# Advanced programming assignment

# Student name: Monika Baricza

# Student number: 10389618

# Teacher’s name: Paul Laird

# 25/06/19

## Introduction

The aim of the assignment was to develop a student management system for DBS. The project’s learning outcome is to get more familiar with the windows form applications and to learn how to manage databases that are connected to the application.

### Requirements

Requirement 1: the application must be built using a tiered architecture

Requirement 2: the application must contain a login screen and only a file menu should be visible

Requirement 3: a working login system that validates the username and password

Requirement 4: once the user is logged in the menu should change according to the user type

Requirement 5: a grid should be displayed with student data on the welcome screen

Requirement 6: ability to enrol a student

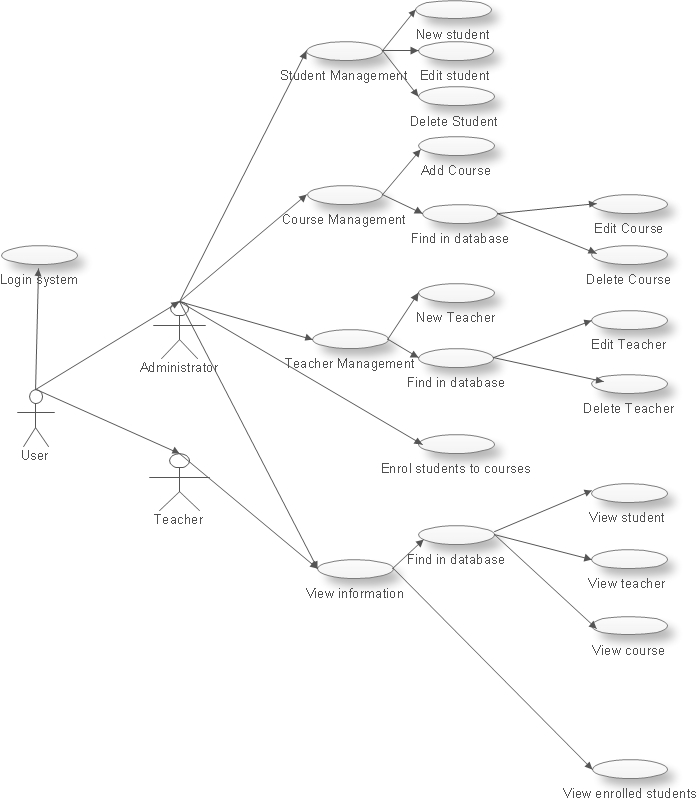
Requirement 7: ability to update students

Requirement 8: the system should provide for extracting data in XML format

## UML diagrams

### Use Case Diagram

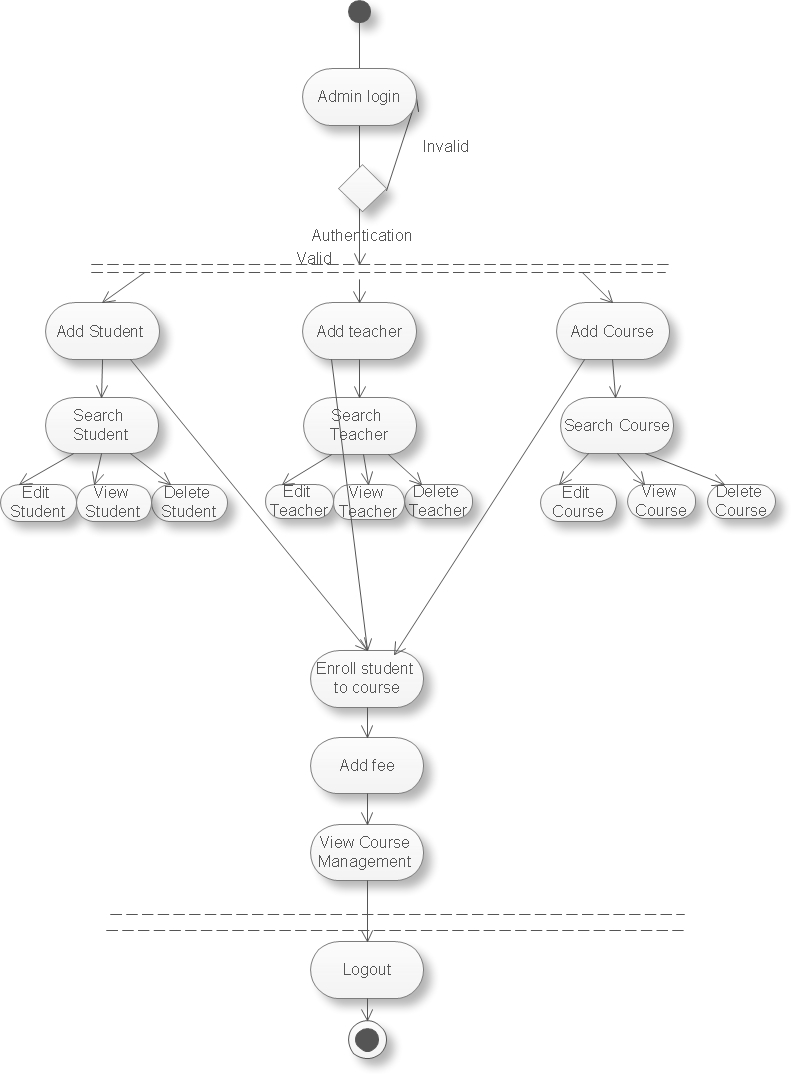
The use case diagram can be found under Figure 1 below. As you can see, this diagram shows that the user can sign in as a regular user or as an admin. If the user is an admin it means that they have a much wider access to managing data in the application. They can add, edit, delete and view students, teachers or courses while a regular user can only view them. Admins can also enrol students to courses and set the fee for that student for that particular course. The user model itself can be improved further later on; at the moment the admin doesn’t manage users. There is a search engine built for the application which is used to find students, courses or teachers in the database.



Figure

### Activity Diagram

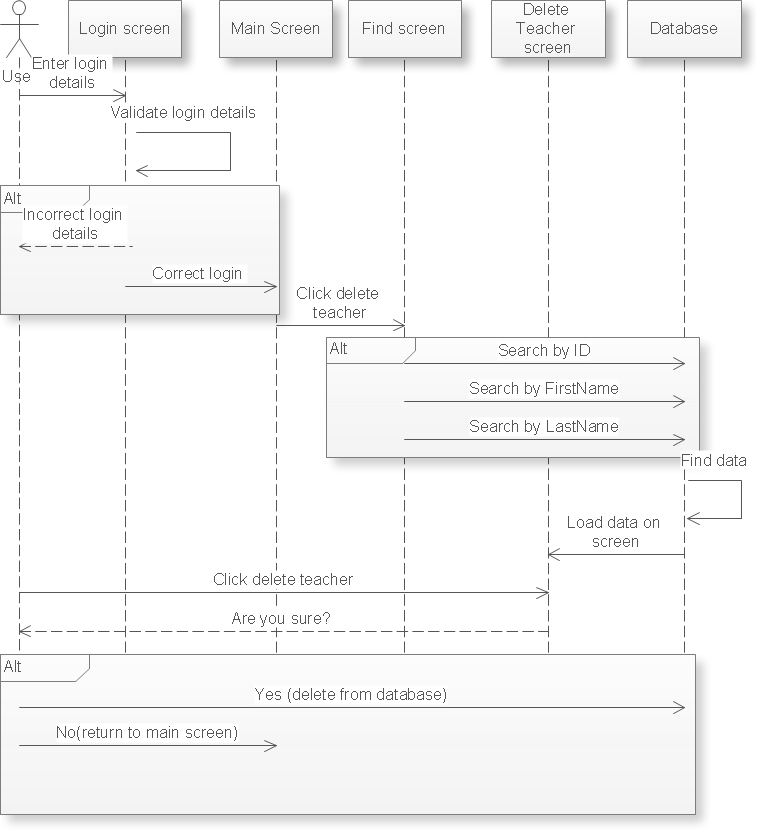
The activity diagram shows what an admin can do in the application. After validating the login credentials, they can add a student, add a teacher, or add a course. After adding data to the database, they can manage the data by editing, deleting students, teachers or courses. Data can be found by entering details into the search engine which pops up after “edit”, “delete” or “view” has been clicked. The course table includes a foreign key referring to the teacher table, so if the admin deletes a teacher, it will automatically delete the course related to the teacher. The admin can also enrol a student to a course by selecting the student from the dropdown list and then selecting a course from the dropdown list. On this screen they add the fee for the course for the student as well. They can also view student and course relations under Course Management -> Course Management Info tab.



Figure

### Sequence diagram for deleting teacher

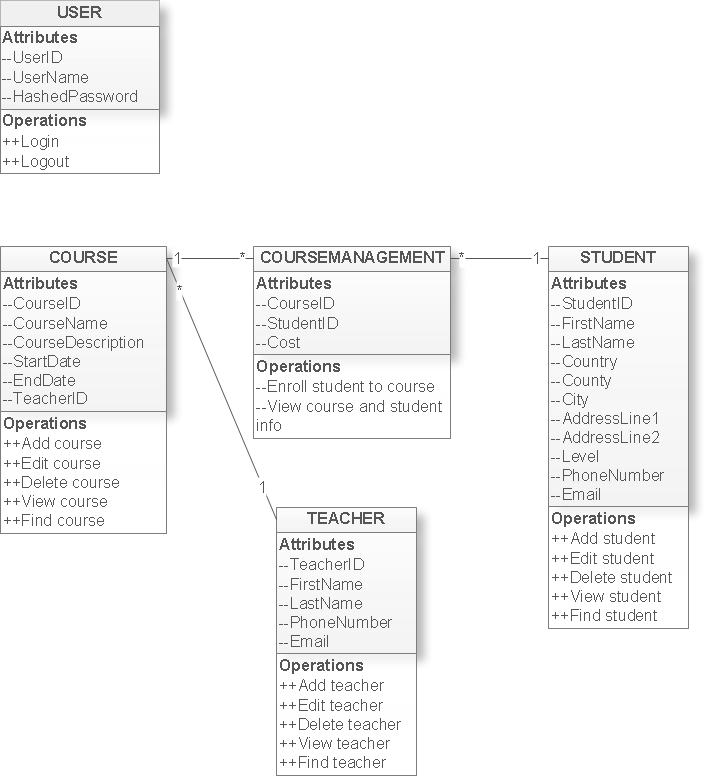
This is an example of how the application works. If the admin signs in with the right credentials, the main screen appears then they click Teacher -> Delete Teacher (assuming there is already at least one teacher in the database). The Find in Database screen pops up, where the admin can search by ID, first name or last name. While the ID has to be an exact number, it is enough to enter a few letters of the first name or the last name. Once they have found the teacher that they are looking for, they click on the right row in the datagrid which displays the search results. Then the delete teacher screen pops up, where if they hit “Delete”, the teacher and all associated records are going to be deleted from the database. If they decide not to go ahead with the deletion, the application returns to the main screen.



Figure

### Class diagram

The following class diagram represents the database structure. The Teacher and the course have a one to many relationship as a teacher can teach many courses but a course can only have one teacher. The course and the student tables have a many to many relationship as a student can attend many courses and a course can be attended by many students. This has been resolved by adding the course management table, which brakes the many to many relationship down to two one to many relationships. As it was mentioned before the user table is not being managed by any of the users at the moment, this is subject to further improvement (a role could be added to the user table to define if the user is a teacher or an admin).



Figure

## Matching the requirements