Assignment 1
Analysis and Design Document

Student: Ciucescu Vlad Andrei
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implement a Java application for the management of students in the CS Department at TUCN.

1.2 Functional Requirements

The application should have two types of users (student and teacher/administrator user) which must provide a username and a password to use the application.
The regular user can perform the following operations:

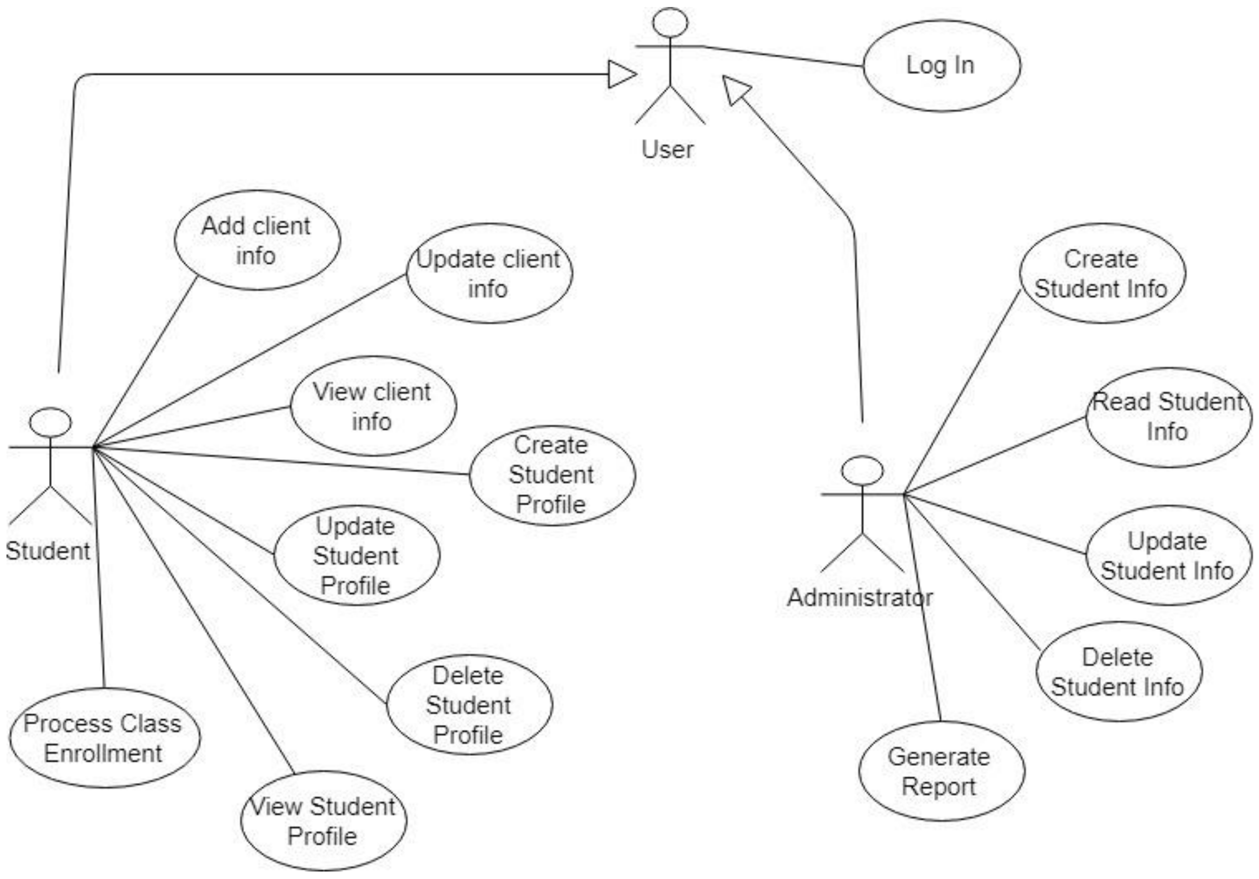
- Add/update/view client information (name, identity card number, personal numerical
  code, address, etc.).
- Create/update/delete/view student profile (account information: identification number,
  group, enrolments, grades).
- Process class enrolment (enroll, exams, grades).

The administrator user can perform the following operations:

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

2. Use-Case Model

Use case: Update Profile
Level: user-goal level
Primary actor: Student
Main success scenario: The Student presses the "Update Profile" button, edits some fields, presses the "Save Changes" button and the changes to the profile are saved.
Extensions: alternatively, the student could press the "Cancel" button, which cancels any modifications to the profile. If the student introduced invalid data, pressing the "Update Profile" button will not update the profile, but instead will display an error message.



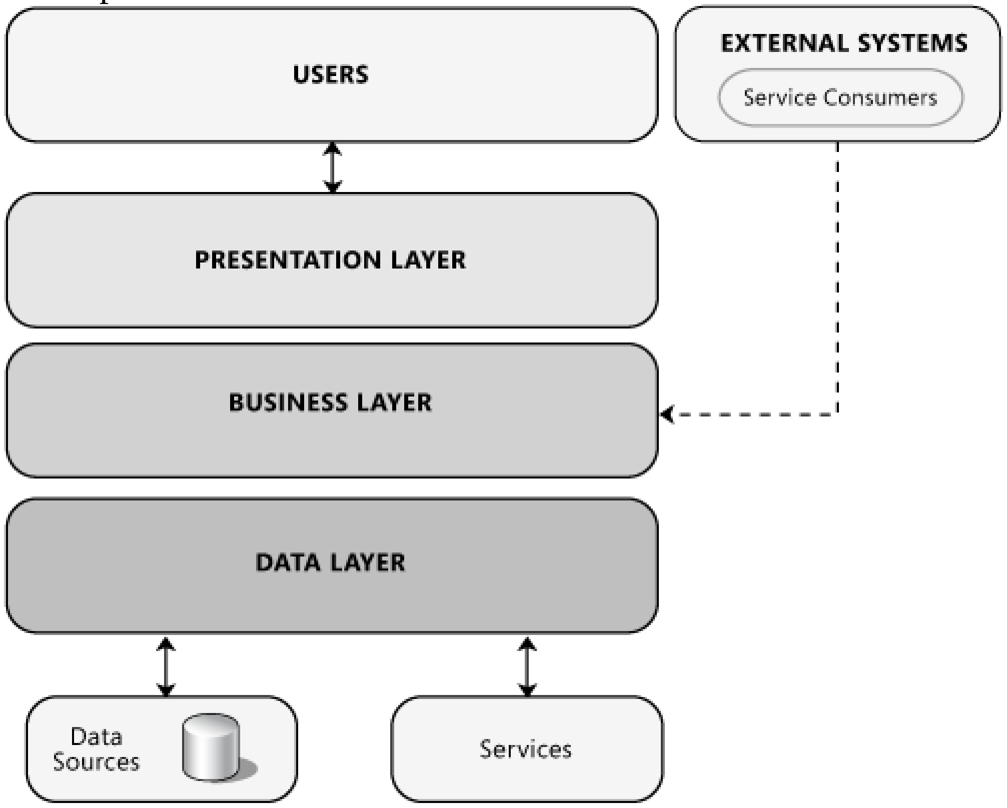
3. System Architectural Design

3.1 Architectural Pattern Description

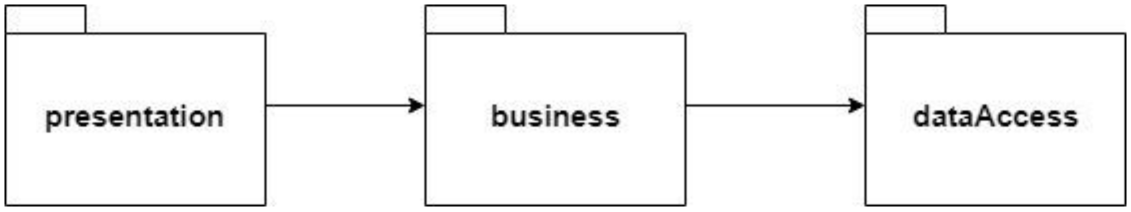
The main architectural pattern used for the application is the layered architecture pattern. Irrespective of the type of application being created, one can decompose the design into logical groupings of software components. These logical groupings are called layers. Layers help to differentiate between the different kinds of tasks performed by the components, making it easier to create a design that supports reusability of components. Each logical layer contains several discrete component types grouped into sub layers, with each sub layer performing a specific type of task.

3.2 Diagrams

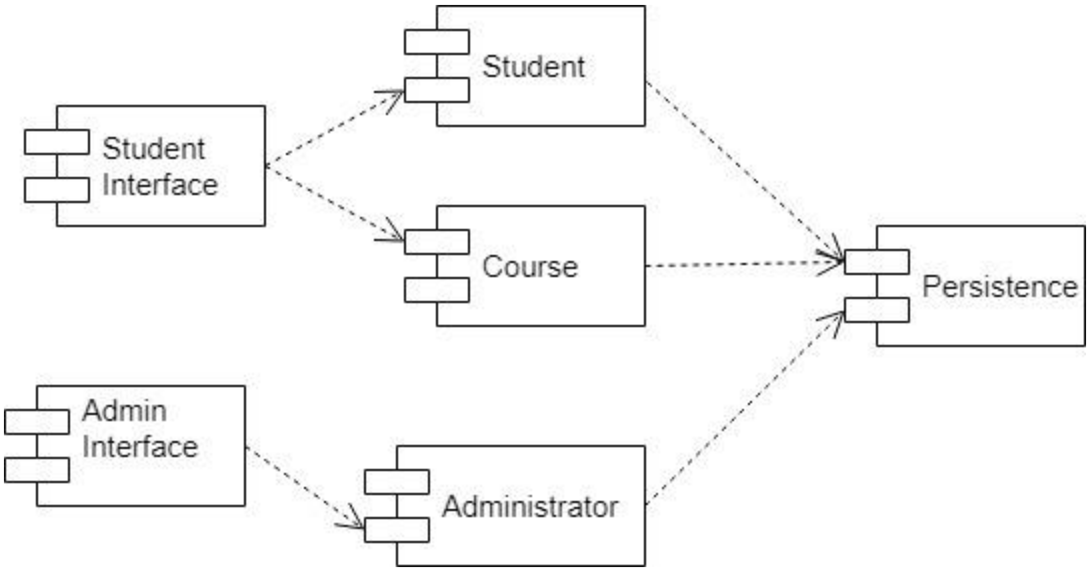
Conceptual Architecture



Package Diagram

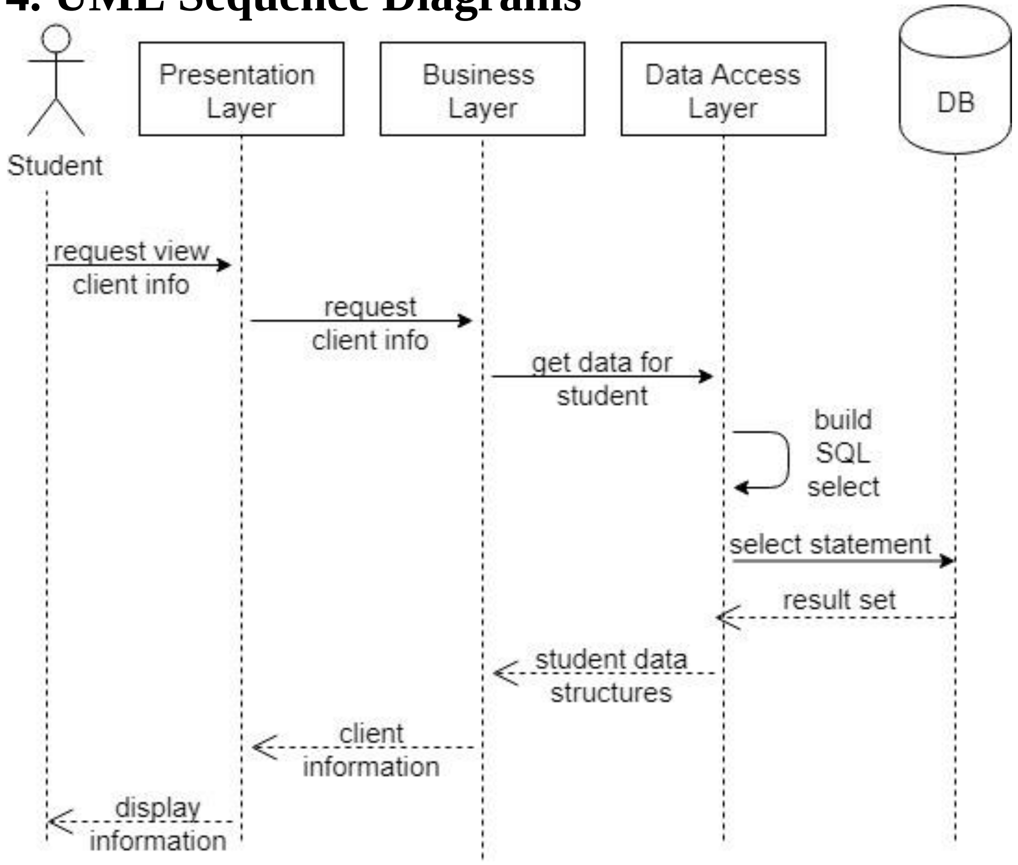


Component Diagram



The is a standalone application, supposed to run on a single computer. As such, no deployment diagram is provided. (deployment diagrams are used for applications that are deployed to several machines)

4. UML Sequence Diagrams

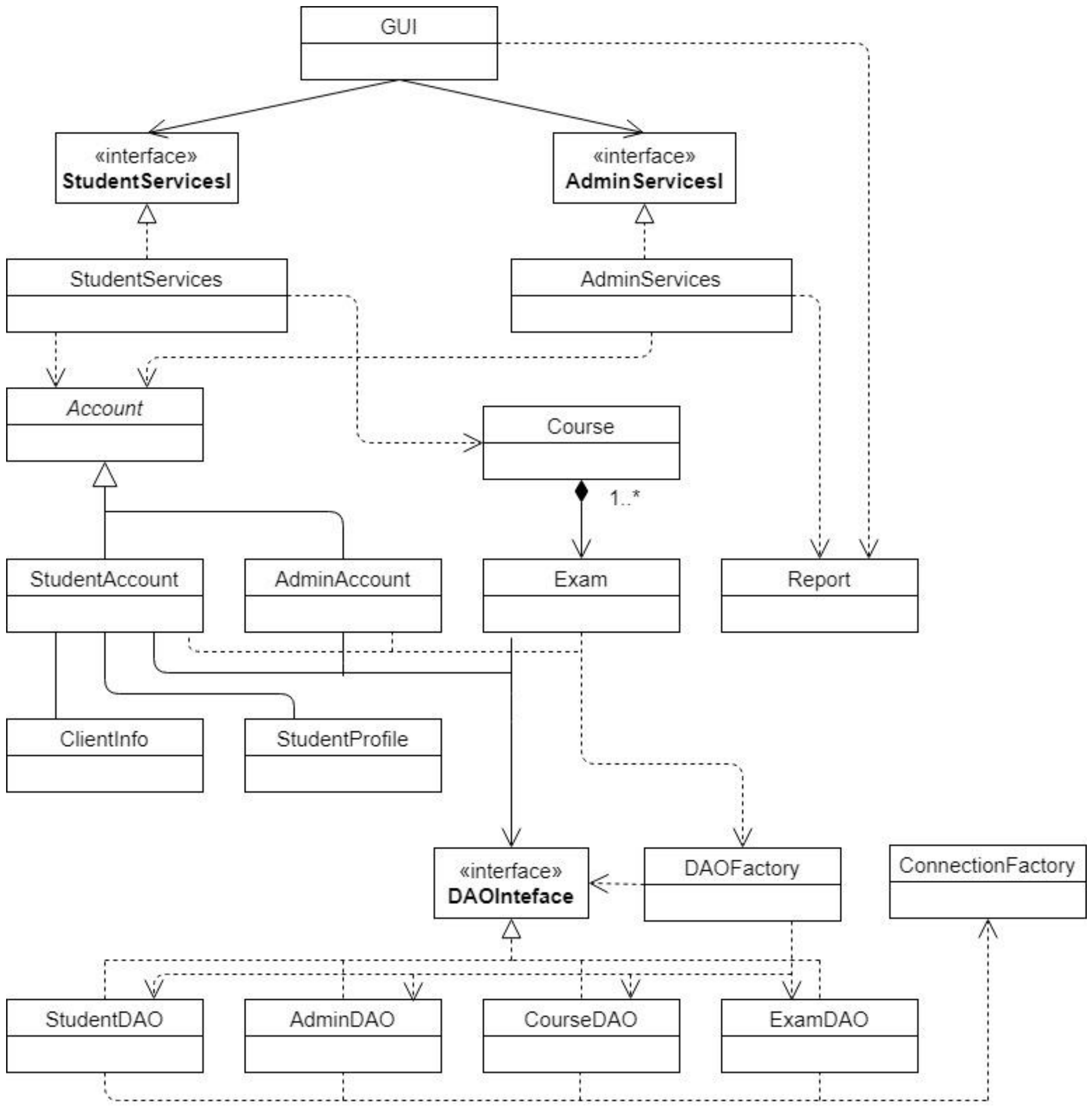


5. Class Design

5.1 Design Patterns Description

One of the used patterns is the Factory method pattern, used in the data access layer. By using this pattern, the responsibility of creating the object is delegated to the factory class and the client is not exposed to the creation logic. Specifically, the 'ConnectionFactory' class will be used by the data access objects to create connections to the database, without them knowing how the connections are created and the 'DAOFactory' class will be used by classes in the business layer to create data access objects.

5.2 UML Class Diagram



6. Data Model

The main data model used is the entity-relationship model, especially UML, used to visualize the design of the system and the relationship between the different components.

7. System Testing

The testing strategy used is unit testing. A unit is the smallest testable part of software. As such, unit testing involves testing individual units or components (such as methods) of a software. Its purpose is to check whether each unit of the software performs as designed.

For this application, unit testing will be performed with JUnit. This is a Java library used for testing source code, which has advanced to the de-facto standard in unit testing. By using JUnit, one can assert whether methods work as designed, without the need to set up the complete application.

As testing methods, equivalence partitioning and boundary analysis are preferred, as they reduce the total number of test cases and ensure that testing is comprehensive, by choosing representative test cases for the input.

8. Bibliography

http://softwaretestingfundamentals.com/unit-testing/
http://www.methodsandtools.com/tools/tools.php?junit
http://www.agilemodeling.com/artifacts/
https://msdn.microsoft.com/en-us/library/ee658109.aspx
https://en.wikipedia.org/wiki/Factory_method_pattern
Factory Method - Design Patterns – Gamma, Helm, Johnson, Vlissides