

Students management Analysis and Design Document

**Student: Cordea Corina
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	4
3. System Architectural Design	5
4. UML Sequence Diagrams	6
5. Class Design	6
6. Data Model	7
7. Bibliography	8

1. Requirements Analysis

1.1 Assignment Specification

The application developed will be used for the management of students in the CS Department of TUCN.

1.2 Functional Requirements

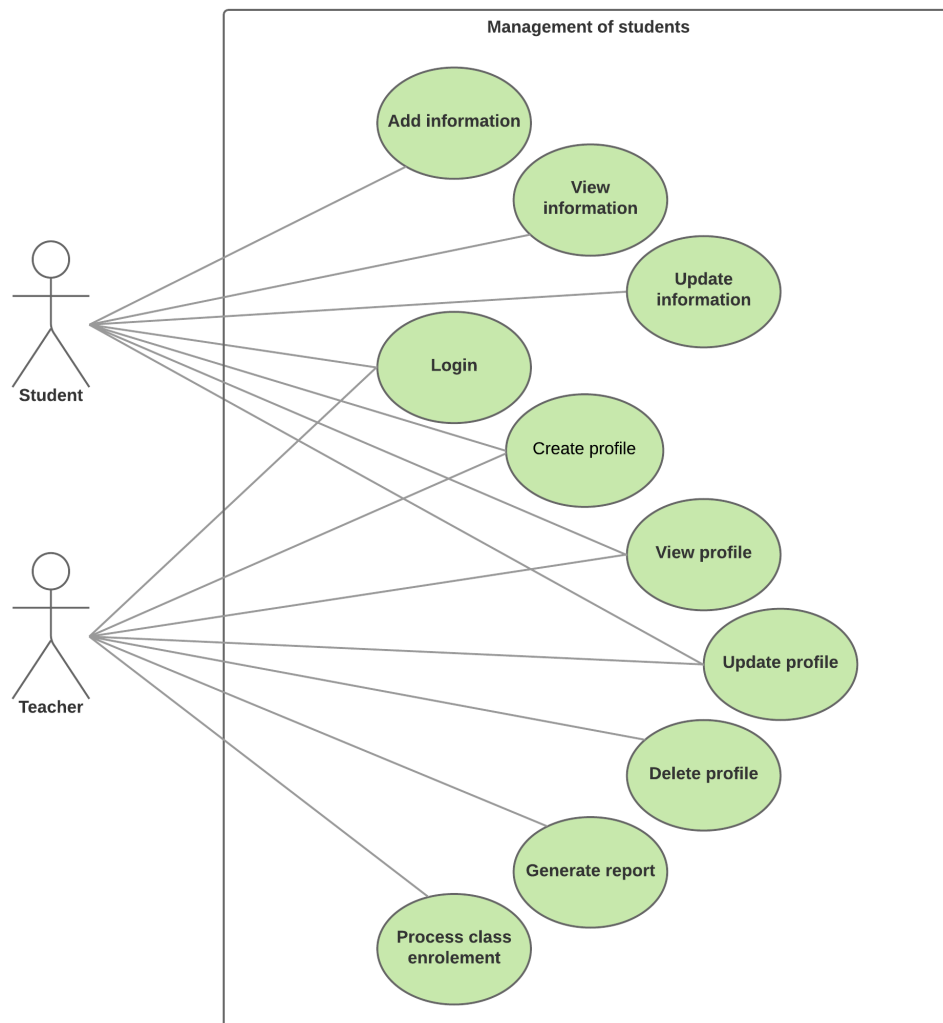
- The application should have two types of users (student and teacher/administrator user)
- Require a username and a password for users in order to use the application
- Add/update/view client information
- Create/update/delete/view student profile
- Process class enrolment.
- CRUD on students information
- Generate reports for a particular period containing the activities performed by a student
- Validate input against invalid data

1.3 Non-functional Requirements

- Availability – the system should achieve 95% available time
- Usability – the system should be easy to use by adult members, self-explanatory and intuitive

2. Use-Case Model

USE CASE DIAGRAM



Generate report

Use case: generate reports for a particular period containing the activities performed by a student

Level: user-goal level

Primary actor: administrator

Main success scenario: the administrator opens the application, logs into the system, selects generate report option, enters the student and the period, presses a button then the report is created

Extensions: - the administrator opens the application, enters the login information wrong and receives an

error message then enter it correctly and successfully logs in, selects generate report option, enters the student and the period, presses a button then the report is created

3. System Architectural Design

3.1 Architectural Patterns Description

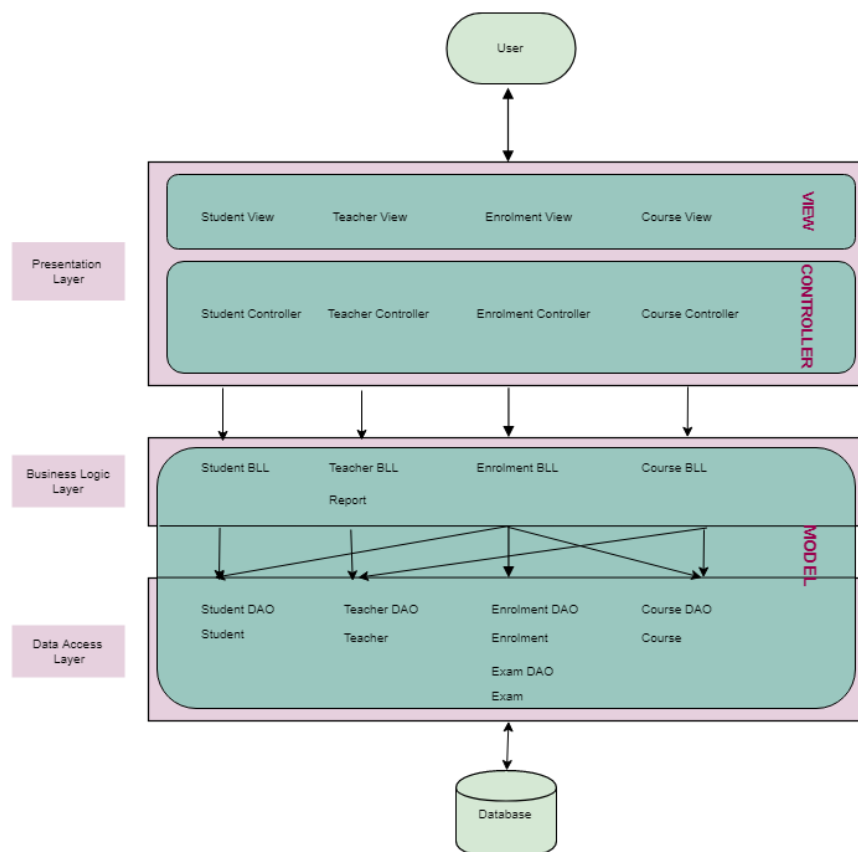
The Layers architectural pattern is used to structure the application by dividing it into groups of subtasks based on their functional responsibility. More specifically, the application is divided in 3 layers (bottom to top): data access, business logic and presentation layer. Each layer can only access the one beneath it. By using this pattern, the maintainability of the application and also the reusability of components are considerably increased.

Also, the MVC architectural pattern is used to separate application's concerns: handle input, processing and output.

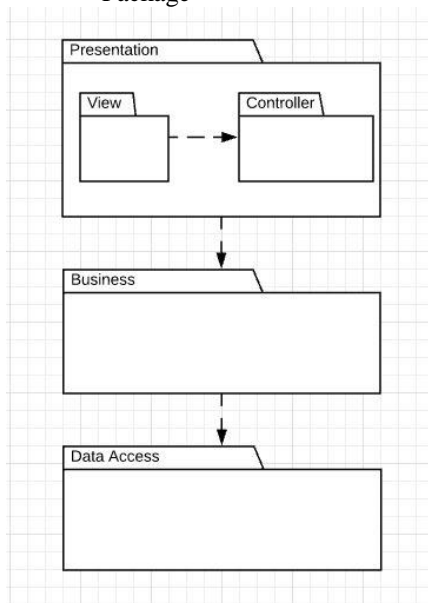
- The model component encapsulates data and functionality (processing).
- View components display information obtained from the model to the user (output).
- Each view has an associated controller component that handles input.

3.2 Diagrams

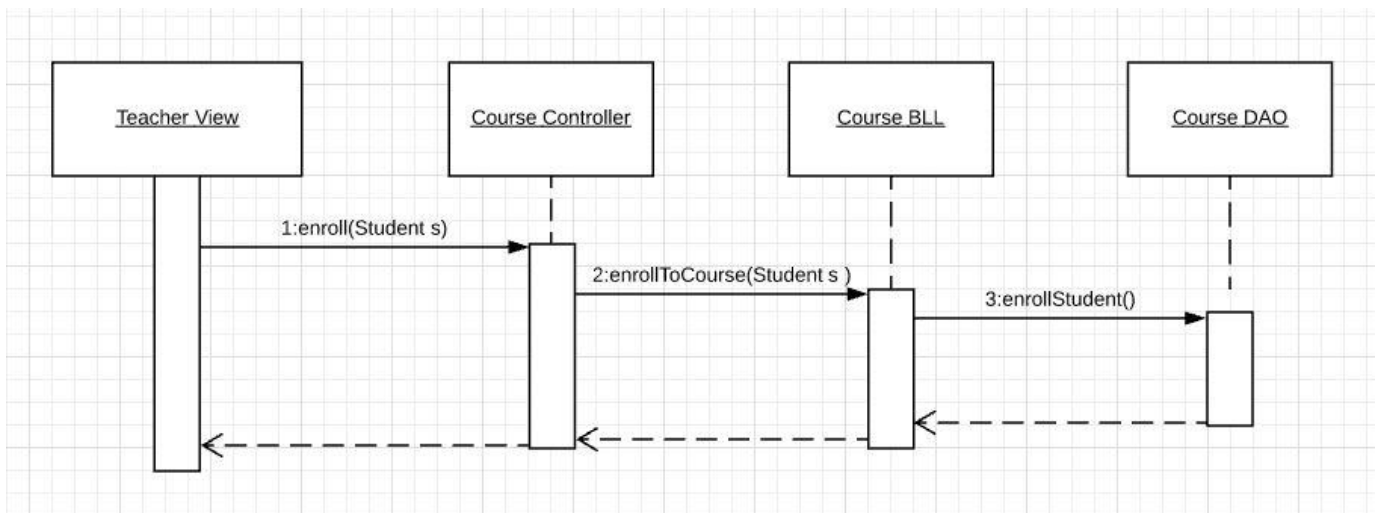
- Conceptual architecture



- Package



4. UML Sequence Diagrams



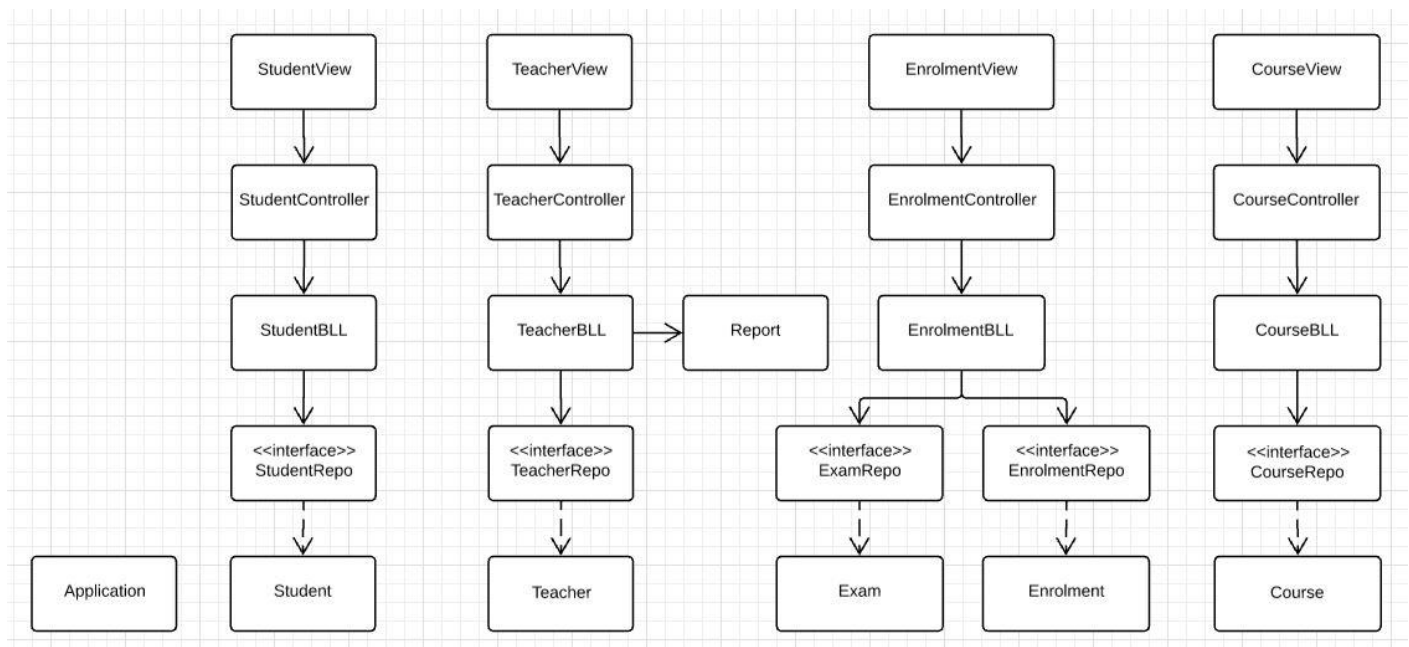
5. Class Design

5.1 Design Patterns Description

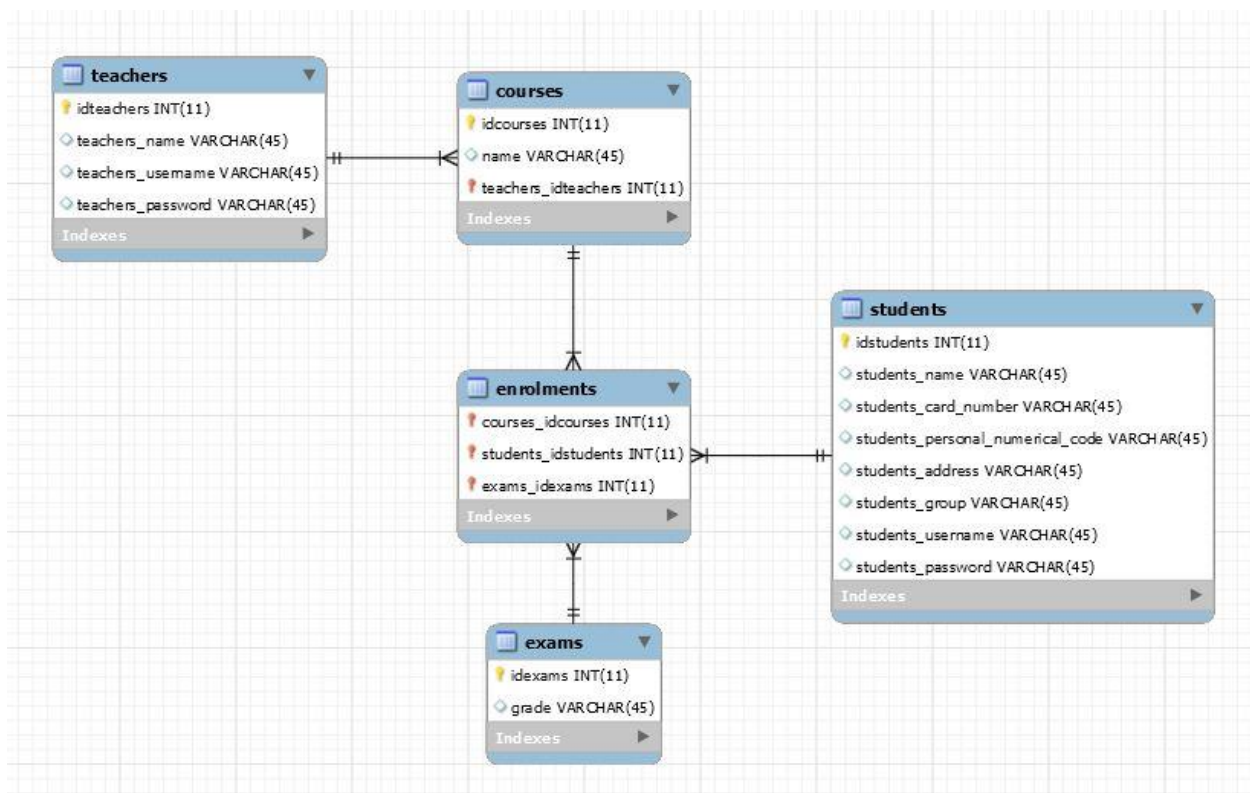
Singleton design pattern involves a single class which is responsible to instantiate itself. It only creates one instance so it is useful for accessing resources that need to be controlled. In this application it is used for database management.

Builder design pattern is a creational pattern that is used to simplify the creation of complex objects. It allows us to decompose clearly the object construction by using internal builder object that passes the values to a parent class. So, the complexity of object construction is hidden behind the builder, an internal static class that accepts the chained method invocation. In this application it is used to create Report objects.

5.2 UML Class Diagram



6. Data Model



7. Bibliography

<http://users.utcluj.ro/~dinso/PS2018/Lectures/>

<https://msdn.microsoft.com/en-us/library/ff650706.aspx?f=255&MSPPErr=-2147217396>

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

https://www.tutorialspoint.com/postgresql/postgresql_using_autoincrement.htm

<http://www.waitingforcode.com/spring-framework/design-patterns-in-spring-framework-part-1/read>

<http://mongodb.github.io/mongo-java-driver/3.4/driver/getting-started/quick-start/>