Student: Adela Mera

**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 4

3. System Architectural Design 4

4. UML Sequence Diagrams 6

5. Class Design 7

6. Bibliography 8

1. Requirements Analysis

# Assignment Specification

# Design and implement an application for the tracking the laboratory activity for the Software Design laboratory. The application should have two types of users (student and teacher) which must provide an email and a password to use the application. Create a desktop application to consume the APIs, which will have screens for login, teacher side: create students, laboratory, assignments, attendance and grading; and student side: first time register, login, laboratory, assignments, post attendance and grading.

# Functional Requirements

* CRUD on student
* CRUD on attendance
* CRUD on assignments
* CRUD on laboratory
* submit assignment
* grade assignment
* login
* register
* see the grades for a given assignment
* see a filtered list of laboratories based on a keyword

# Non-functional Requirements

## Performance

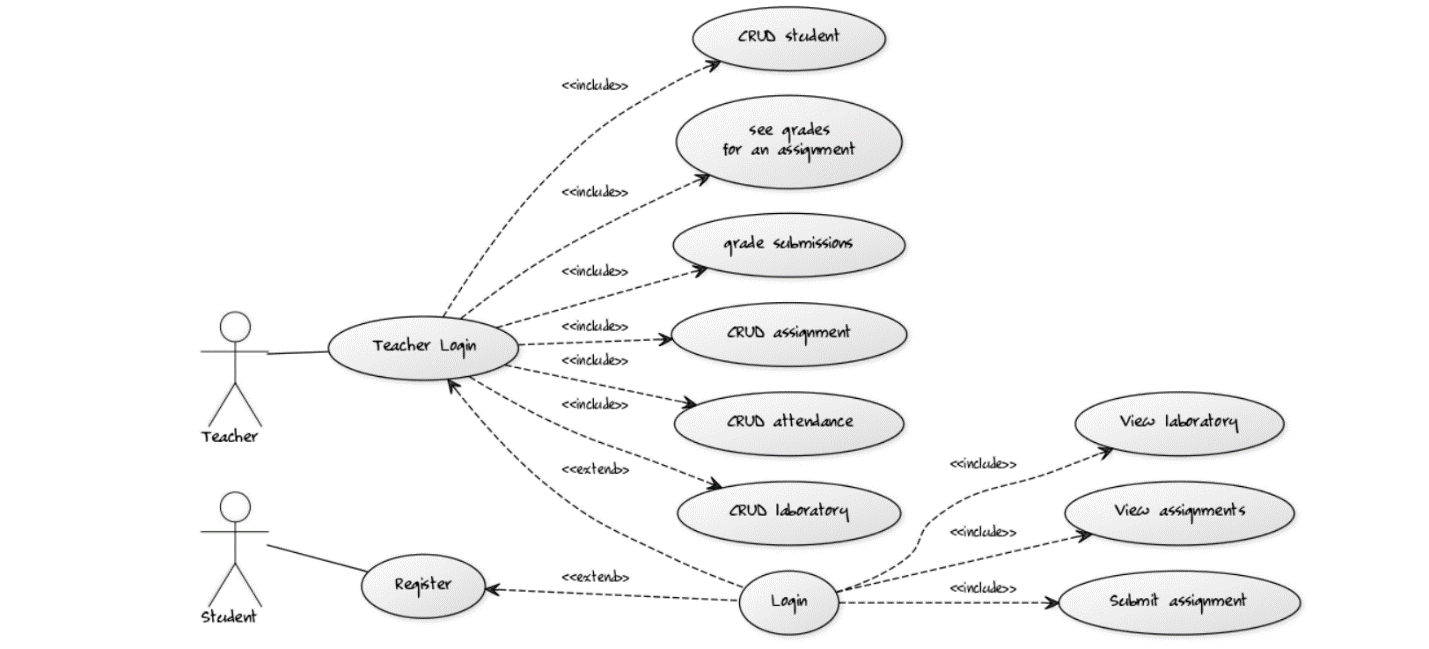
* The operations should take the least time possible to complete
* Minimal time between click and system response

**Availability**

* The system shall meet or exceed 99.99% uptime.

## Security

* Password will be encrypted
* Some operations are only available to a certain type of user (e.g. teacher)

2. Use-Case Model

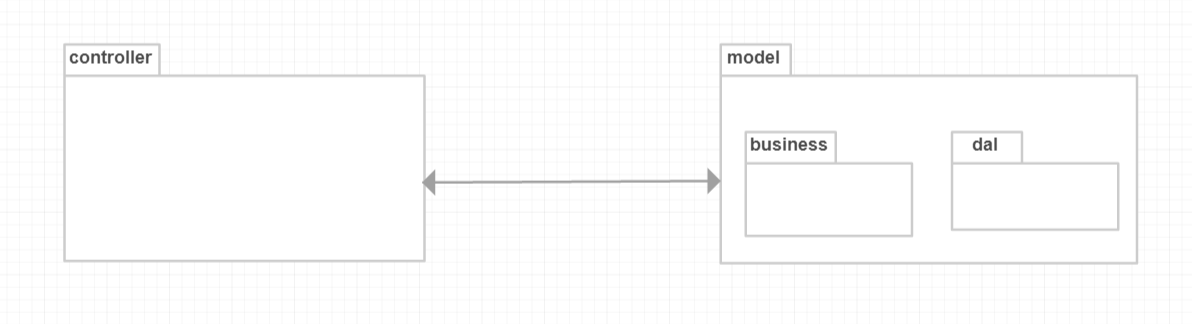
3. System Architectural Design

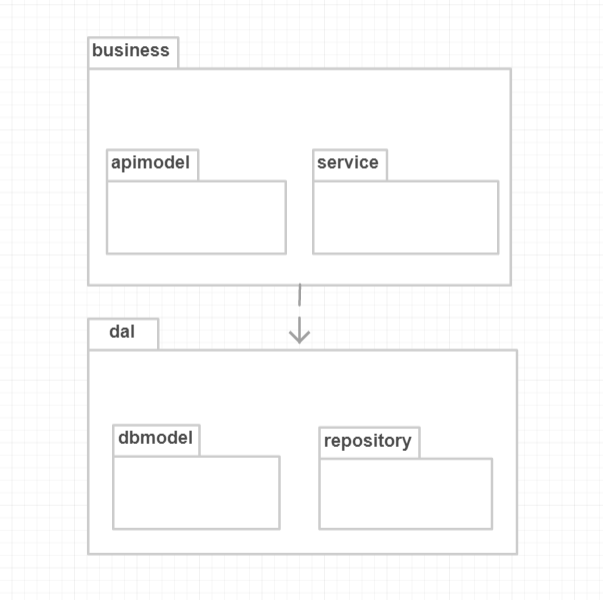
**3.1 Architectural Pattern Description**

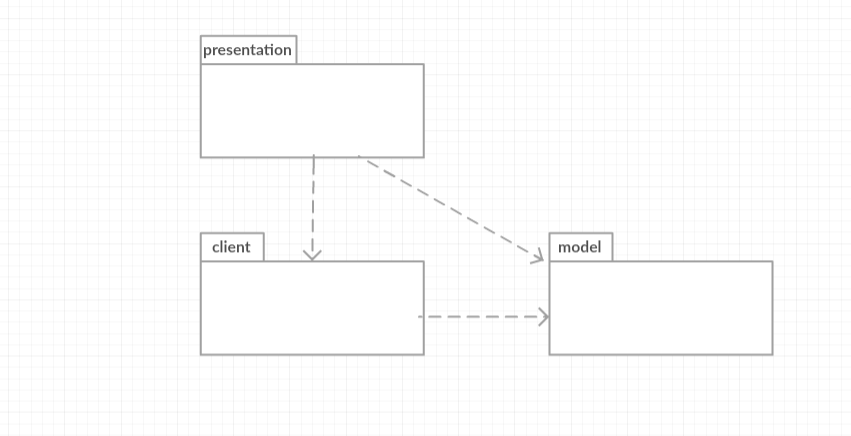
MVC architectural pattern was used in order to isolate the application logic from the user interface layer and to support separation of concerns. This pattern separates an application into three main components: the model, the view, and the controller.

* **Model** − is responsible for managing the data of the application, responds to the request from the view and it also responds to instructions from the controller to update itself.
* **View** – is responsible for displaying all or a portion of the data to the user.
* **Controller** − is responsible for responding to the user input, performing interactions on the data model objects; it receives the input and then performs the business operation that modifies the state of the data model

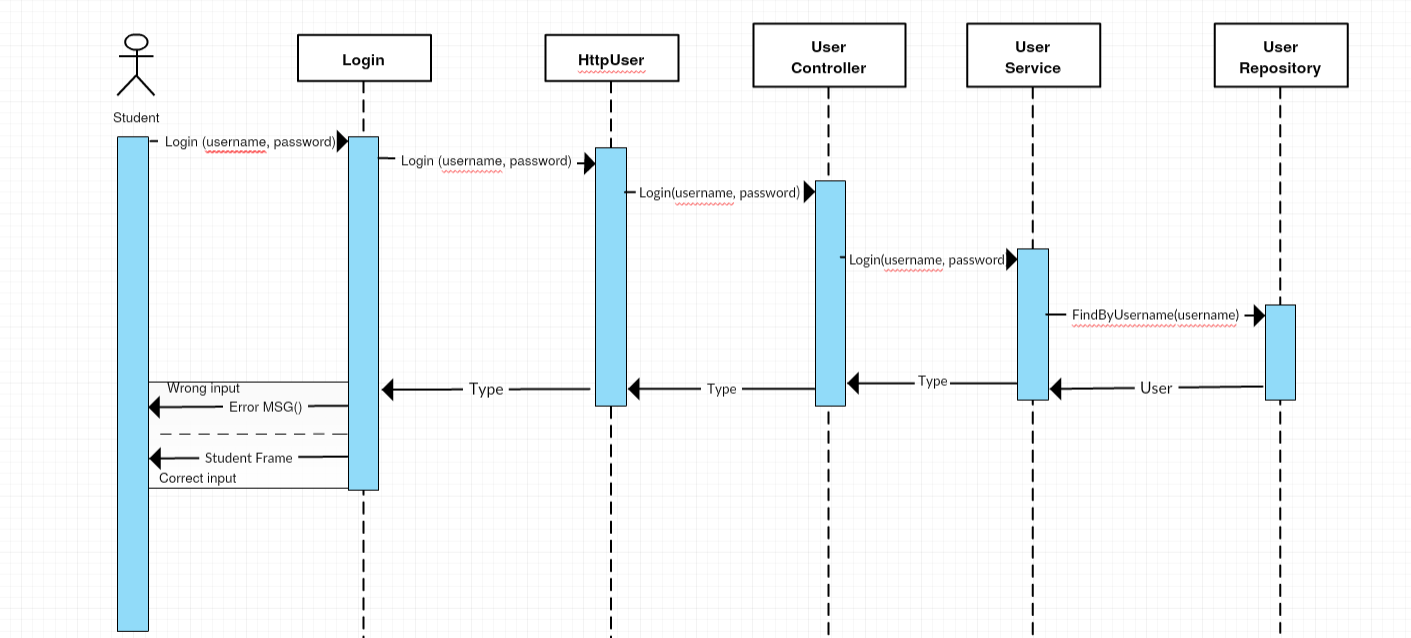
**3.2 Diagrams**





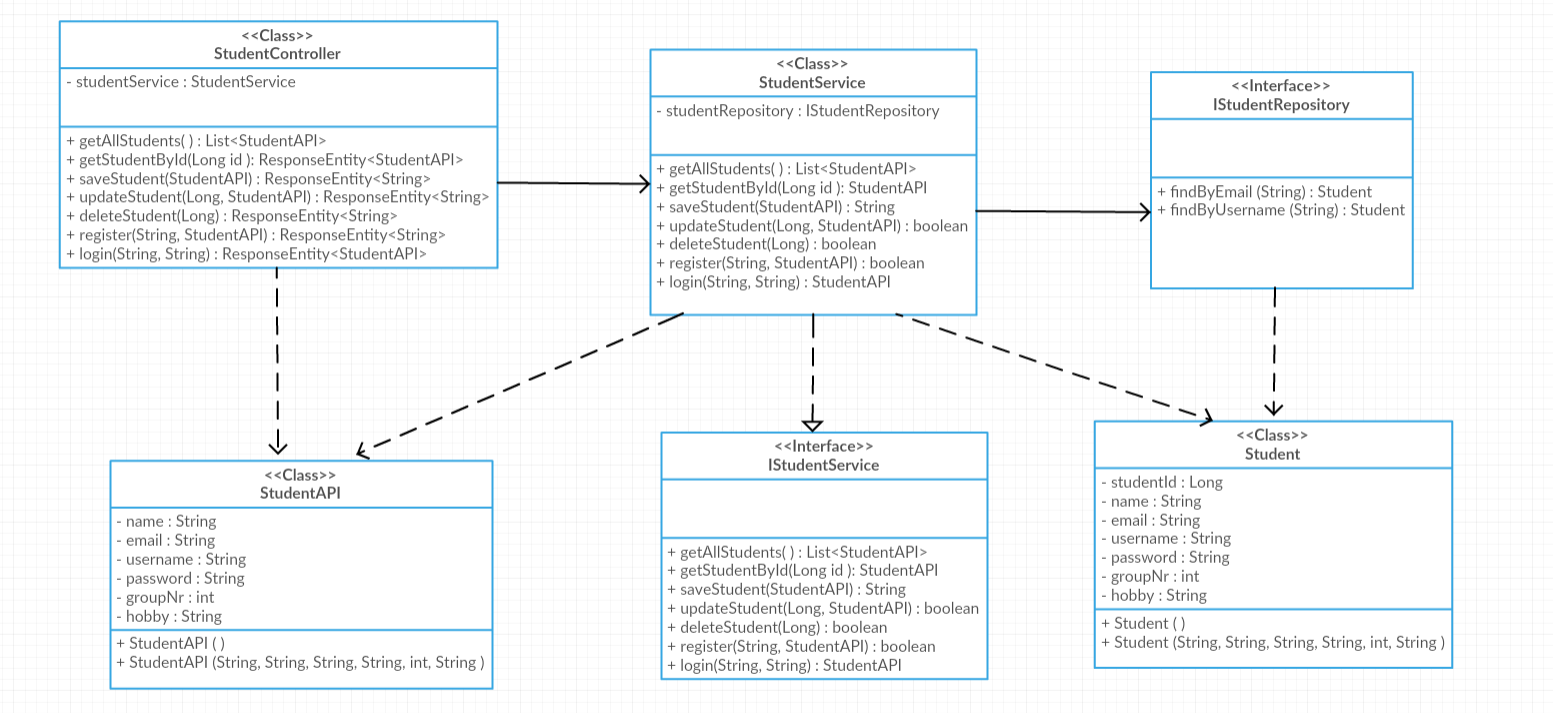


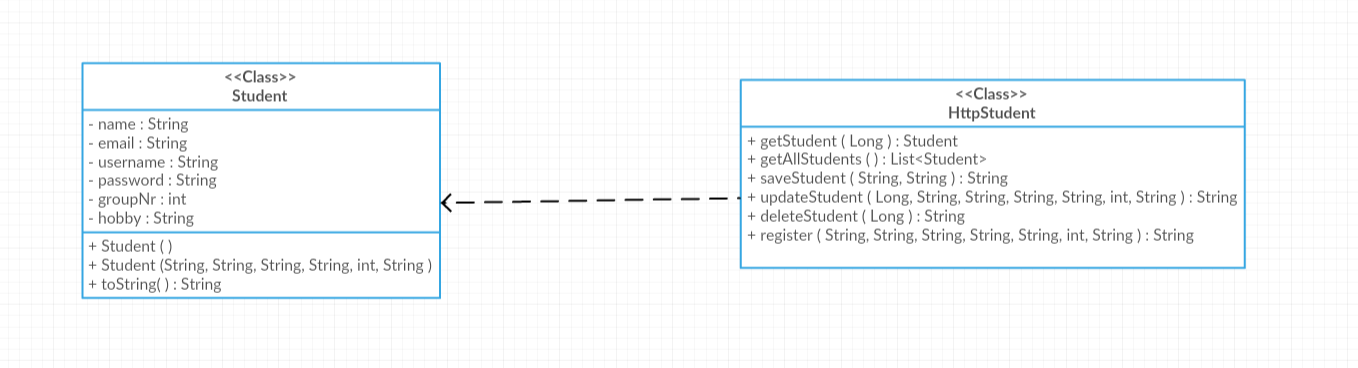
4. UML Sequence Diagrams



5. Class Design

**5.1 UML Class Diagram**





6. Bibliography

<http://www.baeldung.com/httpclient-post-http-request>

<https://www.mkyong.com/java/apache-httpclient-examples/>

<http://www.vogella.com/tutorials/ApacheHttpClient/article.html>

<https://www.mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/>