Student Management Application
Analysis and Design Document

Student: Ioan Iustin Fodorut
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The application we are implementing for this assignment has the purpose of helping the Computer Science Department at the Technical University of Cluj-Napoca manage students more easily, efficiently and offer to both groups of the academic community new ways of interacting with each other by developing an educational process that is reliable, easy to follow and is based on the principles of transparency.

1.2 Functional Requirements

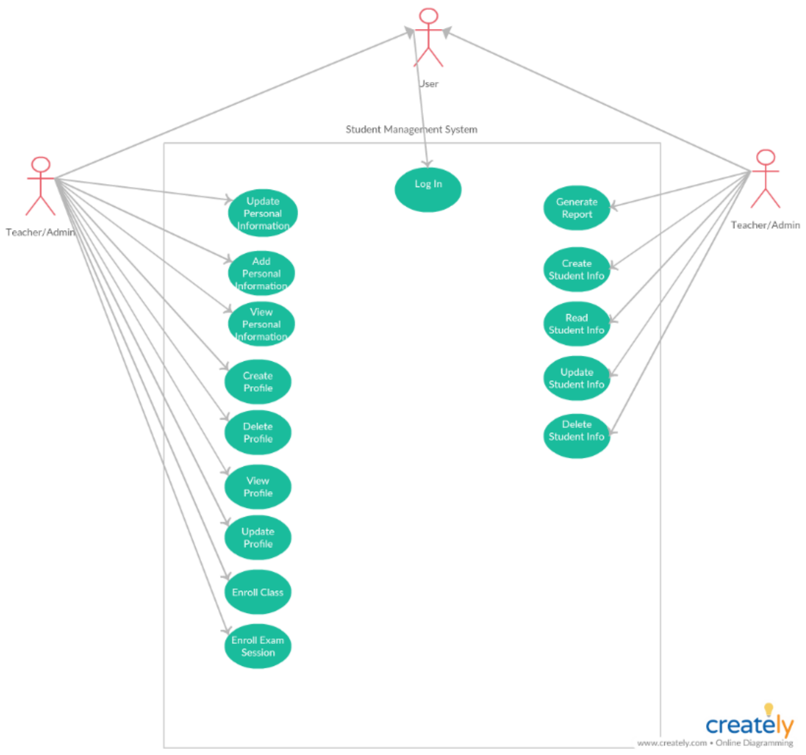
The application should have 2 classes of users: students and teachers.
The later class of users extends the functionalities offered for the first class (add, update, view and delete personal and public info, process class enrollment) with the role-specific features (CRUD, report generation).

1.3 Non-functional Requirements

1.3.1 *Performance:* For 90% of users requests over one hour, the server will grant access in 3 or less seconds after a request is received.
1.3.2 *Usability:* The system is very easy to use by experienced computer users, with a learning time of less than 5 minutes, while an unexperienced computer user should be able to adapt in less than 10 minutes to the application UI.
1.3.3 *Accessibility:* Any operation should be able to be accessed in less than 3 screens from the login screen.
1.3.4 *Security:* The personal user password stored on the server should be encrypted to a user that accesses the data base physically. A powerful algorithm that can be used is SHA356 algorithm. Data validators will be used against all input data.
1.3.5 *Portability:* The application can be accessed from any desktop computer running Java, with a connection to internet.

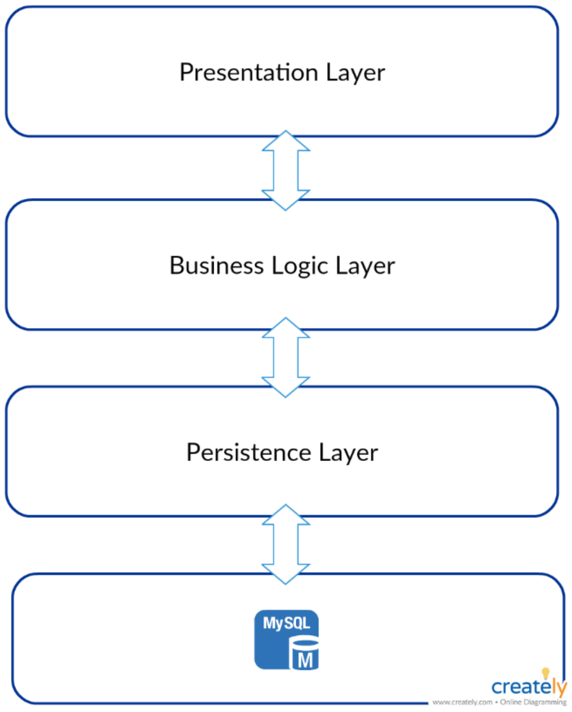
2. Use-Case Model

Use case: Create Profile
Level: user-goal level
Primary actor: student
Main success scenario: The student pressed the “Create profile” button. A new screen/window appears. The screen contains fields for profile data. The data is introduced. The student presses the “Create” button and the profile is created.
Extensions: If the input data is not validated, the student is informed of the invalidity of the data and is prompted to introduce correct data that fits the specified parameters. The student can also press the “Back” button, which takes him back to the main menu, without creating a new profile.



3. System Architectural Design

3.1 Architectural Pattern Description

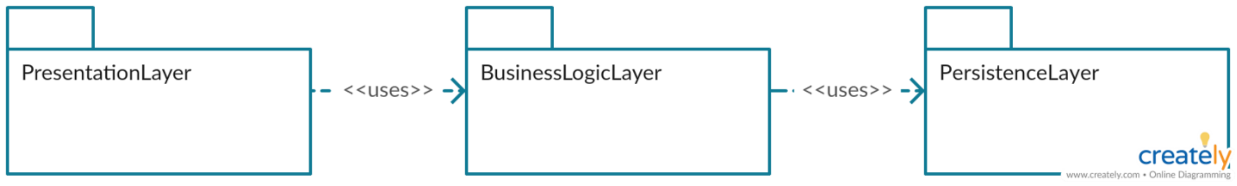


The most important architectural pattern that is used in this application, which shapes it entire macro-structure is the **Layer Architectural Pattern**. This architecture is a well-known standard in Java EE applications, being used frequently in business application development. The layer architecture divides the application in 4 big closed groups of classes, called layers, physically implemented using packages: presentation, business, model/persistence and data/-base. Each **layer** has a specific purpose in the application, abstracting the work needed for a certain step in a business process.

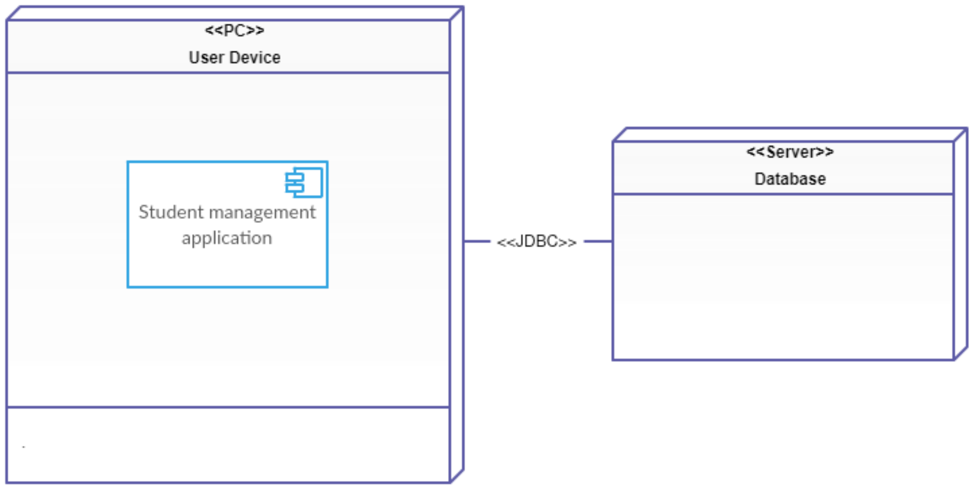
Another architectural pattern that is used is the **Model-View-Controller Architectural Pattern**. The pattern is very popular among Java applications, since it isolates the human interaction with the application from the application data representation and the operations that are executed on it.

3.2 Diagrams

3.2.1 Package Diagram



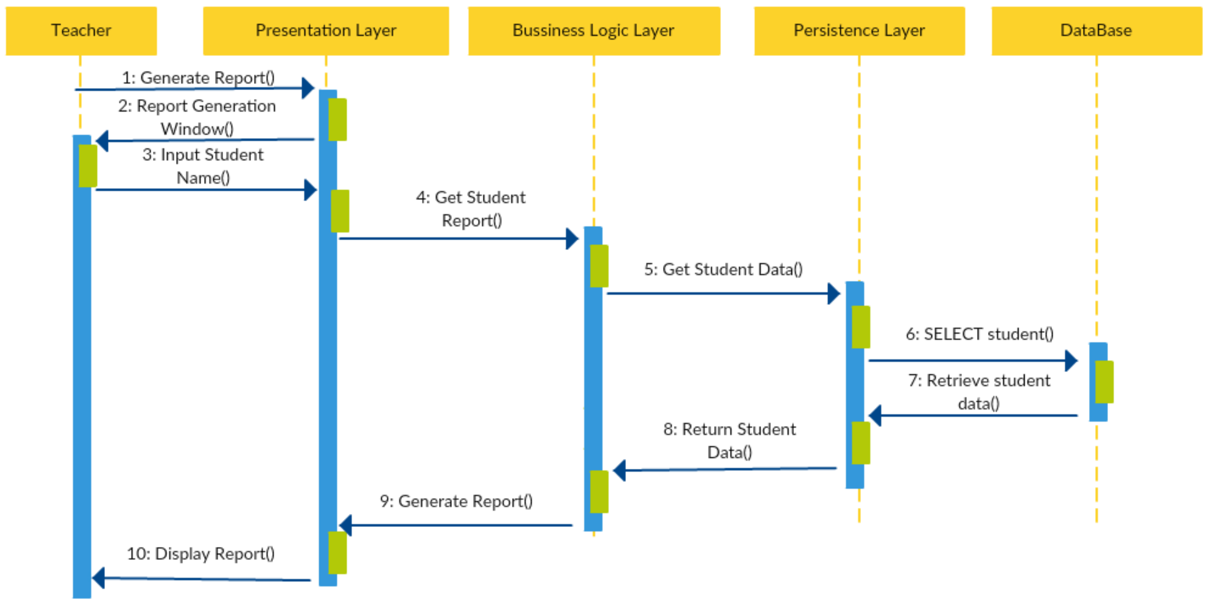
3.2.2 Deployment Diagram



The application can store the database on both localhost (for demo purposes) or on a different device – a server in real life scenarios.

3.2.3 Component Diagram

4. UML Sequence Diagrams



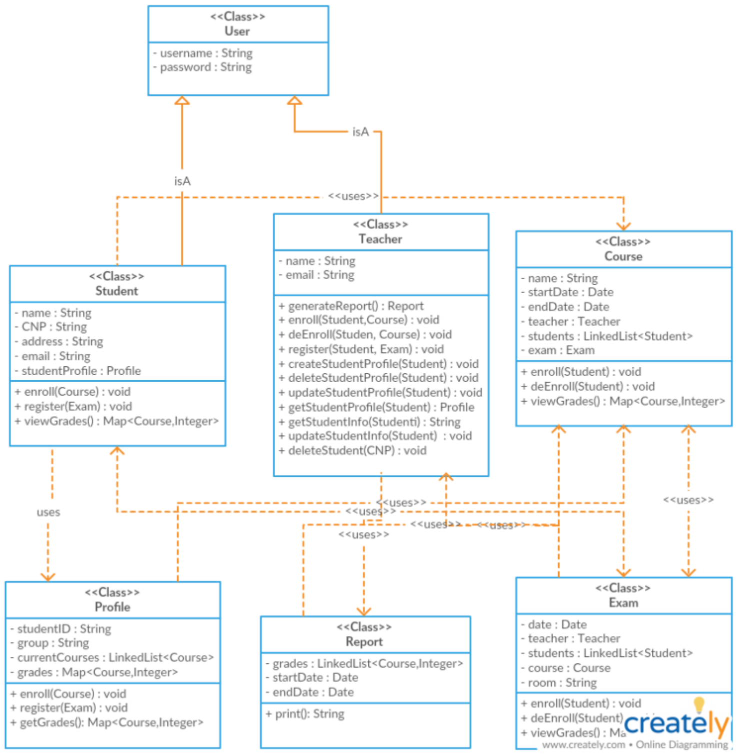
The sequence diagram represents the report generation operation.

5. Class Design

5.1 Design Patterns Description

5.1.1 Observer Design Pattern: used in the implementation of the view-controller relationship, by implementing the event listener interface in the GUI classes.
5.1.2 Factory Design Pattern: used for the creation of the connection with the database.
5.1.3 Singleton Design Pattern: only one connection factory is necessary, so a singleton is appropriate.

5.2 UML Class Diagram


<<Class>>
User
- username : String
- password : String
isA
isA
<<uses>>
<<Class>>
Student
- name : String
- CNP : String
- address : String
- email : String
- studentProfile : Profile
+ enroll(Course) : void
+ register(Exam) : void
+ viewGrades() : Map<Course,Integer>
<<Class>>
Teacher
- name : String
- email : String
+ generateReport() : Report
+ enroll(Student,Course) : void
+ deEnroll(Studen, Course) : void
+ register(Student, Exam) : void
+ createStudentProfile(Student) : void
+ deleteStudentProfile(Student) : void
+ updateStudentProfile(Student) : void
+ getStudentProfile(Student) : Profile
+ getStudentInfo(Studentt) : String
+ updateStudentInfo(Student) : void
+ deleteStudent(CNP) : void
<<Class>>
Course
- name : String
- startDate : Date
- endDate : Date
- teacher : Teacher
- students : LinkedList<Student>
- exam : Exam
+ enroll(Student) : void
+ deEnroll(Student) : void
+ viewGrades() : Map<Course,Integer>
uses
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<Class>>
Profile
- studentID : String
- group : String
- currentCourses : LinkedList<Course>
- grades : Map<Course,Integer>
+ enroll(Course) : void
+ register(Exam) : void
+ getGrades(): Map<Course,Integer>
<<Class>>
Report
- grades : LinkedList<Course,Integer>
- startDate : Date
- endDate : Date
+ print(): String
<<Class>>
Exam
- date : Date
- teacher : Teacher
- students : LinkedList<Student>
- course : Course
- room : String
+ enroll(Student) : void
+ deEnroll(Student)
+ viewGrades() : Ma
creately
www.creately.com • Online Diagramming

6. Data Model

The main data model that is used in the representation is the concept-oriented model. The model is very close in representation with the upward represented UML diagram, with the operations as stated in the specification: student enrollment in courses and exams, CRUD operations on the student information that the teacher can do.

7. System Testing

The strategy that is used in testing the application is JUnit testing.

8. Bibliography

- https://reqtest.com/requirements-blog/understanding-the-difference-between-functional-and-non-functional-requirements/
- https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html
- https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- https://en.wikipedia.org/wiki/Software_design_pattern
- https://en.wikipedia.org/wiki/Software_testing#Testing_methods