Assignment 1

Analysis and Design Document

Student: Boros Hanniel
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implement a Java application for the management of students in the CS Department at TUCN. The application should have two types of users (students and teacher/administrator user) which have to provide a username and a password in order to use the application.

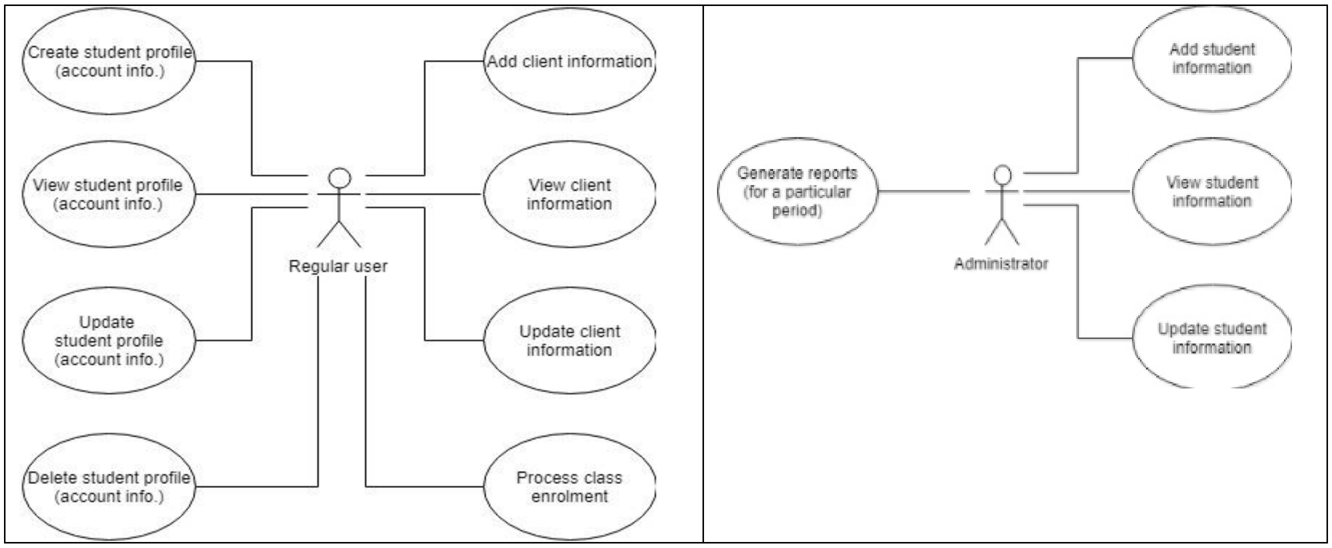
1.2 Functional Requirements

- Authentication
- Edit client information
- Edit profile
- Process class enrolment
- Generate reports

1.3 Non-functional Requirements

- Usability
- Performance (response time < 1s)
- Security
- Data integrity
- Documentation

2. Use-Case Model


Create student profile (account info.)
View student profile (account info.)
Update student profile (account info.)
Delete student profile (account info.)
Regular user
Add client information
View client information
Update client information
Process class enrolment
Generate reports (for a particular period)
Administrator
Add student information
View student information
Update student information

One use case description

Use case: Update client information

Level: sub-function

Primary actor: student <a role name for the actor who initiates the use case>

Main success scenario: <the steps of the main success scenario from trigger to goal delivery>

Student selects Update Personal Information option from menu

Student updates the necessary information (name, identity card number, personal numerical code, address)

Student presses Update button, finishing update.
Student waits a short time while information is updated in the database.
Information is updated and student is notified.
Extensions:
Alternate scenario for failure:
if connection to database fails for some particular reasons, or an Exception is encountered, the information may not be updated.

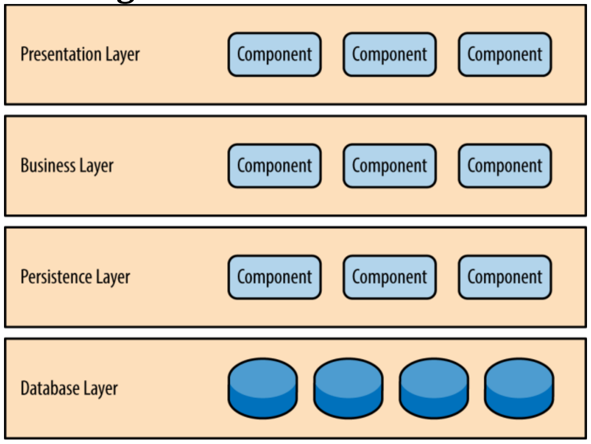
3. System Architectural Design

3.1 Architectural Pattern Description

Layers Architectural Pattern
The architectural pattern used in this project is the Layers architecture, otherwise known as the n-tier architecture pattern. This architectural pattern helps us to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

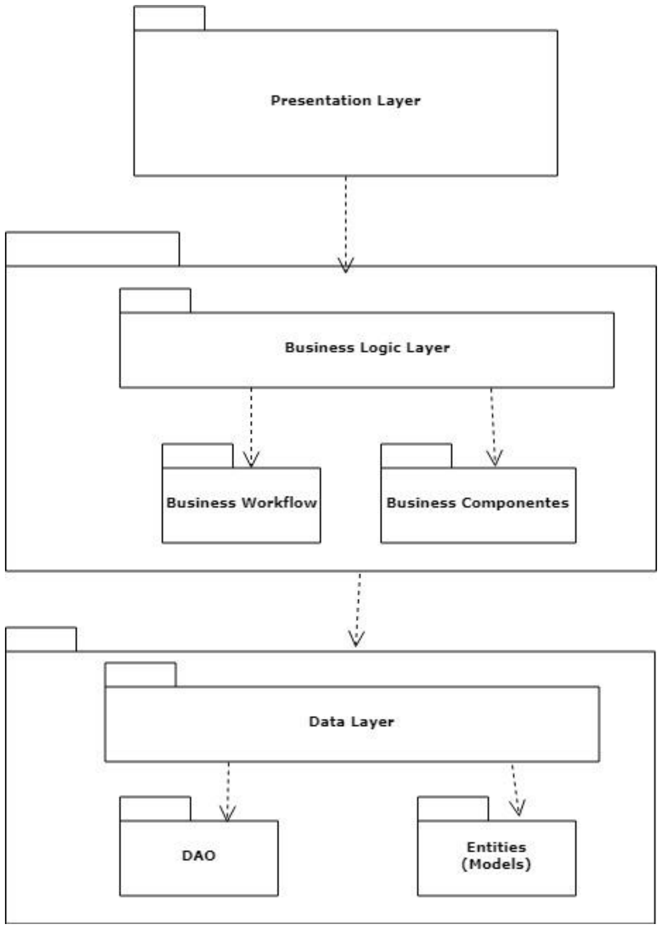
Presentation layer
This layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.
Business layer
This layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use.
Data layer
This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

3.2 Diagrams

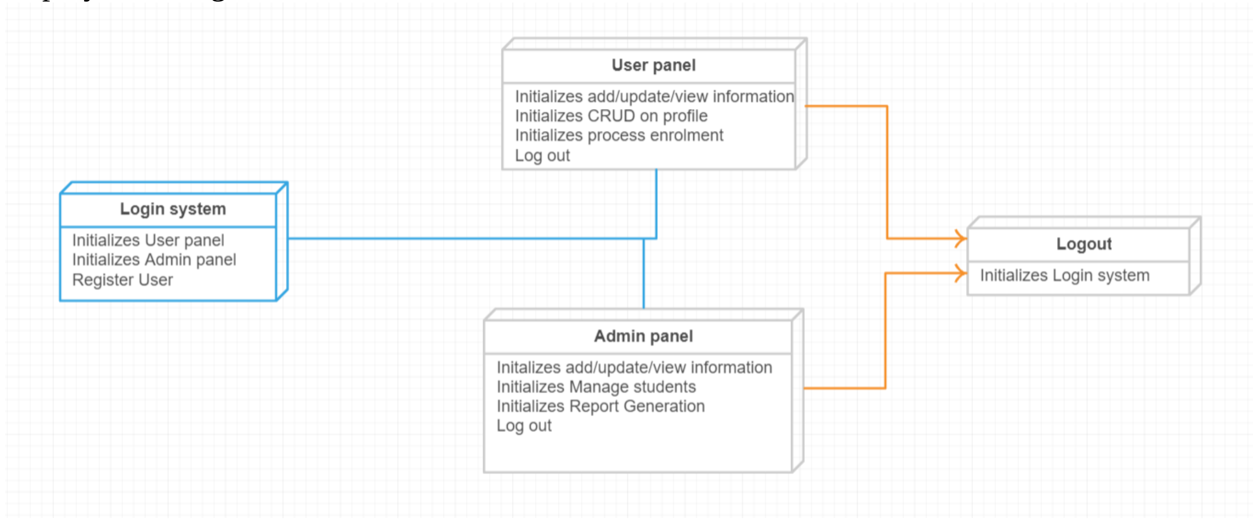


Layered Architecture Pattern

Package diagram

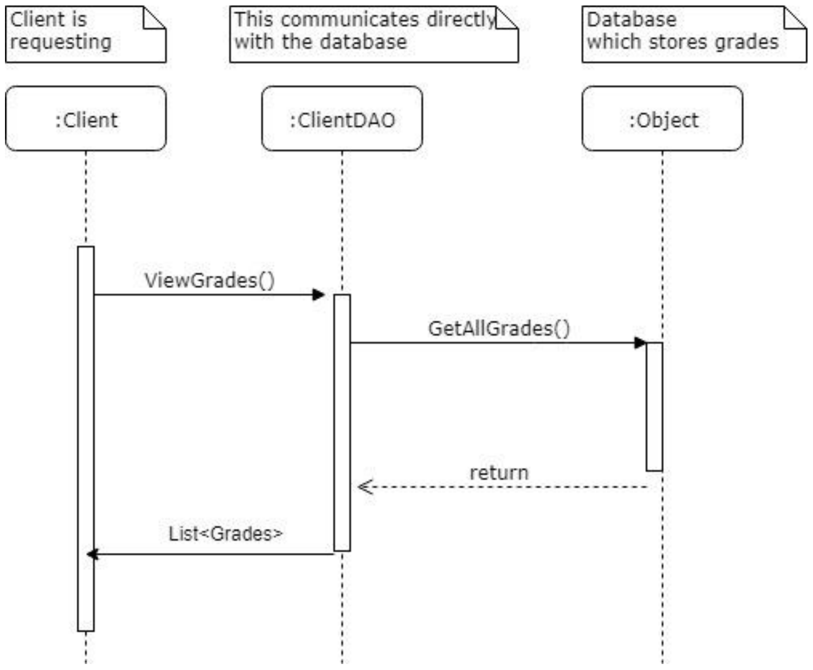


Deployment diagrams


User panel
Initializes add/update/view information
Initializes CRUD on profile
Initializes process enrolment
Log out
Login system
Initializes User panel
Initializes Admin panel
Register User
Logout
Initializes Login system
Admin panel
Initalizes add/update/view information
Initializes Manage students
Initializes Report Generation
Log out

4. UML Sequence Diagrams

Sequence Diagram for Viewing Grades (for a student)



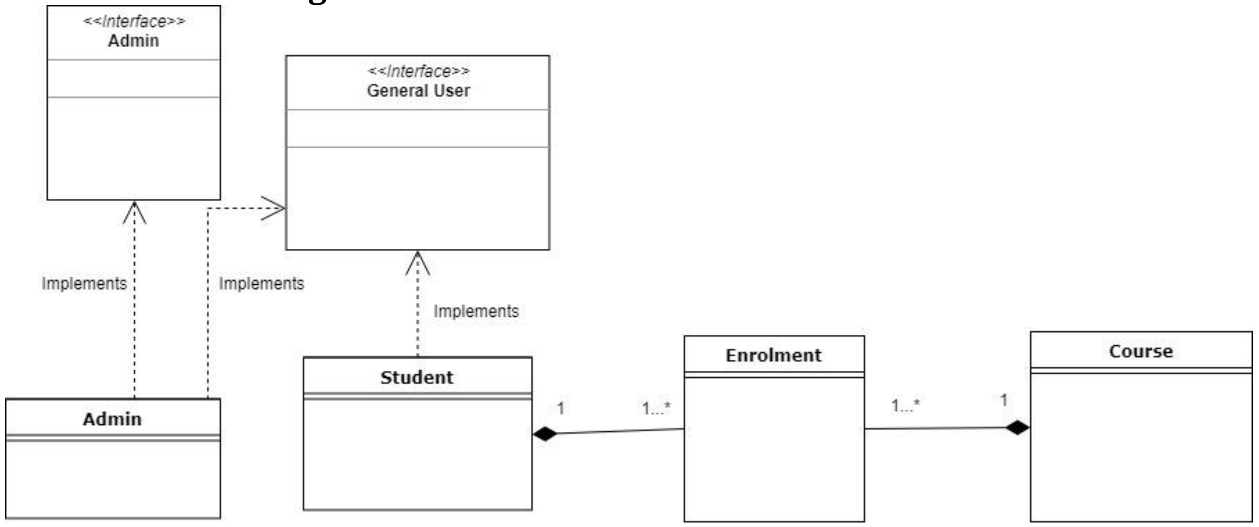
5. Class Design

5.1 Design Patterns Description

Singleton Creational Design Pattern

This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.
This pattern was used for the design of the Connection Factory class which is responsible for connection to the database.

5.2 UML Class Diagram



6. Data Model

The main data models for the user of the systems are the following: Student and Administrator.
I considered the best decision to have a main (abstract class) GeneralUser which has the most basic common functionalities of the system. The Student model and the Administrator model will be a sub model of this basic model.
The two models were implemented based on the SOLID design principles. I focused to have the required functionalities implemented, not overcomplicating the implementation and reasoning.
The Course class is a simple class. There is also an enrolment class which is responsible for the connection/relationship between the student and the courses they are enrolled to.
The DAO classes are responsible for the connection to the database and for data access.
Other classes represent the functionality of the system.
I consider not relevant here the description of the GUI model.

7. System Testing

I will use unit testing (JUnit 4) in order to test my application.
I tried to make an implementation which will be favorable and sustain testing.
Assertions were used in order to verify if the desires functionality is working.
Example of testing if a client/user successfully edited his profile:

8. Bibliography

Here are mentioned some resources which were helpful for writing this documentation and as well gave me a better understanding of the concepts and guided me in the implementation process:

Information and knowledge:
Microsoft Application Architecture Guide, 2nd Edition
Software Architecture Patterns, by Mark Richards
https://www.tutorialspoint.com
https://en.wikipedia.org

Tools helpful for the documentation:
https://www.smartdraw.com
https://creately.com
https://www.draw.io/