CS308 Assignment

1- Description:

This coursework is to design and implement a university student management system (USMS) to help maintaining the students records and to maintain the teaching materials. The proposed university has departments. Each department offers courses, and each course is made up of several modules in each semester. Some courses are undergraduate courses while others are postgraduate courses. An academic year has two semesters, and each semester has an exam period. Students' marks (Lab and exam) are entered to the system. Thus, the system maintains student's exam record. For example, it is possible to check all the marks achieved by a given students in each module (i.e., to see all attempts). A student may resit a module many times (attempts) according to a business rule defined previously. Such a business rule defines the maximum number of allowed attempts. For example, to set the max allowed attempts for a given module, or for a given set of modules, or for all course's modules. The proposed system has different types of users, and each user has different tasks. Also, each user has a manager (managed by other user). The types of users are managers, lecturers, and students. The students and lecturers are managed by a manager. Managers are managed by managers as well, however, a manager cannot manage themselves. Each user type has different tasks as the following:

- **Student** can
    - signup by entering their details (first name, surname, gender, email, DOB),
    - login
    - update their password,
    - view their enrolled course and their modules together with their marks (a mark that is >=50 is considered Pass and Fail otherwise)
    - download a lecture note and a lab note for a given module in a given week.
    - view their decision if it is available:
        - **award** decision is given when all modules in their enrolled course has a Pass mark, and
        - **resit** decision is given otherwise. However, a
        - **withdraw** decision is given when all attempts allowed for any module in their course have been attempted)
- **Lecturer** can
    - signup by entering their details (first name, surname, gender, email, DOB, qualification [PhD, MSc, etc, to be picked up from a list of qualifications]).
    - login
    - update password
    - update module information
    - upload or update module materials in each week: lecture's note and Lab's note
    - display enrolled students in their module
    - update the exam record of every enrolled students: assign Lab's mark and assign exam mark to each student enrolled in the module.
- **Manager** can
    - login
    - update their password
    - view sign up workflow: a workflow that has users who signed up and awaiting approval
    - approve users who signed up to create their accounts.
    - approve lecturers who signed up to create their account
    - manage other users accounts (activate, deactivate, password reset)
    - assign a module to a given lecturer
    - enrol student in a given course
    - issue student decision (award, resit, withdraw)
    - add a business rule such as:
        - the maximum number of attempts for each module that a student can try

        - number of modules that are allowed to be compensated (when a mark is in the range of 40 to 49 inclusive)
    - add new course (code, name, description)
    - add module (code, name, credit)
    - assign module to a course so that this course will include this module
    - display course details (semesters and modules)
    - display module details (name, code, credit)
    - update course information (description). Note that some modules have different credits such as a project module so the systems must allow different credits assignments.

It is quite acceptable that you decide to enhance the current requirements by certain assumptions based on your discussions as a team. In such a case, these new assumptions must be stated when writing the report that documents your work. (A template of this report is provided on the module's webpage on Myplace)

The proposed system comprises GUI(s) (front-end) and back-end. You have to design and implement both GUI(s) and the necessary code: classes, interfaces, algorithms that form the back-end. Your GUI interfaces should be clear, well designed and **easy to use**. **Easy to use can refer to the time of training that a user requires to be able to use your system with minimal errors**. Also, you will work with **relational database system namely MySQL**. you have to design, model and implement the entity relationship diagram. This E/R model will represent the data model that supports the business tasks required from this system.

2- <u>Submission Requirements:</u>

The submission consists of **Three files** (please replace all **xx** occurrences in file names with your group number)

1- **group_xx_.pdf**: a file that contains your documentation of this project according to a template file **group_xx_.doc** that is available to download from the module's webpage on Myplace. To write this file, please follow the instructions that are stated inside this template file.
2- **All_xx_.zip:** a zip file that contains the following files:
    - **2.1. Project_xx_.zip**: this is your <u>IntelliJ project folder</u> zipped: this folder is your project folder and should include all source code files (.java) and the **test folder** that has the test suite(s) in JUnit5. **Please note** that your project folder should be complete e.g., I should be able to open/edit your project and compile it on my own PC. **You have to compress (zip) your IntelliJ project folder** when you have the IDE closed.
    - **2.2. Db.sql**: this file is the backup of your MySQL database (you can automatically generate this file by using the option **export** and the **format** SQL available in phpMyAdmin.
    - **2.3. DbConnect.txt:** This file is a text file that has the connection parameters to MySQL server. Please make sure that the structure of this file is similar to the example DbConnect.txt available to download from the module's webpage on Myplace
    - **2.4. App_xx.jar:** this is the executable of your project that can run on any computer. Please make sure that your **app_xx.jar** can run on any machine (try it on another PC)
    - **2.5. ERD.tiff**: this is the image file of your E/R model that can be zoomed in/out.
    - **2.6. SUCDrgm.tiff**: this is the image file of your <u>system use case diagram (SUCD)</u>
    - **2.7. ClassDgrm.tiff**: this is the image file of your <u>class diagram</u> that can be zoomed in /out.
3- **vedio_xx_.mp4**: a video file recording for Max of 15 minutes that shows your system at work For example, to use your system as a manager, then as a lecture, then as a student. This video file should show how the identified user requirements are met (please make sure that your file is playable and has a sound).

Please note that App_xx.jar is the executable of your project that can run on any machine (try to run it on a PC that is different from the one that you have used to create this JAR file). This JAR file requires the database connection settings that can be read from the DbConnect.txt file. The DbConnect.txt file has only four lines that contains: host, the database name, username and password. Your DbConnect file must be similar to the template DbConnect.txt file that is available to download from the module's webpage on Myplace.