



**EZGrader Teaching Assistant  
Object-Oriented Design  
Design Document**

**CS591: Object Oriented Design and Development in Java  
Group #13**

Zexu Chai, Kenneth Jusino, Xiangyu Zhang  
Boston University   chaizexu@bu.edu, kennethj@bu.edu, zxyzhang@bu.edu

**Boston University College of Arts and Sciences**

# Contents

<b>What is this Product .....</b>	<b>3</b>
<b>Why this Product.....</b>	<b>3</b>
<b>User Document .....</b>	<b>4</b>
Preparation.....	4
Instructions .....	5
How to use the product.....	6
<b>Architecture Design Document.....</b>	<b>13</b>
<i>Summary .....</i>	<i>13</i>
Object Model and Implements .....	13
Object and Class Diagram.....	14
Database .....	15
<i>Database Model .....</i>	<i>15</i>
<i>The E-R Diagram .....</i>	<i>15</i>
<i>The relational schema .....</i>	<i>16</i>
Connection between Database and Objects .....	16
<i>Tables and their relation to objects .....</i>	<i>16</i>
<i>The interface between MySQL database and other parts .....</i>	<i>17</i>
<b>What is Next .....</b>	<b>18</b>

## What is this Product:

The EZgrader is a state of the art grading system, emphasizing the needs of a teaching professional, while minimizing the complexity of modern-day organizational tools like Excel. With our product, teachers and professors alike have the ability to create courses with assignments that weigh themselves automatically, use a class statistics generator with a click of a button, and be able to pull up information about any student by merely selecting their name. With the EZgrader, the everyday nuisances that a teaching professional has with their current grading system become nil.

## Why this Product:

- **Easier to use than Excel:** With this better visualization grading system, you can just focus on the data you have and the operations you want to make. There will be no need to worry about all calculations, data storage and mistakes you can make by accident.
- **User-friendly UI:** You can be fully engaged in this application after installation without any instructions due to its very easy and convenient design. You can follow the project flow and complete everything in this procedure.
- **Better Security:** You own your private account and without your username and password, no one else can get access to your system. All of the courses and students information is protected from all other people.
- **Fully Functional Product:** It is a great chance for you to try our product with multiple functions provided. Create classes using existing classes, change weights automatically, curve an assignment and everything else you can think of in a grading system!
- **Calculators Not Needed:** Every function is completely automated and the modern day professor will never have to pick up a calculator again to compute student scores. Just input points lost, maximum score, and press a button.

# User Document:

## Preparation:

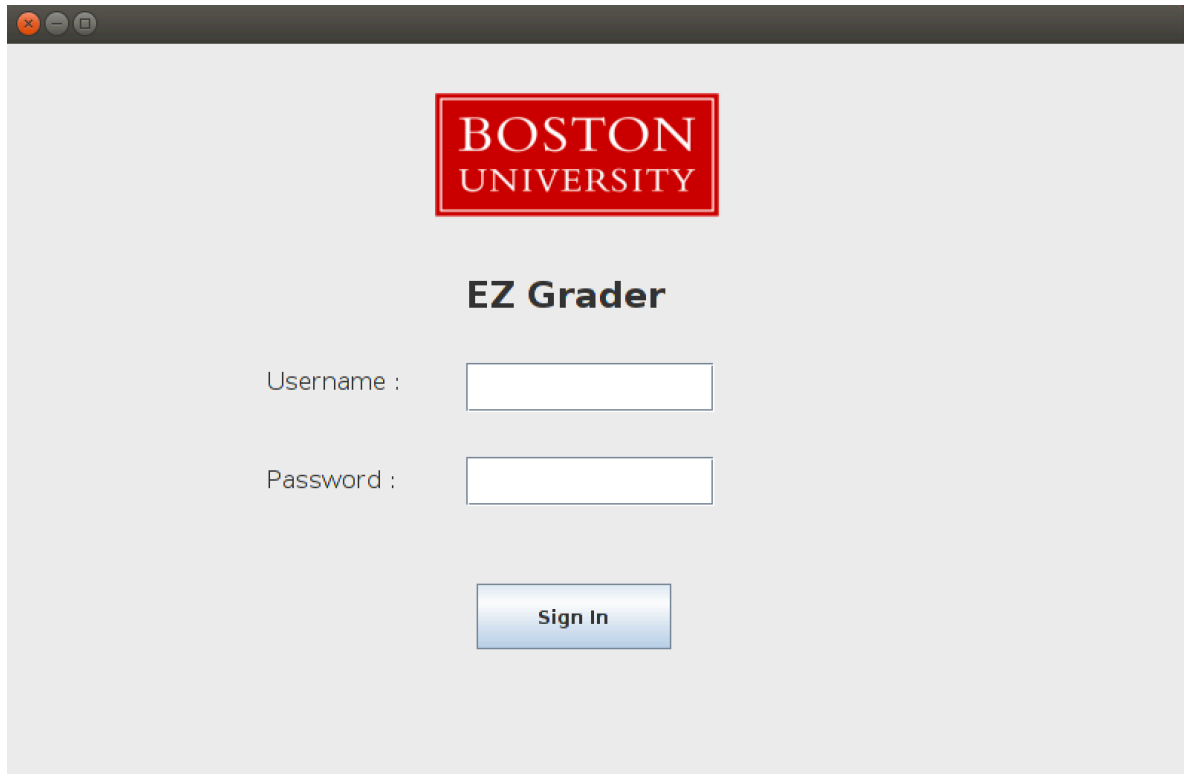
- IDE: We ran our application on Eclipse. So in order to run it successfully, you need to download the newest version of eclipse here:  
<https://www.eclipse.org/downloads/>
- Database: We used MySQL database to store the information of students and courses details. Download the MySQL database from website here:  
<https://dev.mysql.com/downloads/mysql/>  
Go to the bottom of this page and download the version that matches your computer.
- MySQL Java connector: You'll need to add mysql java connector as an external library. Download the connector from here:  
<https://dev.mysql.com/downloads/connector/j/5.1.html>  
Add the connector using Build path->Configure build path -> Libraries -> Add external JARs in eclipse.
- Relational Schema: Before using EZGrader, you must run the relation schema file "schema.sql" with your MySQL server to create the tables. Use command `mysql -u username -p ezgrader < schema.sql` in your MySQL console.
- Change database credentials: You need to change the username and password in "Database.java" to your own credentials.

## Instructions:

### First Step into the project:

Login(please run the file **Login in package gui to start the product**) :

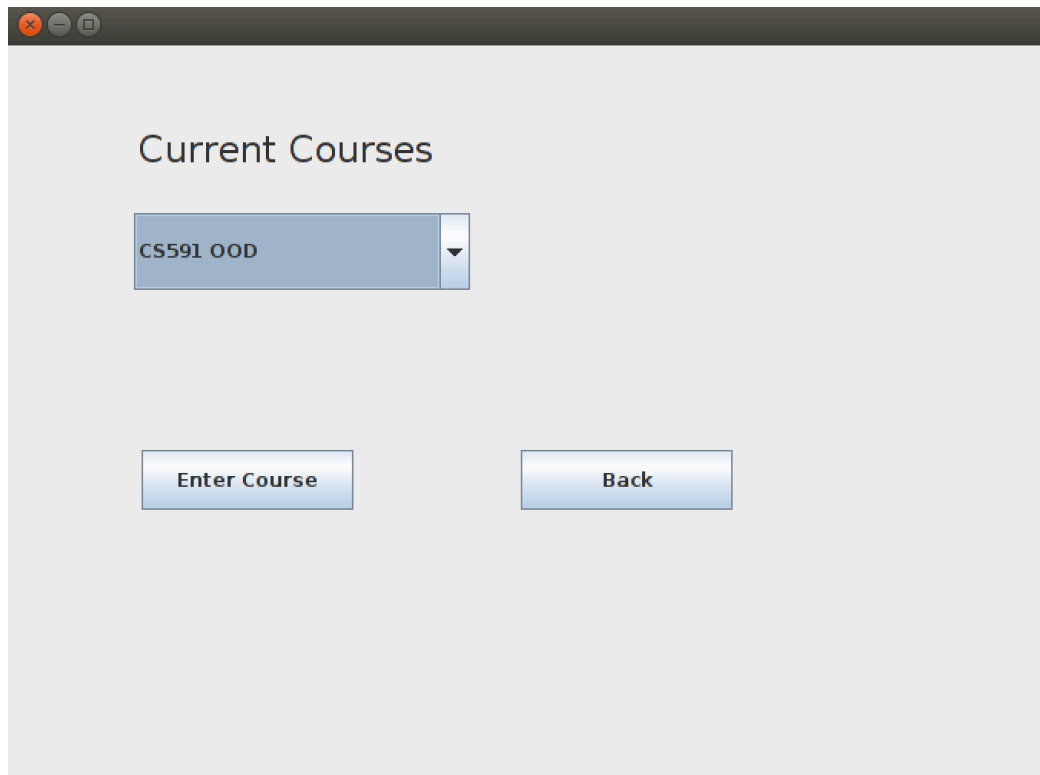
- ☐ Our program starts with a login page where there is only one user and only they can sign into the program.
- ☐ The professor will have to input their **username** (“cpk”) and the **password** (“cs591”).
- ☐ After they sign in, the user will be brought to a page where they are able to open an existing course, create a new course, or logout.

A screenshot of a web application window titled "EZ Grader". At the top center is the Boston University logo, which consists of a red rectangle with the words "BOSTON UNIVERSITY" in white. Below the logo, the text "EZ Grader" is displayed in a bold, black font. Underneath, there are two input fields: the first is labeled "Username :" and the second is labeled "Password :". Both fields are empty. Below the password field is a blue button with the text "Sign In" in white. The entire form is centered on a light gray background.

### Three options after login:

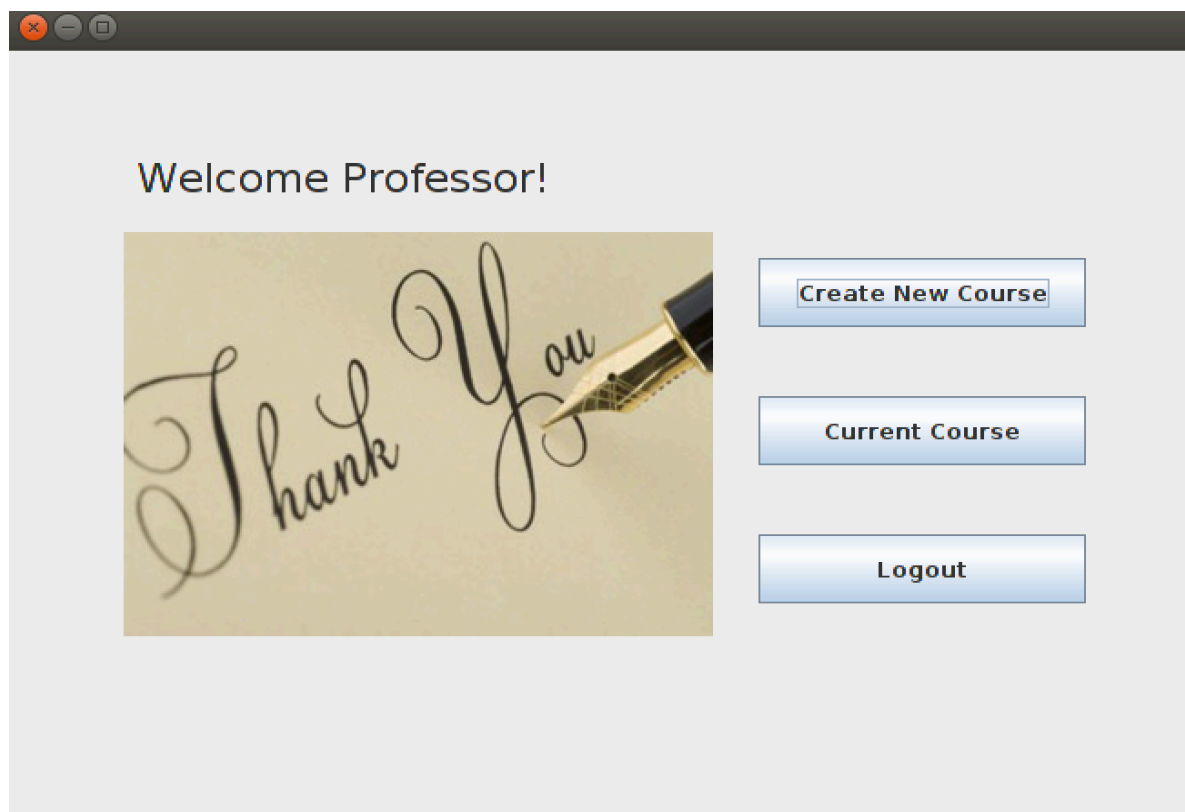
#### 1. Current Course:

- ☐ Assuming the professor would like to open an existing course, the professor will click on the “open existing course” button and be brought to a page with a drop down menu of courses that have already been created and are saved in the database.
- ☐ Once the professor chooses the course they are interested in, the main course page will open and all of the courses’ information will be available, including students in course, assignments, past grades, comments made, and more.

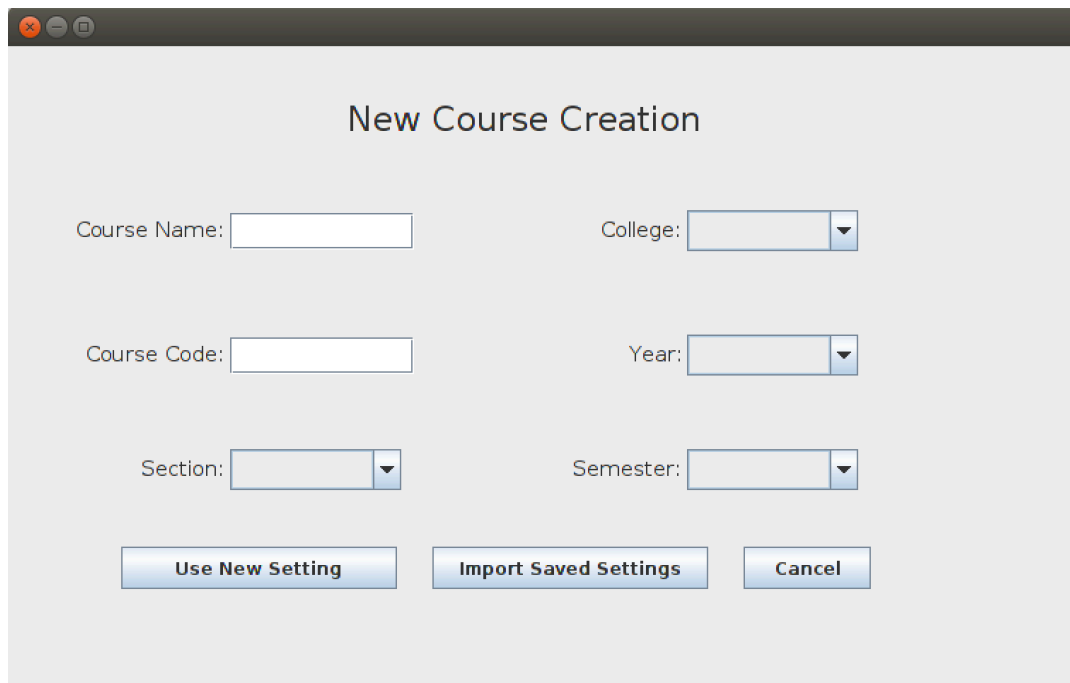


## 2. Create Course:

- ❑ If the professor would like to add a new course for a semester, click on “create new course”.



- ❑ The page that will come up will have 6 components asking for course information, including the name of the course, the course id number, the section, college, year, and semester.



New Course Creation

Course Name:  College:

Course Code:  Year:

Section:  Semester:

- ❑ Once that information has been inputted, the professor will have to choose between using existing course settings or setting brand new ones.
  1. **Importing Existing Course Settings-** the professor is able to select from previous course settings and can initialize the new course to have the same weights and assignments as the existing course, without adding the students and their grades. Once the professor chooses an existing course setting, they will be directed to add students to the course.
  2. **Initializing New Course Settings-** the assignment categories and their weights will be initialized, along with the max score of the first assignment of each of those types (i.e. homework 1 under the Homework category). The page will have radio buttons of the different types of assignments that we could think about, like homework, quizzes, exams, finals, projects, and other. Once the weights of the course categories and the first assignments score have been initialized, the professor will add students to the course.
- ❑ Students can be added to new courses by selecting csv files that follow a strict format. The csv file could come from excel or a regular text editor. The format of the student information should be in the following order: FirstName, MiddleInitial, LastName, BUID number, Major, College, GPA, UnderGrad/Graduate (**or you can use the csv file we provide in the folder called students.csv**).

- ❑ Once this file is ready to be imported, continue using EZGrader. Once the “add students” button is pressed, a new frame will appear with a text-field and a “locate file” button will appear.
- ❑ The locate file button will allow the user to search their computer’s files and select the file.
- ❑ Once the file is selected, the empty text-field will be occupied with the files address so the user is sure that is the right file.
- ❑ Once the correct file is selected, a new button will appear that will add students to the course. Once the button is pressed, the professor will be redirected to the Course Detail page.

3. Logout: You can click logout button to redirect to the login page.

## Finally step into the main course page

### Course Detail:

This is the main page where a professor will be able to interact with the course she is teaching. The page has tabs on the top left-hand corner that allow a professor to quickly **toggle between courses** and give easy access to all of the courses’ information. Below are a multitude of buttons that have a variety of features.

Toggle between course tabs

**Course Settings**

Buttons: Add Component, Delete Component, Curve, Change Weight

**Summarize Data**

Buttons: Calculate Final, Print Statistics

**Command**

Buttons: Back, Logout

**EzGrader** **BOSTON UNIVERSITY**

**Add/drop student from course**

BUID	Name	Type	Homework (G:20.0%/UG:20.0%)			Quizzes (G:40.0%/UG:40.0%)			Projects (G:40.0%/UG:40.0%)			Cumulative
			Homework 1 (G:100.0%/UG:100.0%)			Quizzes 1 (G:100.0%/UG:100.0%)			Projects 1 (G:100.0%/UG:100.0%)			
			PointsLost	Percentage	C	PointsLost	Percentage	C	PointsLost	Percentage	C	
U000000001	Adam Alexander	UG	3.0	70.00								
U000000003	Carl Cockoto	UG										
U000000002	Betty Burgundy	G										
U000000004	Debra Donkey	G										

**Settings and features available**

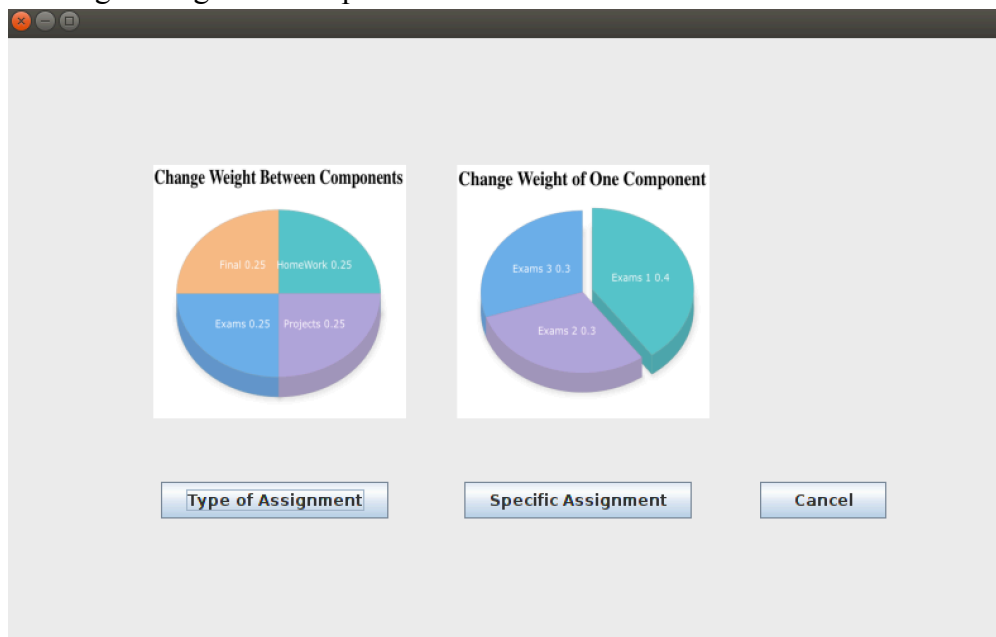
Click on empty space to add comment

Annotations:

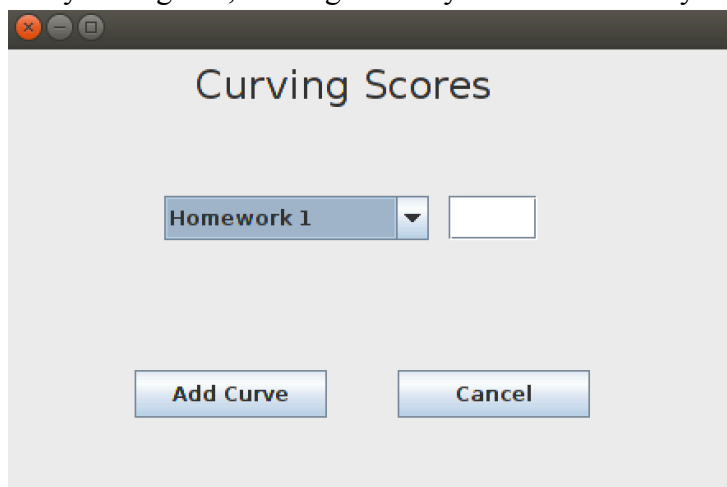
- Add/Delete specific assignment (points to Add Component button)
- Click to view student information (points to BUID column header)



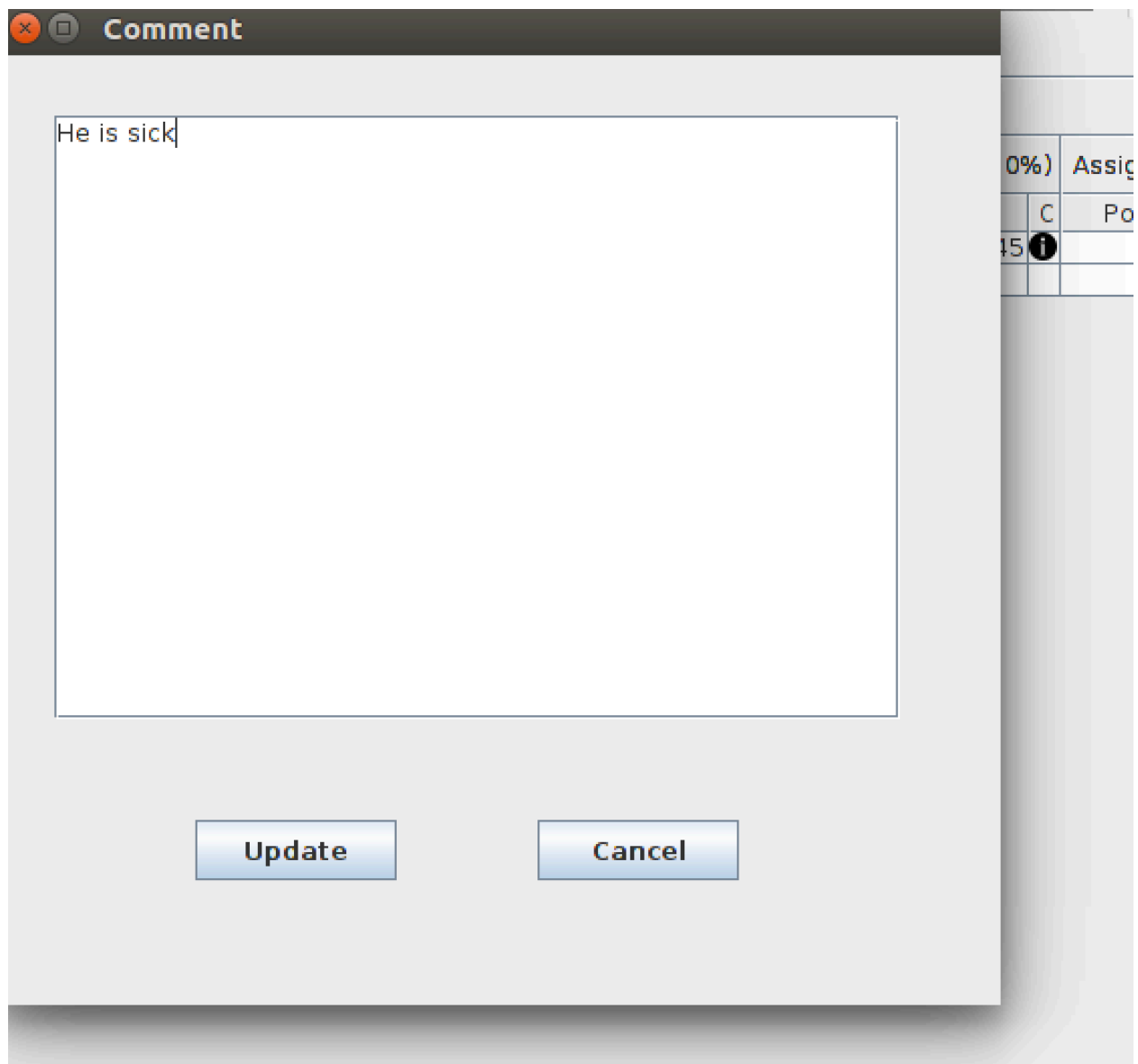
- 1) **Change Weights-** The professor is able to change the weights of each of the categories of assignments by pressing “change components” or changing single assignments by clicking “change sub-components”.



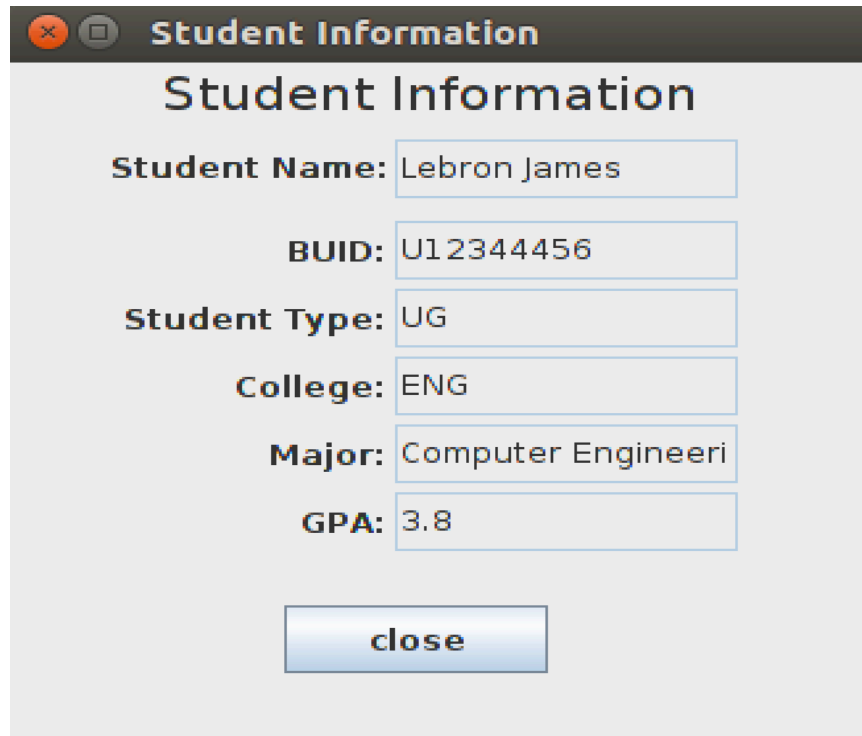
- 2) **Add Component-** By clicking add component, the professor is able to add an assignment to the list of assignments that they already have. This will allow them to add “homework 3” to the existing homework list containing “homework 1” and “homework 2”. They will also be able to initialize what the maximum score possible is for that assignment on the same page.
- 3) **Curve-** The curve button will produce a new frame that’ll allow the professor to add points to every students’ assignments grade. By selecting the assignment that they want from a drop down menu and typing in the number of points they wish to add to everyone’s grade, curving instantly and automatically.



- 4) ***Adding Comments to Assignments***-The professor will be able to add a comment to any assignment that any student has completed. To add comments, the professor must input the number of points lost by the student in an assignment, and then a black icon with the letter “i” will appear. If the professor presses the button, a new popup will appear where the professor will be able to write a comment about that students’ assignment. This can include notes about the assignment being late, the student coming to talk to the professor about that assignment, students coming to office hours about that assignment, and anything else the professor would deem worthy of commenting.



- 5) **Show Student Information**-To view a students' general information, the professor can press on their BUID numbers to the left of the screen. By pressing the BUID button, a new pop up menu will appear with a students' full name, their BUID number, the college they are in, their major, the students' GPA, and if they are undergrad or graduate students. The professor can press back and will go back to the course details page.

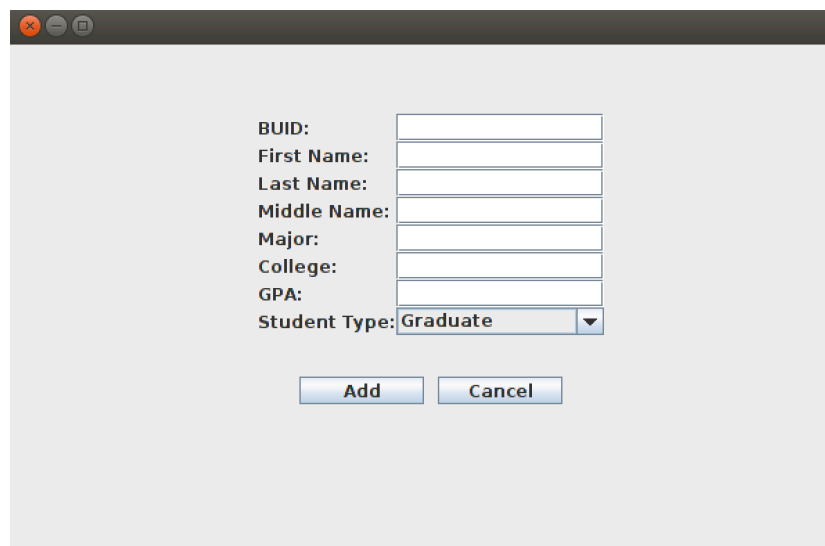


The screenshot shows a window titled "Student Information" with a dark header bar containing standard window controls. The main content area is light gray and displays the following information in a form-like layout:

- Student Name:** Lebron James
- BUID:** U12344456
- Student Type:** UG
- College:** ENG
- Major:** Computer Engineeri
- GPA:** 3.8

At the bottom center of the window is a blue button labeled "close".

- 6) **Adding Students to Course**-To add a new student to an existing course, the professor can press the + icon on the left hand-side of the page above the BUID numbers. A new frame will appear and the professor will have to input student information like name, build number, major, gpa, college, and undergrad/graduate type. Once they have completed imputing the information, they can press add and the student will appear on the course details page.

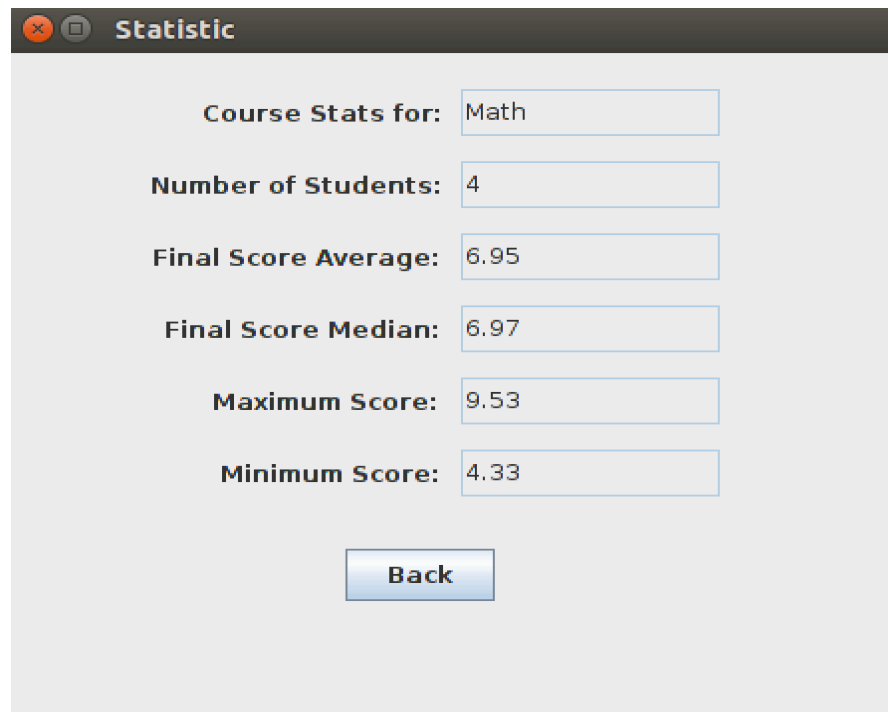


The screenshot shows a form for adding a new student. It features a light gray background and a dark header bar with window controls. The form fields are arranged vertically on the left, with corresponding input boxes on the right:

- BUID:** [input box]
- First Name:** [input box]
- Last Name:** [input box]
- Middle Name:** [input box]
- Major:** [input box]
- College:** [input box]
- GPA:** [input box]
- Student Type:** Graduate (dropdown menu)

At the bottom of the form are two buttons: "Add" and "Cancel".

- 7) **Dropping Students from Course**-To drop a student from an existing course, the professor can press the – icon to the left hand side of the page above the BUID numbers. A new frame will appear and drop down menu of all of the students in the course will appear. They will then select one of the students and press the drop button. The student will be removed from the course object.
- 8) **Calculate Final**- Once all of the grades for all students are inputted, calculating final grades according to each assignments weight and curves becomes completely automated. The professor will be able to press a single button and every students' total grade will be printed to the side on the same row as the student.
- 9) **Print Statistics**- To calculate statistical information about the course and the student's final grades, the professor can press “print statistics” and a pop up with information will appear. The info that will be portrayed includes the course name, the number of students, average final score, median final score, max and min final scores.



The image shows a software window titled "Statistic" with a dark header bar containing standard window controls (close, maximize, minimize). The main area is light gray and displays the following statistics for the course "Math":

Statistic	Value
Course Stats for:	Math
Number of Students:	4
Final Score Average:	6.95
Final Score Median:	6.97
Maximum Score:	9.53
Minimum Score:	4.33

At the bottom center of the window is a blue button labeled "Back".

- 10) **Logout**- Will save any changes and will bring the professor to the main sign-in page.

# Architecture Design Document

## Summary:

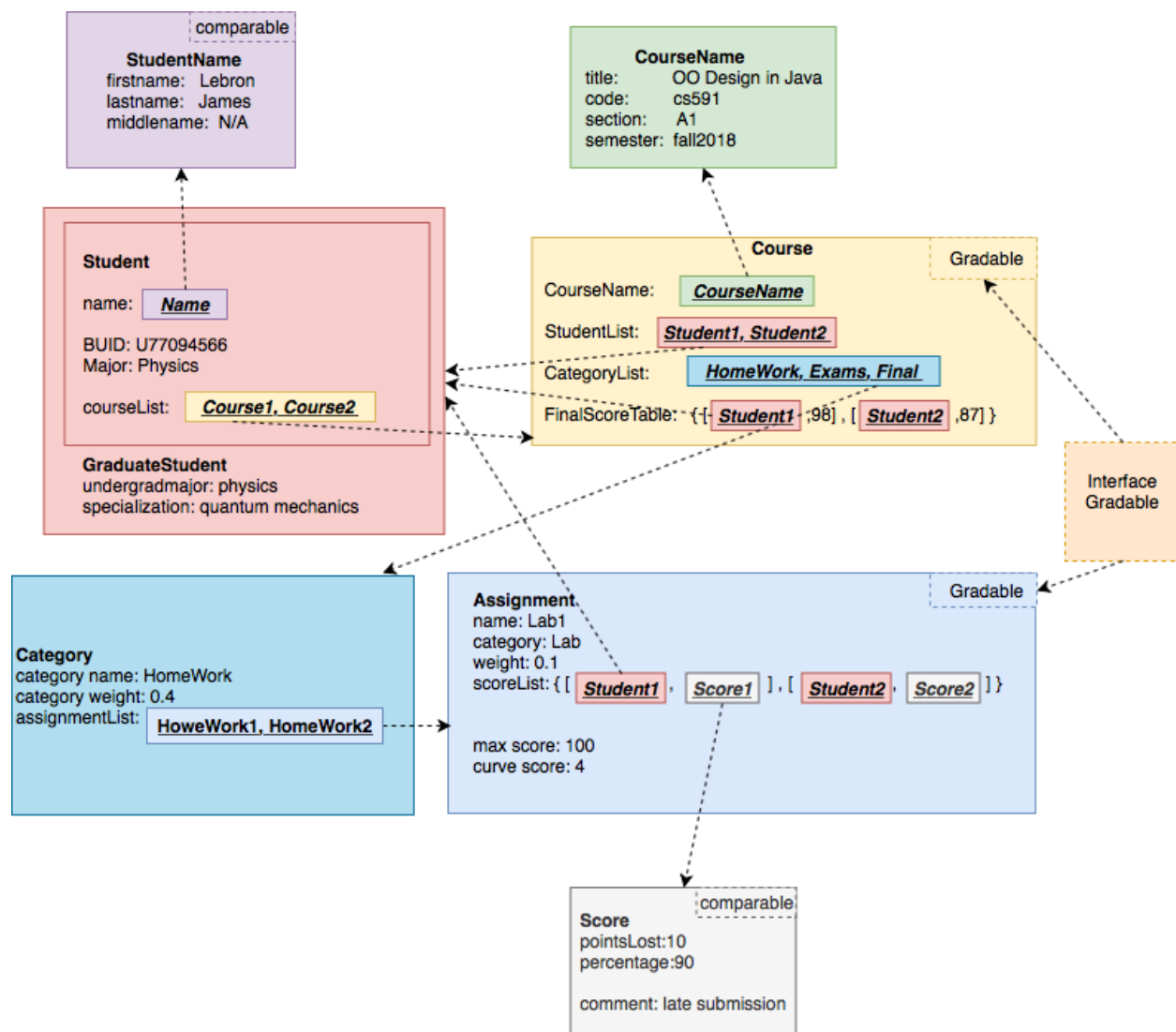
Our Grading System EZGrader is implemented by JAVA based on Object-Oriented design. We create three packages in our final code which are objects, gui and db. objects contains all the class we designed for this projects. gui includes the user interface pages developed by JAVA SWING. db is the package that associates with database(SQL).

## Object Model and Implements:

- Interface Gradable
  - This interface provides functions of calculating summarized statistics like mean, median.
- Abstract Class Student
  - This class will store individual student information. This will include an object of type StudentName as name, major, gpa, college, the type of student they are, and a list of courses that the student will be a part of.
- Class StudentName
  - This class will store three components, including the students' first name, middle initial character, and last name.
- Class Graduate
  - This class will store the students undergraduate major.
- Class Undergraduate
  - This class will store the students' academic year they're in (senior, junior, sophomore, firstyear).
- Class Course
  - This class will be where instances of Course objects will originate. Each course will include class ID, the class name, list of students in the class, list of categories that will be assigned for the class, and more found below under the Course class.
- Class CourseName
  - The CourseName class will include the name of the course, the course code, the section that the teacher is teaching, and the semester that they are currently teaching in.
- Class Category
  - This class will store information about each category, it shows the components of one course. The class will include the category name, undergraduate weight, graduate weight and a list of assignments of this category
- Class Assignments

- The assignments class will be an umbrella class for all of the assignments that the teacher will create, including homework, projects, exams, quizzes, finals, and other assignments. This will have a name, weight, category, and hashmap that connects students to their assigned scores on each assignment.
- Class Score
  - This class will store information about each assignment, including the maximum number of points an assignment could be worth, the points lost by the student, and the percentage of the total score the student received as a score.
- Class File
  - This class provides the function to read the students information from the existing file and load the information into the database.

## Object and Class Relationship Diagram:

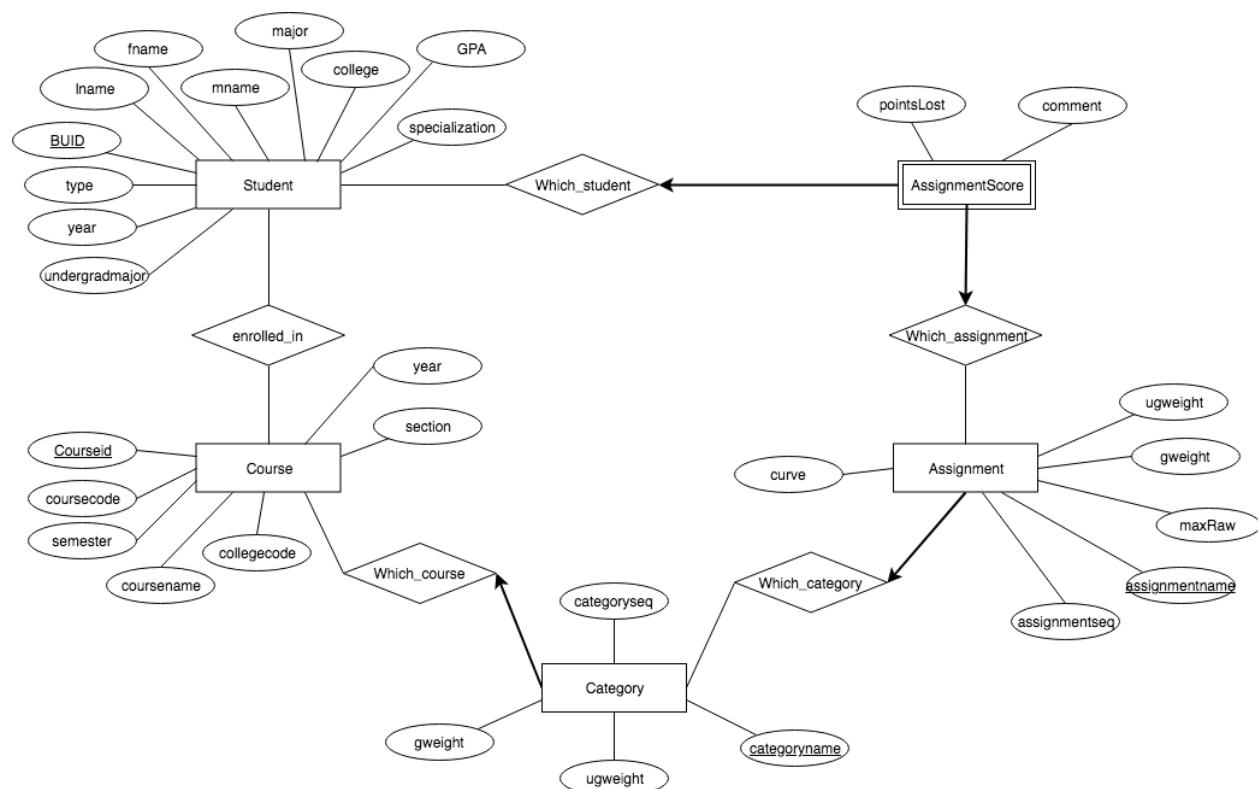


## Database:

EZGrader uses relational database with MySQL as the DBMS. The data are stored in 6 tables. The design of the relational model is very similar to the design of objects. For each of the main object (student, course, category, assignment, score), there is a corresponding table to store the data. We also have a table to store the enrollment information between student and course. Foreign keys and delete cascading are implemented to guarantee the correctness of the data.

To separate the database from the other parts, a Database class is implemented to be the interface. Methods in Database class are responsible to do the SQL query and update. Application functions that need to make changes to database need to call methods in Database class and provide the inputs, SQL sentence will be generated automatically.

### ● Database Model: 1. The E-R Diagram



## 2. The relational schema:

```
CREATE TABLE Course (  
  courseid INT NOT NULL AUTO_INCREMENT,  
  coursecode VARCHAR(10) NOT NULL,  
  semester VARCHAR(10) NOT NULL,  
  coursename VARCHAR(20) NOT NULL,  
  collegecode VARCHAR(3) NOT NULL,  
  year INT NOT NULL,  
  section VARCHAR(2),  
  PRIMARY KEY (courseid)  
);
```

```
CREATE TABLE Category (  
  courseid INT NOT NULL,  
  gweight FLOAT NOT NULL,  
  ugweight FLOAT NOT NULL,  
  categoryname VARCHAR(20) NOT NULL,  
  categoryseq INT NOT NULL,  
  PRIMARY KEY (courseid, categoryname),  
  FOREIGN KEY (courseid) REFERENCES Course(courseid) ON DELETE CASCADE  
);
```

```
CREATE TABLE Student (  
  buid CHAR(10) NOT NULL,  
  fname VARCHAR(20) NOT NULL,  
  lname VARCHAR(20) NOT NULL,  
  mname VARCHAR(20),  
  major VARCHAR(30),  
  college CHAR(3),  
  gpa FLOAT(4),  
  specialization VARCHAR(40),  
  stu_type VARCHAR(10) NOT NULL,  
  stu_year CHAR(4),  
  undergradmajor VARCHAR(30),  
  PRIMARY KEY (buid)  
);
```

```
CREATE TABLE Assignment (  
  courseid INT NOT NULL,  
  gweight FLOAT NOT NULL,  
  ugweight FLOAT NOT NULL,  
  assignmentname VARCHAR(20) NOT NULL,  
  assignmentseq INT NOT NULL,  
  categoryname VARCHAR(20) NOT NULL,  
  maxraw FLOAT,  
  curve FLOAT,  
  PRIMARY KEY (courseid, assignmentname),  
  FOREIGN KEY (courseid) REFERENCES Course(courseid) ON DELETE CASCADE,  
  FOREIGN KEY (courseid, categoryname) REFERENCES Category(courseid, categoryname) ON DELETE CASCADE  
);
```

```
CREATE TABLE Enrollment (  
  buid CHAR(10) NOT NULL,  
  courseid INT NOT NULL,  
  rawtotal FLOAT,  
  PRIMARY KEY (buid, Courseid),  
  FOREIGN KEY (buid) REFERENCES Student(buid) ON DELETE CASCADE,  
  FOREIGN KEY (courseid) REFERENCES Course(courseid) ON DELETE CASCADE  
);
```

```
CREATE TABLE AssignmentScore (  
  buid CHAR(10) NOT NULL,  
  courseid INT NOT NULL,  
  assignmentname VARCHAR(20) NOT NULL,  
  pointslost FLOAT,  
  comment VARCHAR(500),  
  PRIMARY KEY (buid, courseid, assignmentname),  
  FOREIGN KEY (buid) REFERENCES Student(buid) ON DELETE CASCADE,  
  FOREIGN KEY (courseid) REFERENCES Course(courseid) ON DELETE CASCADE,  
  FOREIGN KEY (courseid, assignmentname) REFERENCES Assignment(courseid, assignmentname) ON DELETE CASCADE,  
  FOREIGN KEY (buid, Courseid) REFERENCES Enrollment(buid, Courseid) ON DELETE CASCADE  
);
```

## ● Connection between Database and Objects:

### 1. Tables and their relation to objects:

- Student table: Stores the information of each student, including the attributes in **Student**, **StudentName**, **Graduate**, **Undergraduate** objects. The field “type” indicates the type of the student.
- Course table: Stores the attributes of **Course** objects. Auto increment id will be assigned.
- Category table: Stores the information of **Category**, a category sequence number will be saved to decide the display order in UI.
- Assignment table: Stores the information of **Assignment**, an assignment sequence number will be saved to decide the display order in UI.
- Assignment Score table: Stores the attributes of **Score** objects.



- Enrollment table: Stores the enrollment information between student and course. Corresponding to the **studentList** in Course object, and **courseList** in Student objects.

## 2. The interface between MySQL database and other parts:

All database and SQL related functions are implemented in a public class "Database". The class includes a list of Student objects, a list of Course objects and methods that query or update the database.

- **Lists**: The two lists store the object version of data. We have a special function `db.update()` that loads data from MySQL database into these two lists. When `db.update()` is called, all student in database will be loaded into the studentlist, all courses will be loaded in to the courselist. Categorys, assignments and scores will be loaded too, stored in the course/student objects.
- **Methods**: **`db.connect()`**: Connect to the MySQL database. Need to provide the username and password of the MySQL database.

**`db.disconnect()`**: Close the connection after using the database.

**`db.update()`**: Load everything from the database to the `db.studentlist` and `db.courselist`. Courses are loaded in to `db.courselist`, students are loaded in to `db.studentlist`, categories are loaded in to `courses.categorylist`, assignments are loaded in to `categories.assignmentlist`, scores are loaded in to `assignments.scorelist`.

**`db.dropEntireDB()`**: Delete all records in database.

*Data operation methods*: Methods for query, insert, update. E.g. **`db.updateCurve()`**, **`db.addStudent()`**.

## **What is Next:**

- ✓ Implement a “what-if” mode that will show the result of all operations before they are applied. This visualization will help the professor better understand course data and make the appropriate changes without losing original information.
- ✓ The user prefers to change weight by clicking the button near a specific assignment. However, due to the time constraint, we implemented change weight function in another way. We will finish this part in the next step.