**LMS Report**

[**https://github.com/rherrera5/CS3321LMSProject**](https://github.com/rherrera5/CS3321LMSProject)

*I was unable to attend class the day teams were being decided, and could not find one after. Subsequently, I had to complete this project alone. I did, however, learn about project based programming projects after considering how I would have completed each step if I had a team.*

Team Members: Robert Herrera

**Team Model and Development Life Cycle:**

The team model chosen for this project was the *chief programmer team* model. Since there is only one programmer, this seemed like the natural choice. The *rapid prototyping lifecycle* was the life cycle model chosen as it allows for a quick and efficient analysis of requirements followed by the development of the actual system once requirements are understood.

**Project Summary:**

Making a Learning Management System. The project is going to keep track of various information about students one semester at a time. There will be limited algorithms necessary, it will be mostly just a data output program. The program will work as an offline application and users will be able to log in provided they are students or instructors in the same database outputting the information stated above.

**Requirements:**

A database that will track information on 2 levels.

People:

-Their Names

-ID's within the school system.

-Courses taken

-each exam's score in each course.

Courses:

-Teacher name

-CourseID

-Course Name

A GUI that makes user navigation easy

1 screen for logging in. This screen should verify the credentials and forward the user to the appropriate screen.

The next screen will show all personal information for the user or options for admin to add or remove students depending on whose credentials are entered.

The application will have to talk with a SQL database and run queries with some user information inserted into the WHERE clause.

User should be able to see personal information, their grades, all available courses, and calculate their GPA.

Admin should be able to add a user, delete a user, add a course, and delete a course.

**Analysis:**

The following is the analysis of the requirements given previously

Project will need a GUI login and display screen containing the following:

-Username field

-Password field

-Login Button

-Close/Cancel Button

Login button will put you at the second GUI screen depending on job of user logging in

-All personal data about the student

-Full name

-Student ID

-GPA Calculation

-Will have buttons for the following:

-Log out

-A button labelled 'course info' that will move user to different GUI window with the following information:

-Course Names

-Course ID

-Professor Name

Or an admin GUI which will allow for the functions listed below

There will need to be a database with the following tables:

-Person Table with the following columns:

ID(PK)

Name

Job(Student/Administrator)

Course ID(FK)

-Course Table with the following columns:

Course ID(PK)

Course Name

Professor

Student ID(FK)

-Grades

TestID(PK)

Grade

CourseID(FK)

The database connector class will have the following functions to pass to the various GUI windows:

-deleteStudent

-insertStudent

-getGrades

-calculateGPA

-getJob (for determining which GUI user is sent to)

-getConnection

The only algorithm to be used will be the GPA calculation it will be tied to the module that runs the algorithm.

A module to connect to the database will be necessary and will contain functions to create the connection, another function to check for name comparison and after that has been passed, a function will be passed to see if the password matches.

Another will be necessary to deal with the login process. It will read the input at the front GUI: username and password then be passed to the above module for checking in the database.

Another will be needed to bring up the second screen, which will show full name, ID and the calculated GPA. There will be a button that puts you to the 3rd and final GUI window showing courseNames courseID and professorName. All GUI's beyond the login screen will have a logout button.

**Updated “Quality of Life” Analysis from second iteration:**

Make studentLogin class have an overloaded constructor so extraneous step of username input can be skipped.

On student login class, have table display 4 different result sets based on 4 reworked buttons.

-One button for Student Info, which is shown by default

-One button for all course Info

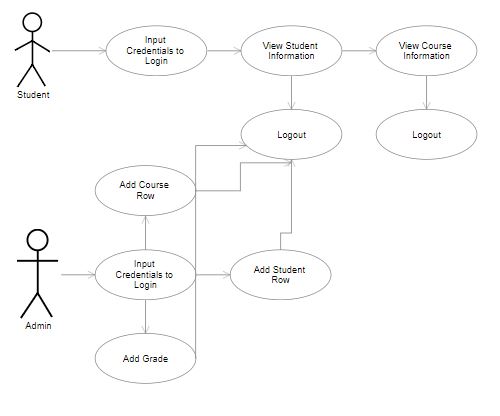
-One button to show course in which student is enrolled

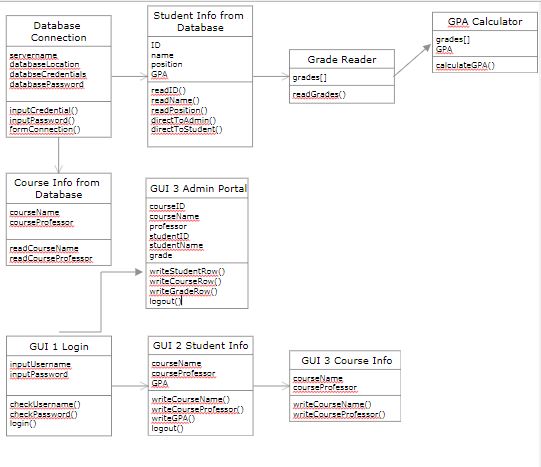
-One button that will calculate the GPA

Create a separate class called Math, which will receive an int array parameter and process the average grade.

In all tables with textboxes, have them auto highlight all text on left click.

**Design:**



****

**Implementation:**

Source Code attached.

**Test Cases:**

Test cases used are commented out in the Connector class below the functions.

Testing the whole application with the following:

At Login:

use known credentials User and Pass

use unknown user and pass

use known user but incorrect pass

use unknown user but known pass

use int

use negatives

use null

-Everything functions as specified in design and analysis-

At Student data:

All buttons work as proposed, test cases are commented out in studentLogin file.

Still need to find a way to bring txtUsername variable from Login class so user doesn't have to re-input username. Can currently access records for all students.

At AdminLogin

All buttons work as proposed test cases commented out. Need to find a method to highlight all text on click focus. Quality of life issue.