Student Management System
Analysis and Design Document

Student: Andreea Ionutas
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The application will serve as a management tool for the CS Department at UTCN. It can be used by both students and teacher/administrator. Having a user-friendly graphical interface and being connected to a database, the application can perform specific actions according to each type of user.

1.2 Functional Requirements

There are two types of users: student or teacher/administrator.

Operations that can be performed by a regular user:

- Add/update/view client information (name, identity card number, personal numerical code, address, etc.)
- Create/Update/Delete/View student profile (account information)
- Process class enrolment

Operations that can be performed by the administrator user:

- CRUD on students information
- Generate reports for a particular period containing the activities performed by a student.

1.3 Non-functional Requirements

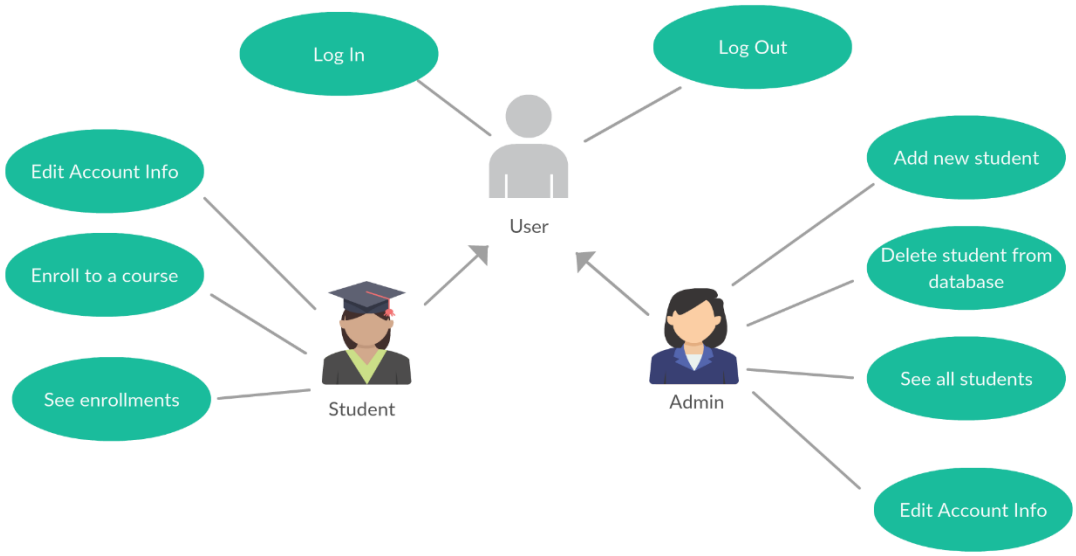
- Adaptability: the system should be able to adapt itself fast and efficiently to any type of changes.
- Stability: most of the objects will be stable over time and will not need changes
- Reusability: the system can be used in various platforms/contexts.

2. Use-Case Model

Use case: Generate report
Level: User Goal Level
Primary actor: Administrator
Main success scenario:

- Log in into application
- Select a student
- Click on generate report
- Visualize a report containing the student's performances

Extensions: Fail to log in into application (the login id or password were introduced incorrectly)



3. System Architectural Design

3.1 Architectural Pattern Description

The architectural pattern used is Layers. The most widespread use of multitier architecture is the three-tier architecture.
N-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application. A three-tier architecture is typically composed of a presentation tier, a business logic tier, and a data storage tier.

- ***Presentation layer***

  Presentation of the web pages, UI forms and end user interracting API's

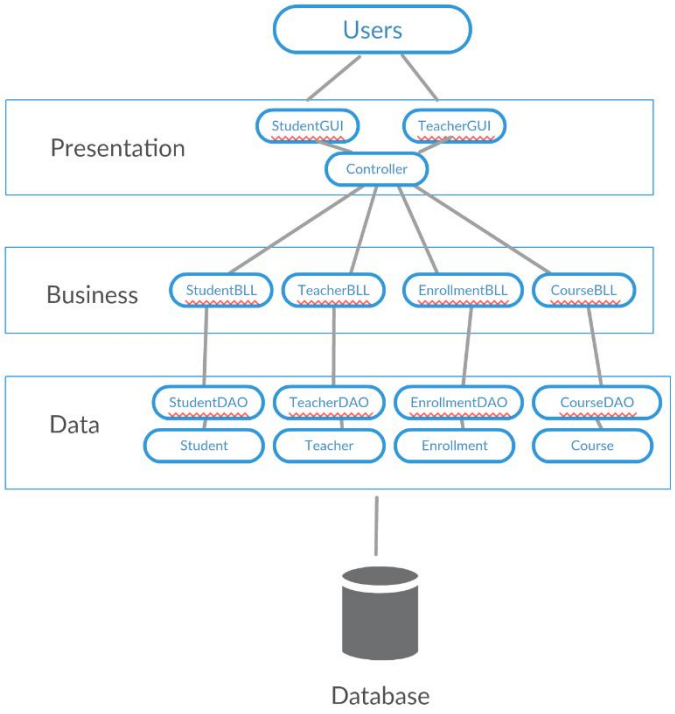
- ***Business layer***

  The logic behind the accessibility, security and authentication happens in this layer.

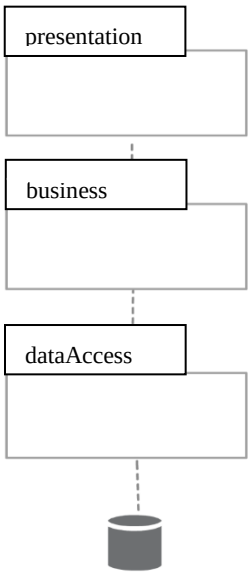
- ***Persistent layer***

  This is the presentation layer for the Data. This includes the DAO (Data Access Object) presentation, ORM (Object Relational Mappings) and Other modes of presenting persistent data in the application level.

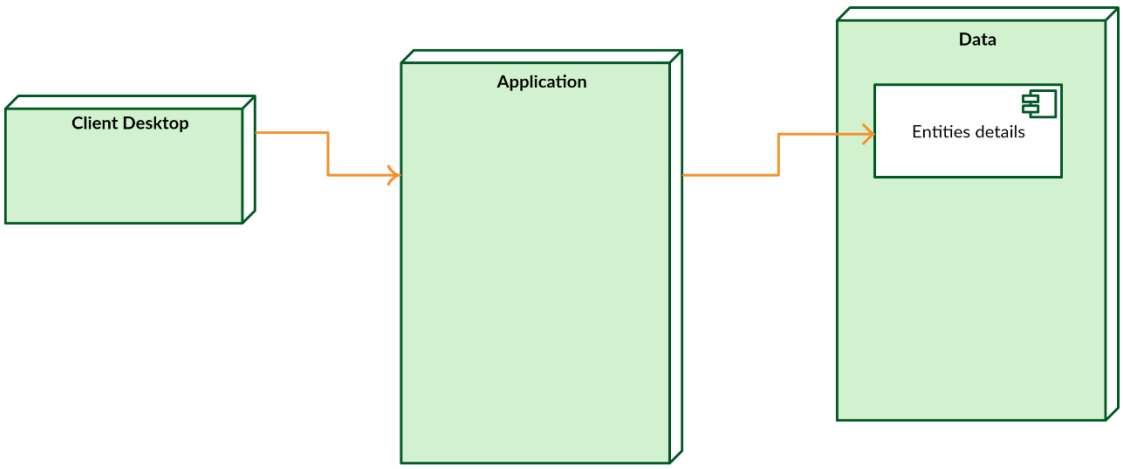
3.2 Diagrams



Layered Architecture

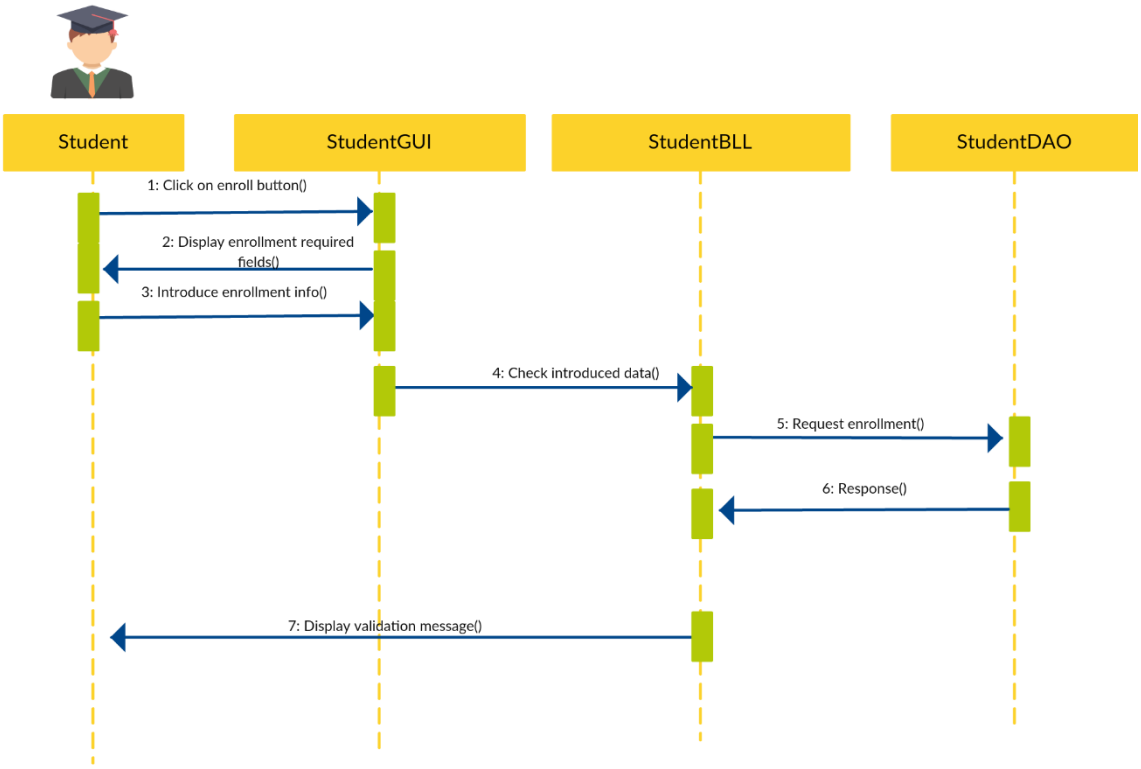


Package Diagram



Deployment diagram

4. UML Sequence Diagrams



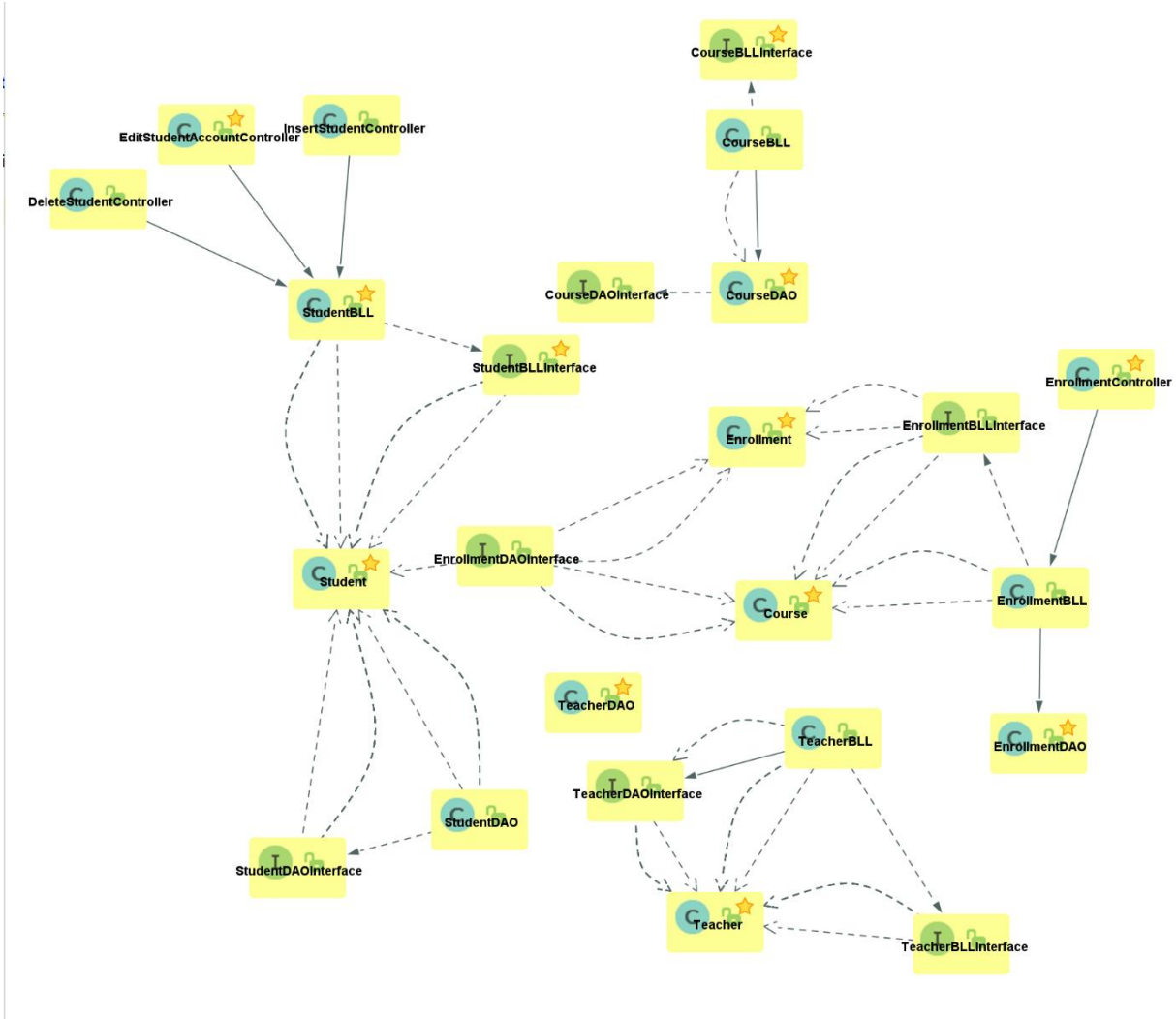
Sequence Diagram for Student Enrollment Process

5. Class Design

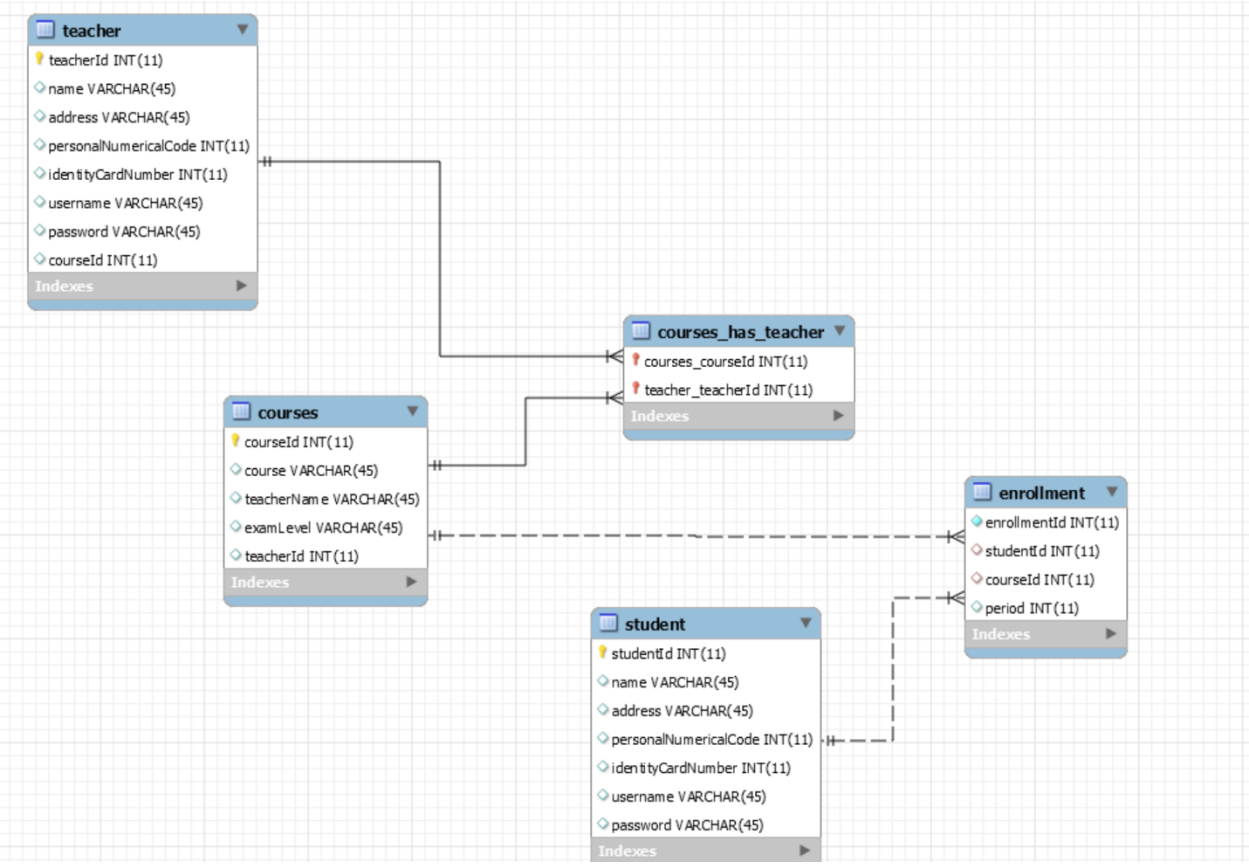
5.1 Design Patterns Description

In software engineering, the singleton pattern is a software design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. In terms of practical use Singleton patterns are used in logging, caches, thread pools, configuration settings, device driver objects. In our system it will be used together with Factory Design Pattern.

5.2 UML Class Diagram



6. Data Model



7. System Testing

The system will be tested mainly with unit tests. Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. For the unit testing, we will use Junit4 together with Mockito.

8. Bibliography

https://en.wikipedia.org/wiki/Singleton_pattern
https://www.tutorialspoint.com/uml/index.htm