ASSIGNMENT A1

**Students Management Application**

Student: Biris Alexandra

**Group: 30432**

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

Java application for the management of students in the CS Department at Technical University of Cluj-Napoca.

# Functional Requirements

The application should have two types of users (student and teacher/administrator user) which have to provide a username and a password in order to use the application.

The regular user can perform the following operations:

- Add/update/view client information (name, identity card number, personal numerical code, address, etc.).

- Create/update/delete/view student profile (account information: identification number, group, enrolments, grades).

- Process class enrolment (enroll, exams, grades).

The administrator user can perform the following operations:

- CRUD on students information.

- Generate reports for a particular period containing the activities performed by a student.

# Non-functional Requirements

1. Availability: the system needs to be available 90% for the user with less activity during summer time
2. Accuracy: the system should accurately provide real time information
3. Performance: the system should respond to the user in less than several seconds from the time of the request submittal
4. Security: all system data must be backed up every 24 hours and this can be achieved by developing a second database
5. Usability: the system will have a GUI, it will be user friendly and it will not require a special training

2. Use-Case Model

*Use case:* View grades

*Level:* User-goal level

*Primary actor:* Student

*Main success scenario:*

-the student logs in her/his account using a username and a password

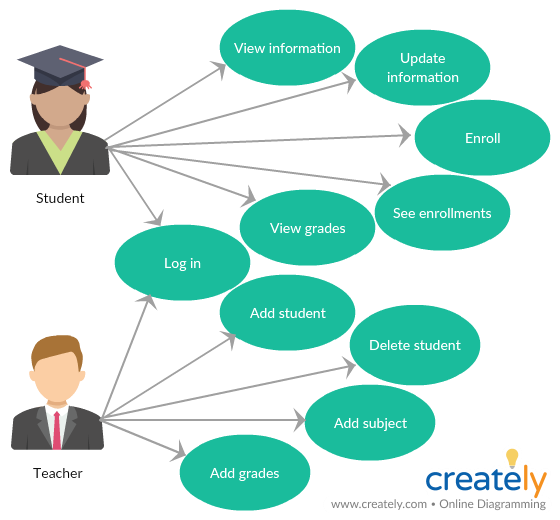
-the view grades button is pressed

-a list with all the exam grades is displayed on the screen

*Extensions:*

-problems at the log in, such as student forgetting the username or the password

-error encountered when accessing the database

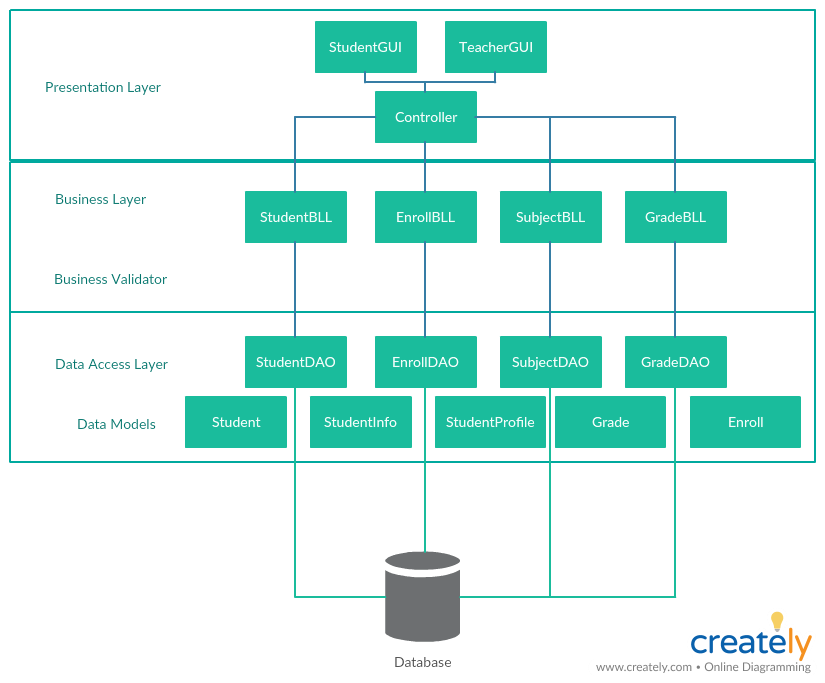


3. System Architectural Design

**3.1 Architectural Pattern Description**

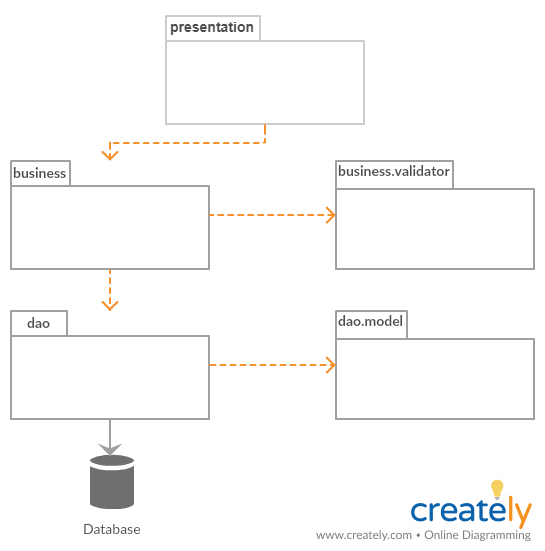
The architectural pattern used is Layers. Components within this pattern are organized into horizontal layers, each layer performing a specific role within the application. Although it does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of three standard layers: presentation, business and database.

1. Presentation layer: responsible for handling all user interface and browser communication logic
2. Business layer: responsible for executing specific business rules associated with the request
3. Database layer: responsible for executing SQL statements to retrieve the corresponding data and pass it back up in the business layer.

****

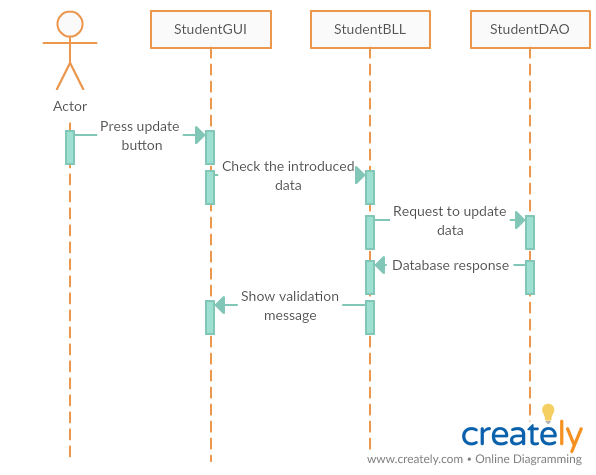
**3.2 Diagrams**

Package diagram:



4. UML Sequence Diagrams

The sequence diagram for updating student information:

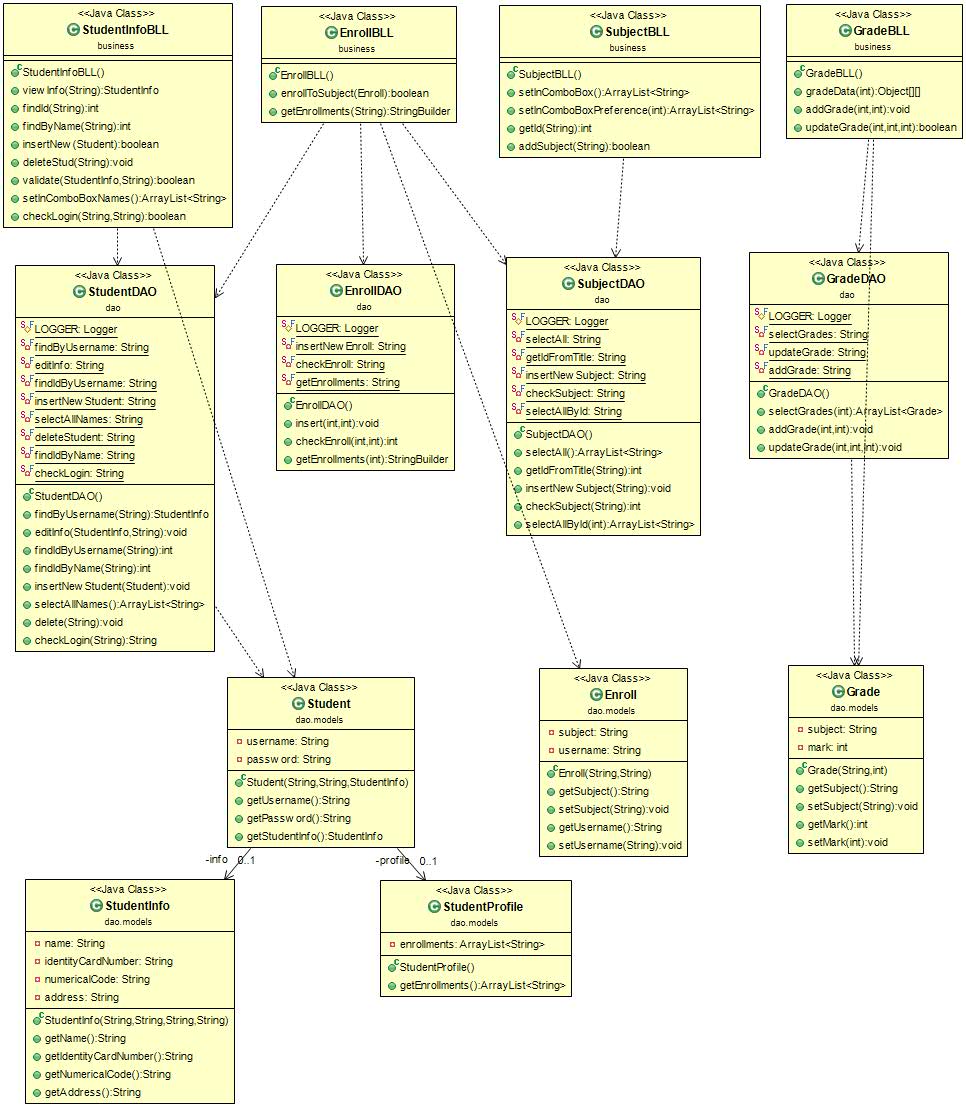
**

5. Class Design

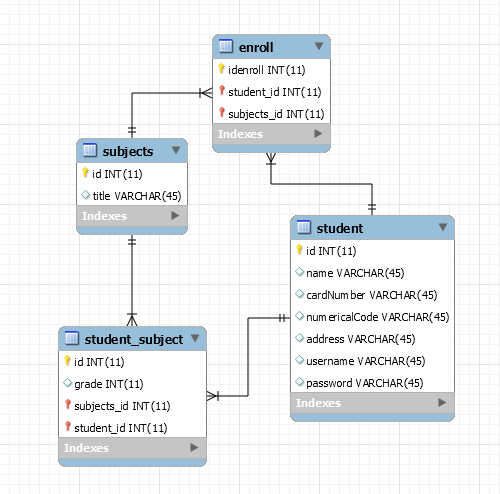
**5.1 Design Patterns Description**

Singleton pattern involves a single class which is responsible to create an object while making sure that only single object gets created. I used this pattern in the creation of the connection to the database, to ensure the fact that a single connection is established and to maintain it.

**5.2 UML Class Diagram**

**

6. Data Model

**

7. System Testing

Mockito is a popular mock framework which can be used in conjunction with JUnit. Mockito allows us to create and configure mock objects, by simplifying the development of tests for classes with external dependencies significantly.

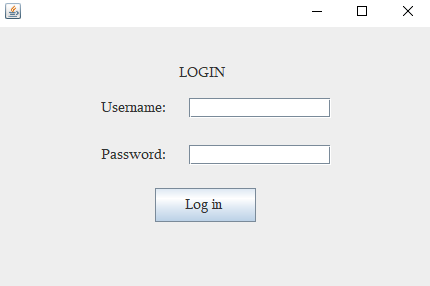
A mock object is a dummy implementation for an interface or a class in which we define the output of certain method calls. Mock objects are configured to perform a certain behavior during a test.

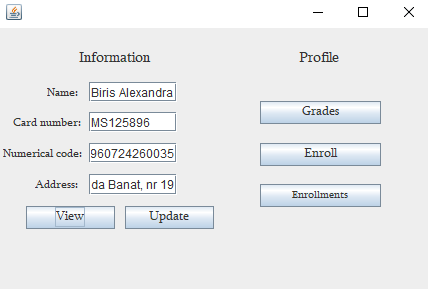
8. Bibliography

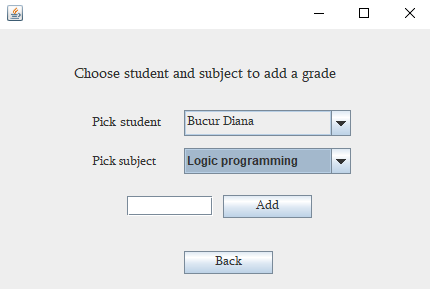
<https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

<https://msdn.microsoft.com/en-us/library/ee658109.aspx>

**Screenshots of the application:**

****

****

****