Project

CS 510/410 - Databases

Due Wednesday, December 10, 2025 at 11:59 p.m. This project will be graded out of 100 points, and will be worth 15% of your grade.

Undergraduate students must work on this project with a partner (only submit one copy). **Student teams are listed** here. **Graduate students will work solo.**

Summary

In this project, you will design and implement a Java application for managing grades in a class. This needs to be a command shell application.

Problem Setup

- We need to be able to track grades for multiple classes. Each class has a course number (e.g. CS410), term (e.g. Sp23), a section number, and a description, along with other fields you think might be relevant.
- Each class has multiple categories (e.g., homework, exam, project, etc.), each of which has a name. We can assign a class a weight for a category.
- The class has multiple assignments, each of which also has a name, description, and point value, and is in one category. No two assignments in the same class can have the same name.
- We can have multiple students, each of which has a username (i.e. the first part of their e-mail address), a student ID, and a name. Each student can be enrolled in one or more classes.
- We can assign a student a grade for an assignment.

Part 1: Data Model

Draw an E-R model for the data needed for this problem.

Part 2: MySQL Schema

Write SQL CREATE TABLE statements to implement your model. Include foreign key relationships and suitable indexes.

Part 3: Java Program

Write a Java command shell program to implement the application. You need to implement the following commands, along with others you deem necessary or helpful:

Class Management

- Create a class: `new-class CS410 Sp20 1 "Databases"`
- List classes, with the # of students in each: `list-classes`
- Activate a class:
    - `select-class CS410` selects the only section of CS410 in the most recent term, if there is only one such section; if there are multiple sections it fails.
    - `select-class CS410 Sp20` selects the only section of CS410 in Fall 2018; if there are multiple such sections, it fails.
    - `select-class CS410 Sp20 1` selects a specific section
- `show-class` shows the currently-active class

All other commands are to be interpreted in the context of the *currently active class*. This is like your current directory in a Unix command line. You can use a field in your shell class to track the currently-active class.

Category and Assignment Management

- `show-categories` – list the categories with their weights
- `add-category Name weight` – add a new category
- `show-assignment` – list the assignments with their point values, grouped by category
- `add- assignment name Category Description points` – add a new assignment

Student Management

- `add-student username studentid Last First` — adds a student and enrolls them in the current class. If the student already exists, enroll them in the class; if the name provided does not match their stored name, update the name but print a warning that the name is being changed.
- `add-student username` — enrolls an already-existing student in the current class. If the specified student does not exist, report an error.
- `show-students` – show all students in the current class
- `show-students string` – show all students with ‘string’ in their name or username (case-insensitive)

- `grade assignmentname username grade` – assign the grade 'grade' for student with user name 'username' for assignment 'assignmentname'. If the student already has a grade for that assignment, replace it. If the number of points exceeds the number of points configured for the assignment, print a warning (showing the number of points configured).

Grade Reporting

- `student-grades username` – show student's current grade: all assignments, visually grouped by category, with the student's grade (if they have one). Show subtotals for each category, along with the overall grade in the class.
- `gradebook` – show the current class's gradebook: students (username, student ID, and name), along with their total grades in the class.

Grade Calculation

For reporting grades, calculate grades out of 100. Rescale category weights so they sum to 100; within each category, compute the fraction of possible points a student has achieved (divide their total grade points in that category by the total possible points based on assignment point counts).

For both `student-grades` and `gradebook`, report grades two ways: a *total* grade, based on total possible points (including assignments for which the student does not have a grade at all), and an *attempted* grade, that is based on the point values of the assignments for which they have a grade.

You should do as much of the grade computations as possible in SQL – minimize your Java computations. Student grades and the gradebook should each be computed with a single query if possible.

Submission

Submit a zip file containing the following files:

- `model.pdf`, a diagram of your data model in E-R syntax.
- `schema.sql`, an SQL script containing your DDL (database schema).
- `dump.sql`, a MySQL dump file containing a dump of your database with example data. I encourage you to include our class, properly configured, with dummy grades, as one of the example classes.
- A zip file of your source code (add a comment before each of your methods)
- A README file describing your implementation.
- **The link to a 5 minutes video where you showcase your project.**
- (Undergraduate only) **One completed copy of the PeerEval-GroupWork-form for each student in the group.**

Submit your project to **Canvas** (**only one submission per group**).

RESOURCES: MySQL Sandbox local guide (pay attention to Sections 3.4 and 3.4.1 which are about Using JDBC to access MySQL from Java programs and also provide an example).