: CRUD on laboratory

Level: user-goal lever

Primary actor: Teacher

Main success scenario: Successfully executed operations on laboratory

Extensions: Unsuccessfully executed operations on laboratory

Use case: CRUD on attendance

Level: user-goal lever

Primary actor: Teacher

Main success scenario: Successfully executed operations on attendance

Extensions: Unsuccessfully executed operations on laboratory

Use case: CRUD on assignment

Level: user-goal lever

Primary actor: Teacher

Main success scenario: Successfully executed operations on assignment

Extensions: Unsuccessfully executed operations on laboratory

Use case: CRUD on grades

Level: user-goal lever

Primary actor: Teacher

Main success scenario: Successfully executed operations on laboratory

Extensions: Unsuccessfully executed operations on laboratory

Use case: Register

Level: user-goal lever

Primary actor: Student

Main success scenario: successfully register via email and token

Extensions: failure to register

Use case: View laboratories

Level: user-goal lever

Primary actor: Student

Main success scenario: Successfully displayed all the labs

Extensions: No labs displayed

Use case: View assignments

Level: user-goal lever

Primary actor: Student

Main success scenario: Successfully displayed the assignments

Extensions: No assignment displayed

Use case: Create submission

Level: user-goal lever

Primary actor: Student

Main success scenario: submission successfully created

Extensions: unable to create submission

3. System Architectural Design

**3.1 Architectural Pattern Description**

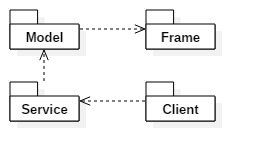
The architectural pattern used was MVC – Model, View, Controller.

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

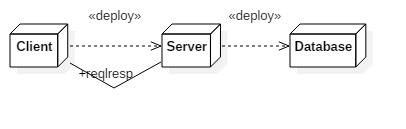
* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

**3.2 Diagrams**

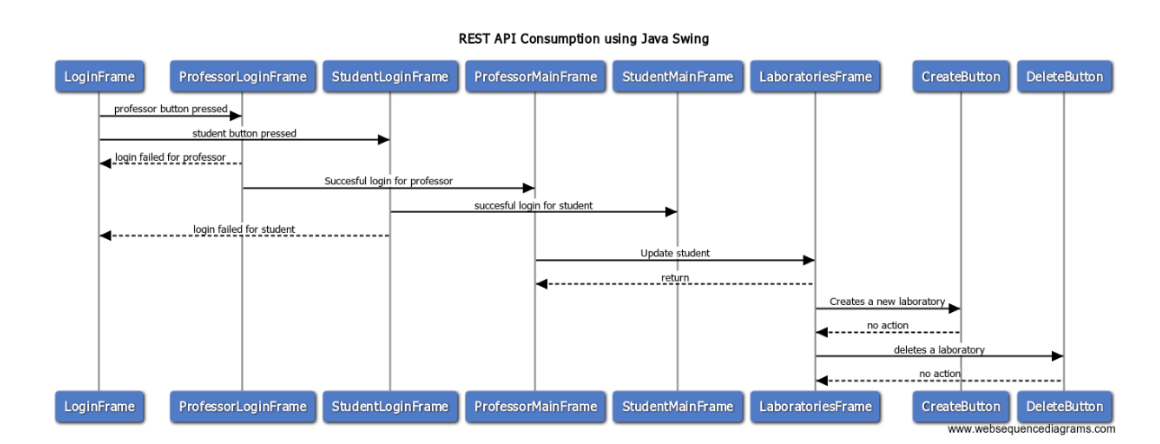
Package diagram



Deployment diagram



4. UML Sequence Diagrams



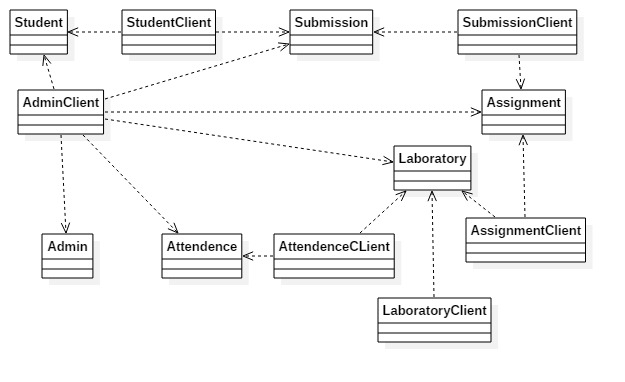
5. Class Design

**5.1 Design Patterns Description**

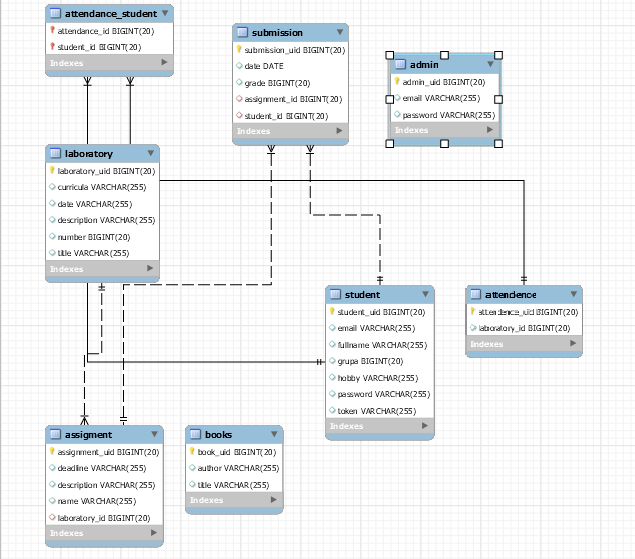
* ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C#, etc.
* An ORM system has the following advantages over plain JDBC −

|  |  |
| --- | --- |
| **Sr.No.** | **Advantages** |
| 1 | Let’s business code access objects rather than DB tables. |
| 2 | Hides details of SQL queries from OO logic. |
| 3 | Based on JDBC 'under the hood.' |
| 4 | No need to deal with the database implementation. |
| 5 | Entities based on business concepts rather than database structure. |
| 6 | Transaction management and automatic key generation. |
| 7 | Fast development of application. |

**5.2 UML Class Diagram**



6. Data Model



7. System Testing

The system testing has been performed step by step in the process of methods creation(each method has been test more than 3 times in order to maximum accuracy).

8. Bibliography

* <https://tutorialspoint.com/>
* <http://tutorialspedia.org/>
* <https://swagger.io/>