1

Student: Rednic Ana

**Group: 30432**

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

This project consists of designing and implementing a Java application for the management of students in the CS Department at TUCN. The application has two types of users which have to provide a username and a password in order to use the application.

# Functional Requirements

The system should allow a normal user (student) to perform basic operations on his profile and to process class enrolment. The administrator user (teacher) should be able to perform CRUD operations on students’ information and to generate reports with activities of students.

# Non-functional Requirements

Performance: The system should have a minimal response time. Any operation should take less than 2 seconds.

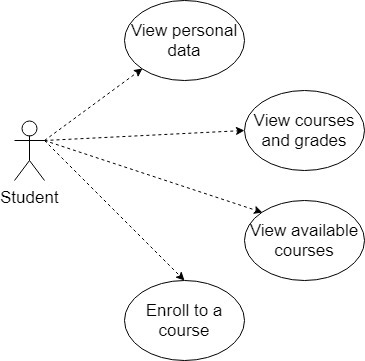
Testability: The application will be tested using JUnit tests.

Usability: The application will be user-friendly. It will have a GUI and all possible operations for a logged-in user will be displayed.

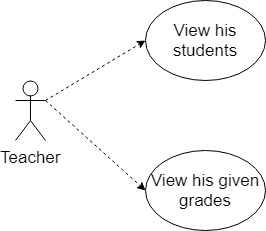
Security: The system is secured with log-in.

2. Use-Case Model

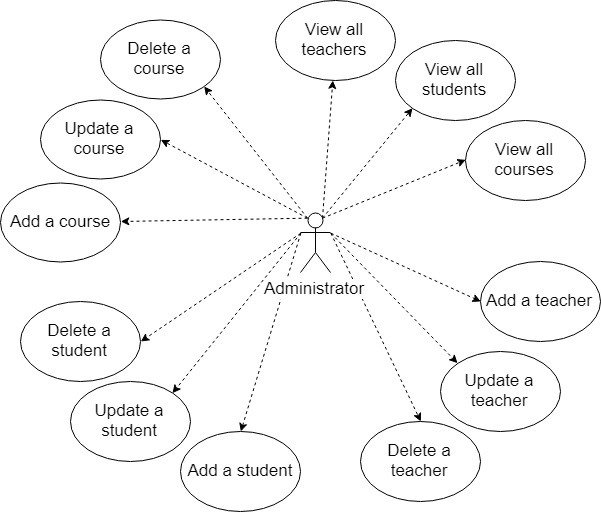
**Use case Diagram for Student**

****

**Use case Diagram for Teacher**

****

**Use case Diagram for Teacher as Administrator**

****

**Use case: Process Class Enrollment**

Primary actor: Student

Main success scenario: The student logs in the application, a student interface shows up, he selects to see all the available courses to which he is not already enrolled, inputs the ID of a certain course, and enrolls in it. If successful, the application returns to the rest of the available courses.

Extensions: The student might not be able to enroll to some of the courses because of some constraints.

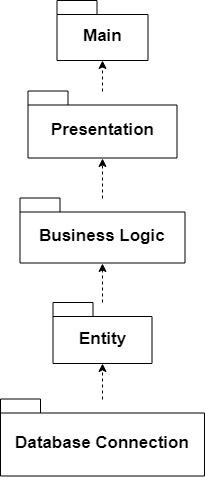
3. System Architectural Design

**3.1 Architectural Pattern Description**

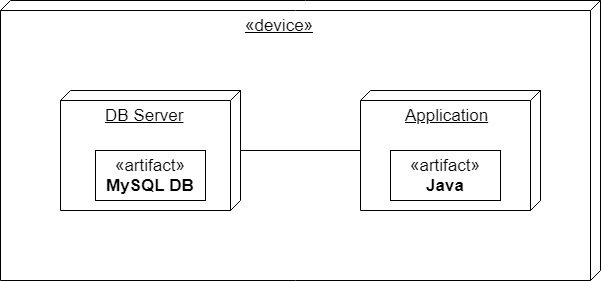
The Layers architectural pattern is used to design this application. Layers represent the logical separation of the application’s components into groups that represent distinct roles and functionalities. I will use three layers: Data Layer, Business Logic Layer, and Presentation Layer. The Data Layer is the part corresponding to a MySQL database that stores the application’s data. The Business Logic Layer is the part in which data is processed. The Presentation Layer is the part in which users interact with the application.

**3.2 Diagrams**

**Package Diagram**

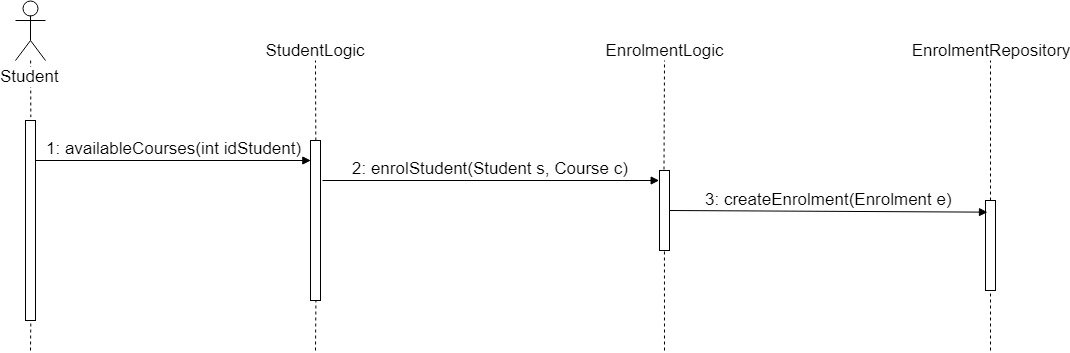
****

**Deployment Diagram**

****

4. UML Sequence Diagrams

**Sequence Diagram for Process Class Enrolment**

****

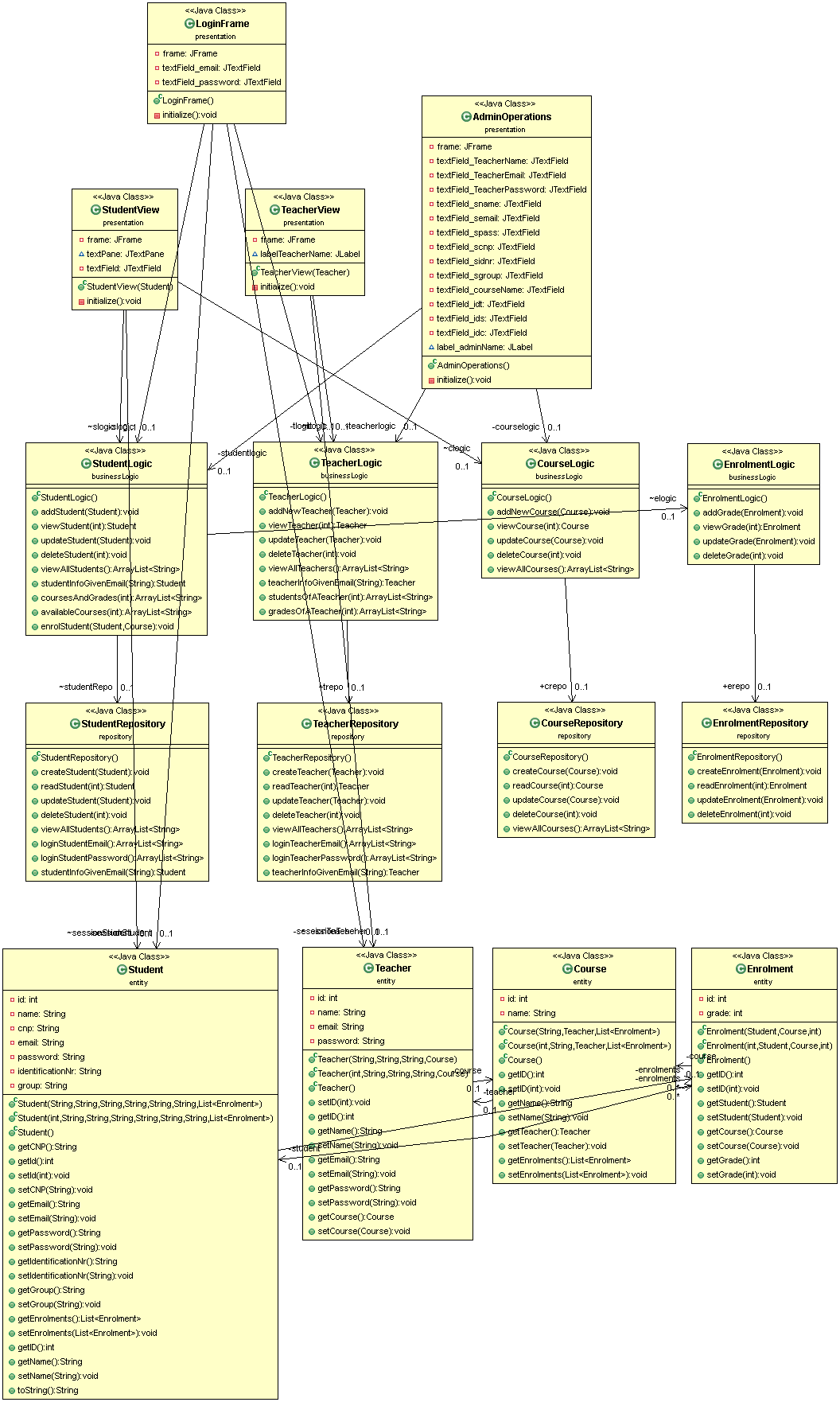
**5. Class Design**

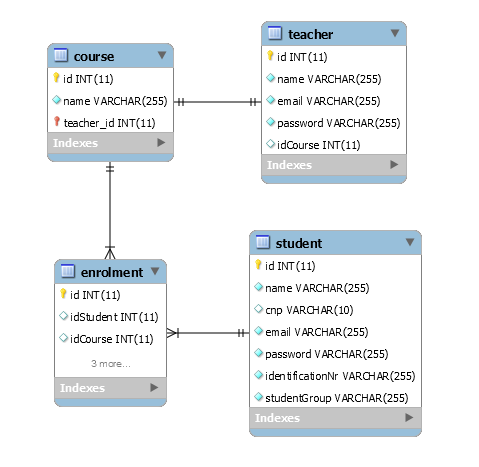
**5.1 Design Patterns Description**

Table Data Gateway (for Data Layer) – An object acts as Gateway to a database table. One instance handles all the rows in the table. A Table Data Gateway holds all the SQL for accessing a single table or view: selects, inserts, updates, and deletes. Other code calls its methods for all interaction with the database.

Table Module (for Business Logic) – There is a single instance that handles the business logic for all rows in a database table or view.

**5.2 UML Class Diagram**

**6. Data Model**



7. System Testing

Unit testing will be performed on each component of the system, validating it. The focus falls on the tests that impact the behavior of the system. This will be done using Dataflow testing.

In the Integration testing part, individual units will be combined and tested as a group. This will look forward to expose defects in the interfaces and in the interaction between integrated components.

The System testing part will verify if the whole application meets the specified requirements.

Validation testing will be performed to decide whether or not the application is ready to be made available to the end-users. The last three stages will be done using Black Box testing.

8. Bibliography

<https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

<https://msdn.microsoft.com/en-us/library/ee658109.aspx>

<https://www.codeproject.com/Articles/36847/Three-Layer-Architecture-in-C-NET>

<https://martinfowler.com/eaaCatalog/>

<http://softwaretestingfundamentals.com/>

<https://www.tutorialspoint.com/software_testing_dictionary/data_flow_testing.htm>