Assignment 1
Analysis and Design Document

Dan Fulga
Department of Computer Science
Technical University of Cluj-Napoca
Group 30432
danfulga96@yahoo.com

March 21, 2018

Contents

1 Requirements Analysis

1.1 Assignment Specification

The task of this assignment is to implement a Java application which can be used for the management of students in the Computer Science department at UTCN.

1.2 Functional Requirements

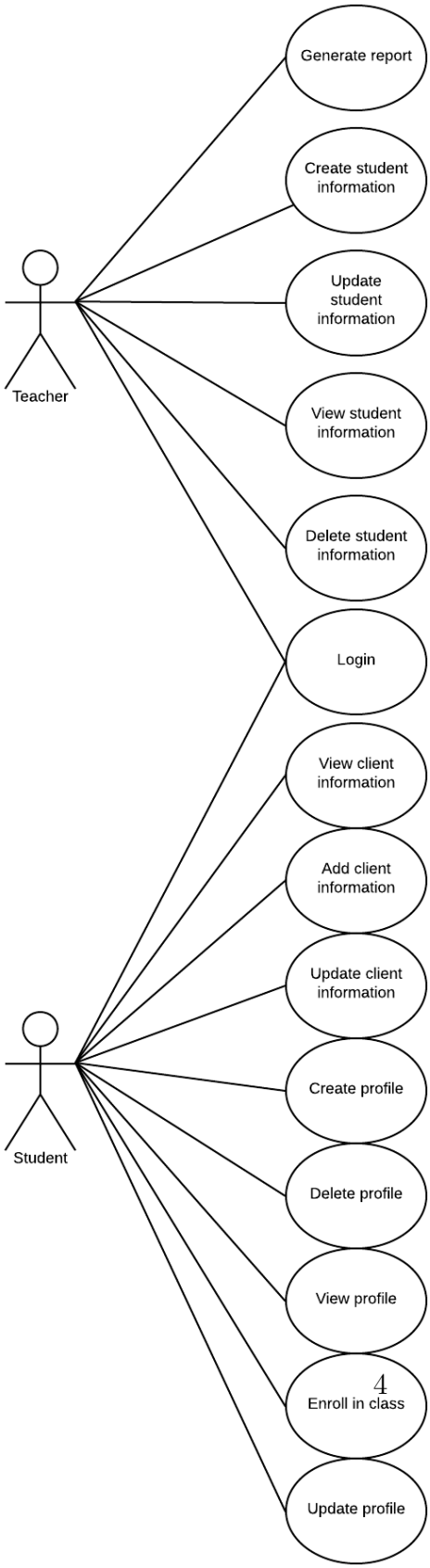
The application should have two types of users (student and teacher/administrator user) which both have to provide an username and a password in order to use the application.

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.

2 Use-Case Model

Use Case: Log In
Level : User-goal level
Primary actor: Student
Main success scenario:The student introduces an username and a password and he is succesfully logged in.
Extensions: 1. In case of failure, the student will see errors such as "user not found" or "invalid password".
2. Data related errors, such as "password must be at least 6 characters long, and contain at least one uppercase and/or one numerical character ".

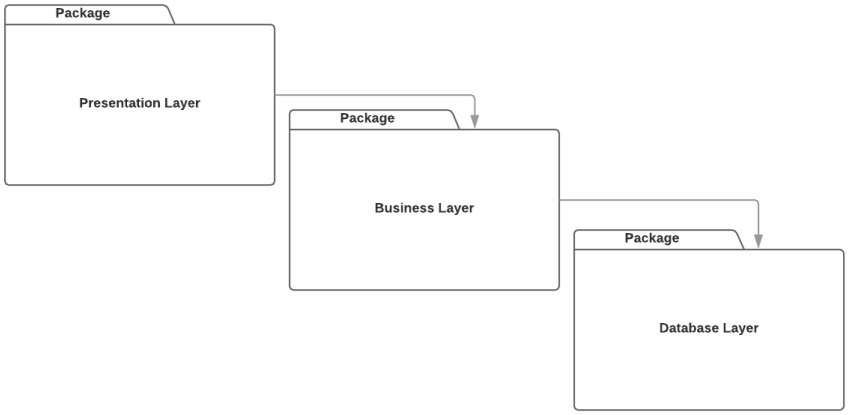

Generate report
Create student information
Update student information
View student information
Delete student information
Login
View client information
Add client information
Update client information
Create profile
Delete profile
View profile
Enroll in class
Update profile
Teacher
Student

3 System Architectural Design

The main design pattern used for implementing this application is the layered architecture pattern. The components present here are organized into layers. Each layer of the layered architecture pattern has a specific role and responsibility within the application.

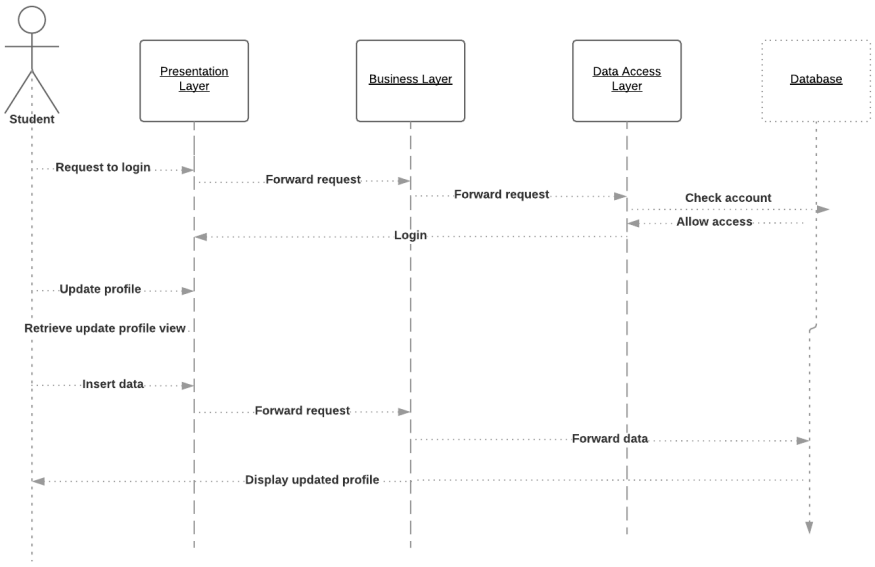
4 Diagrams

4.1 Package Diagram



4.2 Component Diagram

5 UML Sequence Diagram



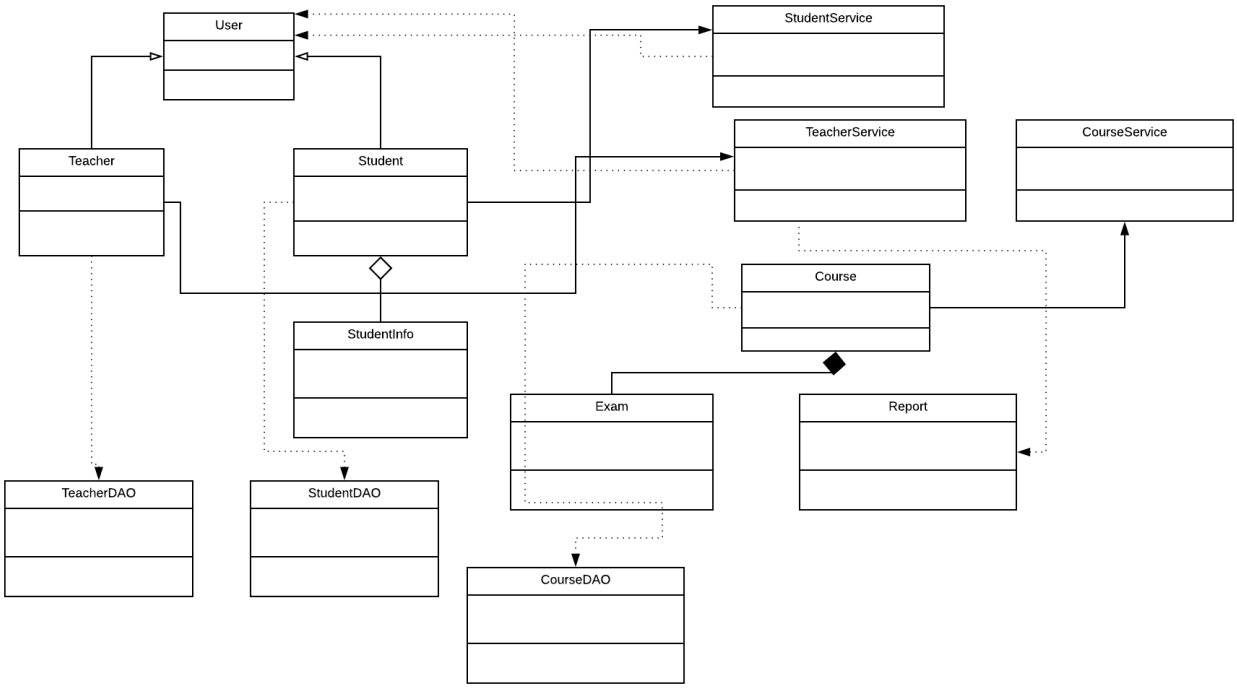
6 Class Design

6.1 Design Pattern Description

Layering is a common architectural pattern employed to help break up complicated software. Each layer is responsible for exposing a principal service, and layers are typically divided based on their primary functional responsibilities. Packages are an excellent way to represent the logical layers composing an application. Higher-level layers (or a package representing a layer) sit atop lower-level layers, using their services. Lower-level layers, however, know nothing of the higher-level layers that use them. Furthermore, each layer usually hides its lower layers from the layers above, though the transparency of individual layers must be a conscious decision. In this application, the layers are:
-Presentation Layer : responsible for display of application user interface components and handling user requests
-Business Layer : responsible for the application specific business logic
-Data Layer : facilitates communications with underlying services such as database connectivity.

The Factory Design Pattern will be used in order to create the Connection-Factory. In this pattern, we create the object without exposing the creation logic to the client and refer to the newly created object using a common interface.

6.2 UML Class Diagram



7 Data Model

8 System Testing

The application will be tested using JUnit. JUnit is a test framework which uses annotations to identify methods that specify a test. Some advantages when using JUnit are:
- is the standard library for unit testing in Java
- is supported by all major IDEs

- has extensions for various purposes beyond unit testing Java objects.

9 Bibliography

https://coderanch.com/t/95022/engineering/pros-cons-JUnit
https://www.lucidchart.com
https://www.techrepublic.com/article/commonly-used-architectural-patterns-in-java-applications/