Assignment 2
Analysis and Design Document

Student:David Mihai Stan
Group:30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implement an application for the “Course enrollment” of the students from the Technical University of Cluj-Napoca.

1.2 Functional Requirements

The application has 2 types of users : student and administrator.
The user student(user) can perform the following:
-> View his personal informations(name, address, unique identification number)
->Update his personal information
->Enroll into classes

The administrator can perform the following :
-> Create/Update/Read/Delete student's information

1.3 Non-functional Requirements

-> The application should be developed in a Java MV* development platform.
->The application should be structured using the Layers and MVC architectural patterns
-> Build automation and dependencies should be handled by a specialized tool such as Maven.
->Integration test or dependency injection should be used

2. Use-Case Model

Use case: Change name
Level: user-goal level
Primary actor: student
Main Success scenario: The student clicks on the edit button-> edits the name-> clicks the save button
Extensions: if the name field is empty, the name cannot be saved

3. System Architectural Design

3.1 Architectural Pattern Description

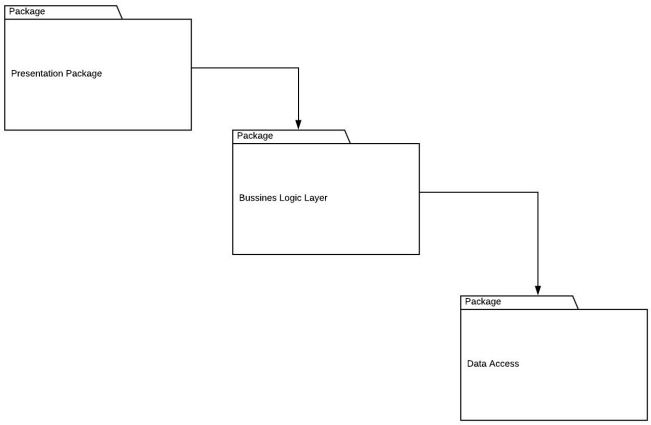
The main architectural design concept used is the layers concept. Layers are logical groupies (software components) that an architectural design can be decomposed into. Layers help to differentiate between the different kind of tasks performed by the components, making it easier to create a design that supports the reusability of the components.

Beside layers, MVC design pattern will also be used. MVC helps by having a better separation between the UI components, logic and data model components.

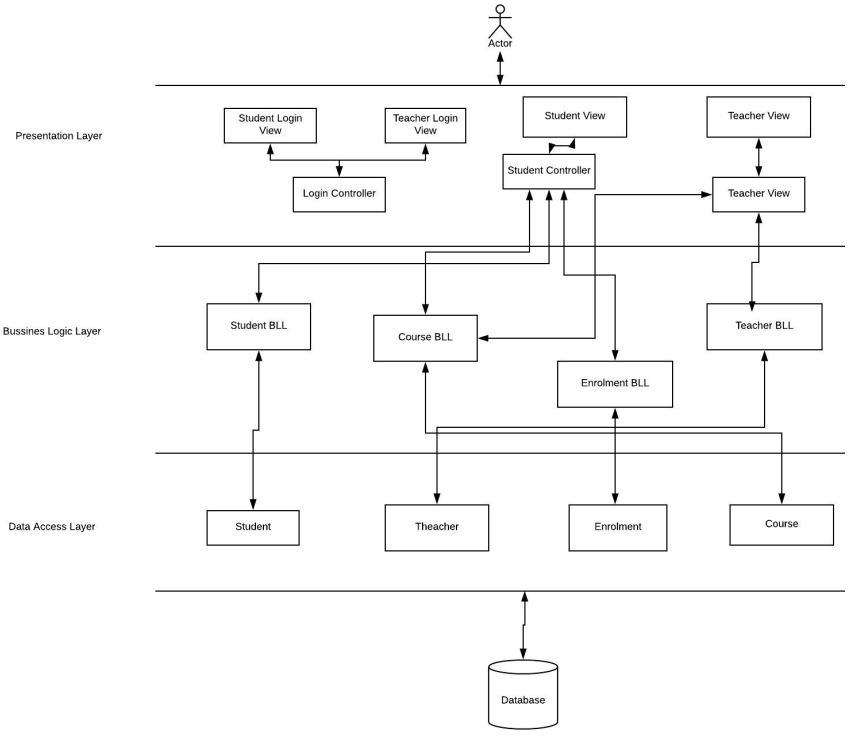
M comes from model, which represents the mapping of the data that will be used.
V stands from View, and refers to the UI components that the user will interact with.
C represents the Controller, and stands for the totality of operations done to the data model, in order to be suitable for the design, or the manipulation of the data model.

3.2 Diagrams

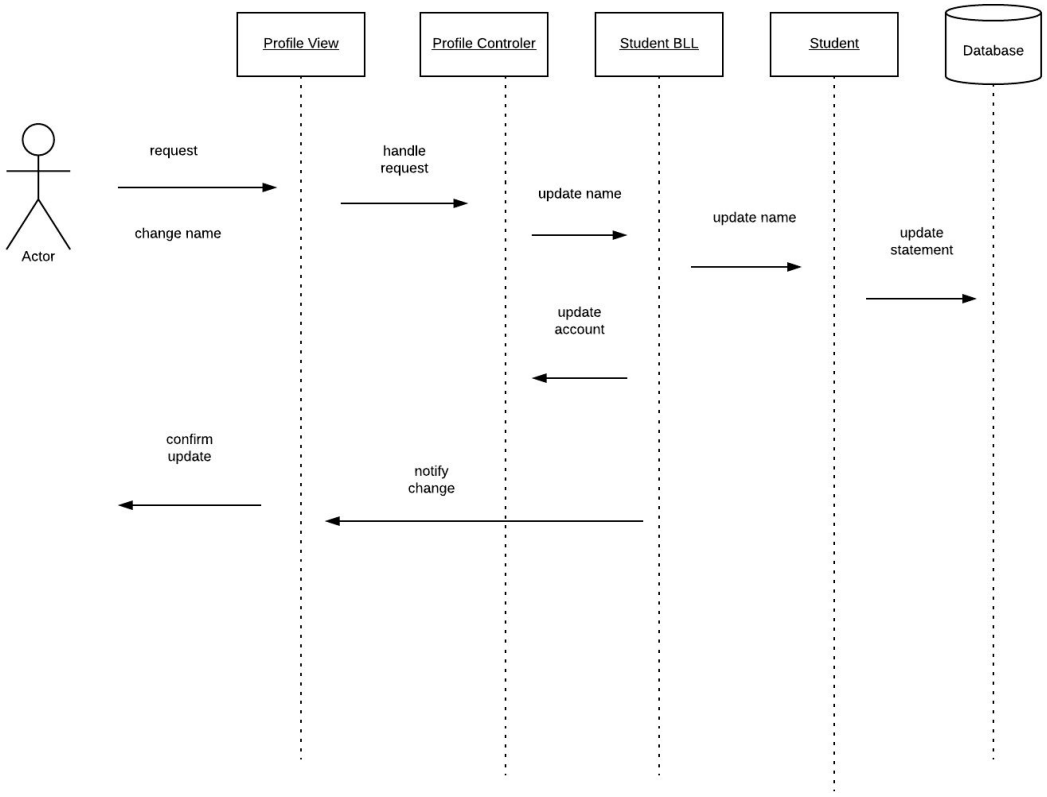
Package Diagram



Layers



4. UML Sequence Diagrams



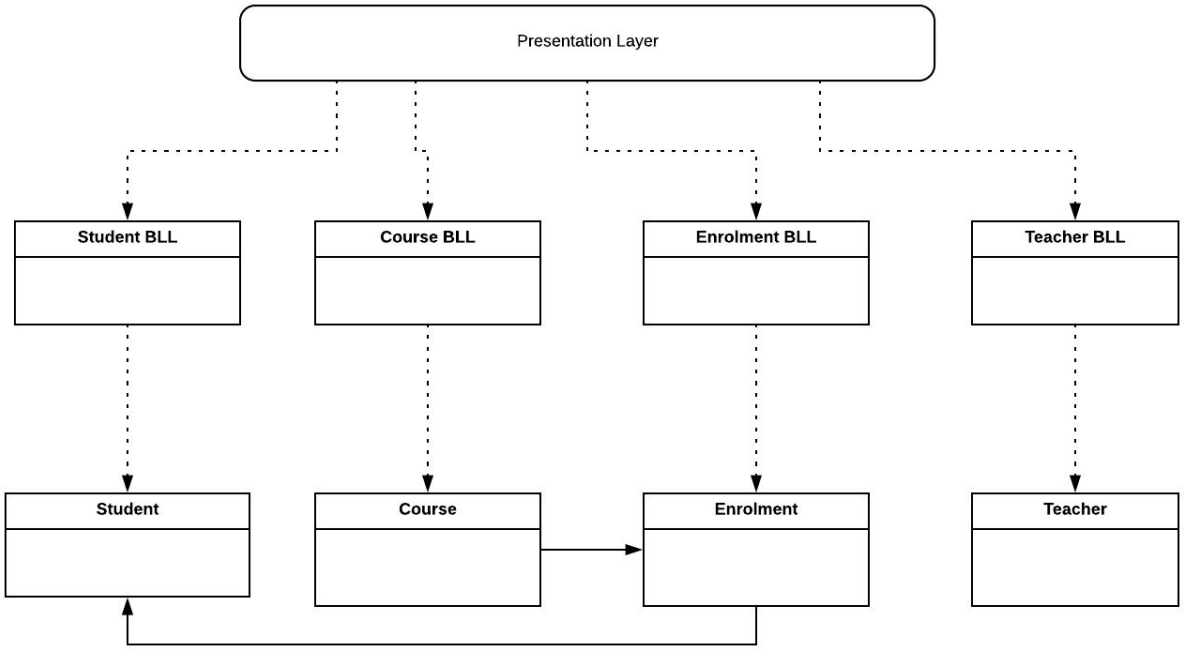
5. Class Design

5.1 Design Patterns Description

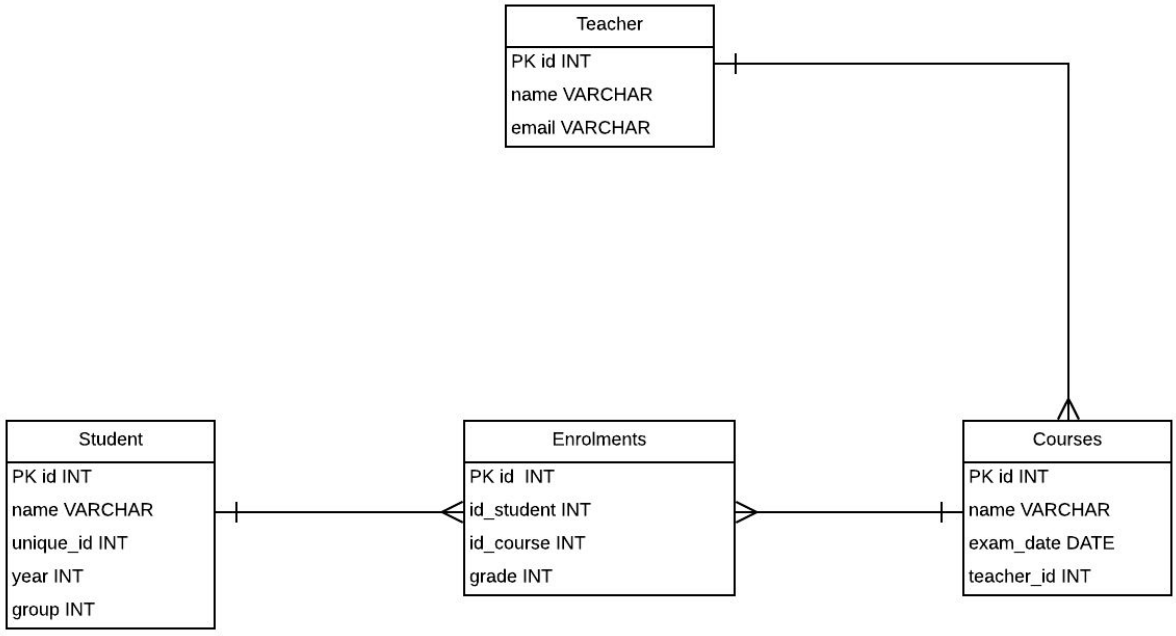
The Builder design pattern is a creational design pattern, designed to provide a flexible design solution to various object creation problems in object-oriented programming. The intent of the Builder design pattern is to separate the construction of a complex object from its representation.

The pattern is implemented with an interface, that provide the pattern with the ability to return itself, a terminate operation is used to return the final object.

5.2 UML Class Diagram



6. Data Model



7. System Testing

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

In object-oriented programming, the smallest unit is a method, which may belong to a base/ superclass, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.)

8. Bibliography

https://en.wikipedia.org/wiki/Builder_pattern

http://softwaretestingfundamentals.com/unit-testing/