Student: Grigor Sonia Eufrosina Maria

**Group: 30233**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Design and implement a Java application for the management of students in the CS Department at Technical University of Cluj-Napoca. The application should have two types of users (student and teacher/administrator user) which have to provide a user-name and a password in order to use the application.

# Functional Requirements

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, enrollments, grades).
* Process class enrollment (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.

# Non-functional Requirements

Non-functional requirements specify the system’s quality characteristics’ or quality attributes. Some of the non-functional requirements are accessibility witch refers to help text to be in an international language, accuracy (date of birth to be in the past), performance (system responses should be no more than one second), efficiency (the system restart cycle must execute in less than one minute), safety(not causing harm, injury or damage).

2. Use-Case Model

Use case title: Student interaction with this system in order to log in.

Level: User-goal level which means something the actor is trying to get done. The properties of a user-goal level use-case model is a high-level interaction between the actor and the system, identifies user and system roles.

Primary actor: Student is the primary actor in this use case model diagram.

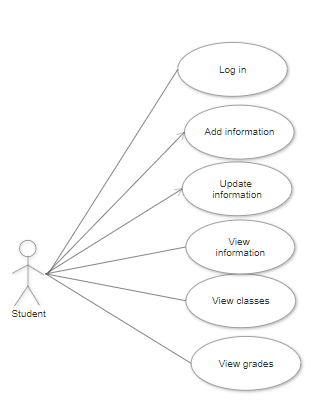
Main success scenario:

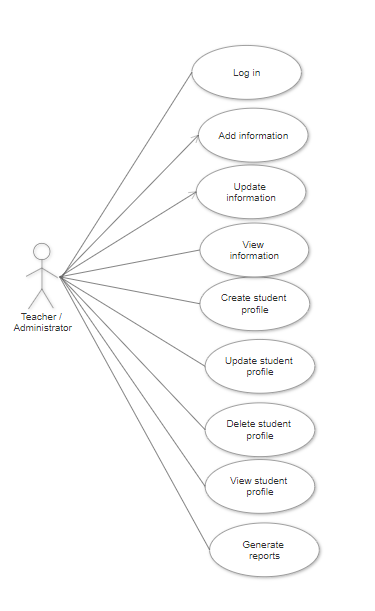
1. The student connects to the system.
2. The user enters his/her username and password.
3. The system validates the username and password.
4. The system determines the user’s role.
5. The system displays a list of actions the student can perform based on the user’s role.

Extensions:

3a. The system determines that the username does not match a user-name for any account.

1. The system displays an error message.





3. System Architectural Design

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. Architectural patterns are similar to software design pattern but have a broader scope.

**3.1 Architectural Pattern Description**

Layered pattern can be used to structure programs that can be decomposed into group of sub-tasks, each of which is a particular level of abstraction. Each layer provides services to the next high layer. The most commonly found four layers of a general application systems are as follows:

* Presentation Layer – User Interface Layer
* Application Layer – Service Layer
* Business Logic Layer – Domain Layer
* Data Access Layer – Persistence Layer

**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*

5. Class Design

**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*

6. Data Model

*[Present the data models used in the system’s implementation.]*

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

8. Bibliography

1. <https://searchsoftwarequality.techtarget.com/tip/Using-a-nonfunctional-requirements-template-plus-examples>
2. <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>
3. <http://kti.tugraz.at/staff/rkern/courses/sa/slides_ca.pdf>