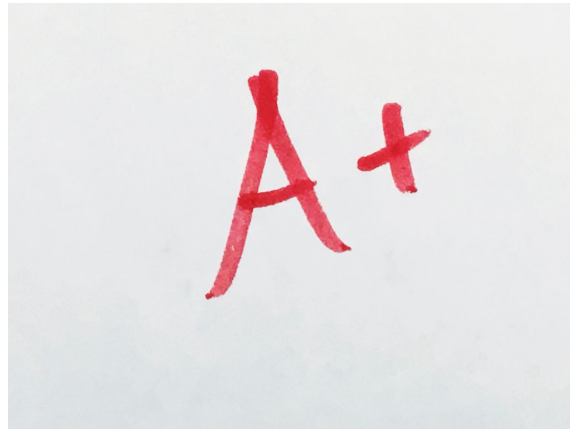


Easy Grader



Emmanuel Amponsah
Yifei Feng
Xiaotong Niu
Sichi Zhang
CS 591 P1 Group 3

Object Oriented Design and Development in Java
Computer Science
Boston University
May 6th, 2019

Table of Contents

1. Introduction

- 1.1 Overview
- 1.2 Introduction
- 1.3 Problem Motivation
- 1.4 Mission Statement

2. Design Document

- 2.1 Introduction
- 2.2 OO Design
 - 2.2.1 Introduction
 - 2.2.2 Student Entity
 - 2.2.3 Assignment Entity
 - 2.2.4 Class Entity
- 2.3 Database Model
- 2.4 UI/UX Decisions
- 2.5 UI Constraints
- 2.6 Services
- 2.7 Customized GUI Components
- 2.8 Data Access Object

3. User Document

- 3.1 Introduction
- 3.2 Class Management
 - 3.2.1 Add/Remove a Class
 - 3.2.2 Load Existing Class
 - 3.2.3 Access Grade View
 - 3.2.4 Access Student View
 - 3.2.5 Access Grading Criteria
- 3.3 Assignment Management
 - 3.3.1 Add Assignment
 - 3.3.2 Remove Assignment
 - 3.3.3 Edit Grading Criteria
- 3.4 Grade Management
 - 3.4.1 Add Grade
 - 3.4.2 Grade Locking
- 3.5 Student Management

- 3.5.1 Add/Remove Student
 - 3.5.2 Load Students from CSV
 - 3.5.3 Student Search

- 3.6 Security Overview

- 3.6.1 Instructor Login

4. Conclusion

1. Introduction

- 1.1. This document is going to be split up into two major parts. The first is the design document, where we go over our object-oriented design, UI decisions, database layout and explain why we made these choices.
- 1.2. The purpose of this document is to build a new grading system for our client, Christine Papadakis, that enables her to easily manage her classes and student grades
- 1.3. Easy Grader is a grading system which aims to replace our client's current method of grading. The current solution is well known for its flexibility such as allowing the user to insert new rows, and perform statistics on a certain column. However, it still has some drawbacks. First and foremost, it is not very reusable, our client has to take time to set up this system at the beginning of every semester. This can be tedious and very time-consuming. Second, the user experience as well as interface on the current system is not very user-friendly, making it very easy to make mistakes that can costs our client hours in man hour to fix.
- 1.4. Easy Grader delivers to our client a new, robust, and user-friendly system for managing her classes, students and their grades.

2. Design Document

2.1. This section of the document outlines how we implemented the system and justifies why we chose to things the way we did. In this section we'll go over the the object oriented design, discuss our database model, and go over the user interface as well as user experience decisions.

2.2. OO Design / Diagram

2.2.1. To solve the problem for grading, the Easy Grader team decided to split this problem into its three most basic form and tackle it from there. Both our object oriented design as well as database model are simple to avoid any complex bugs during development. We realized that a complicated OO Design would also mean a complicated database model, which we did not want since our DB and classes are very tightly. In Figure 2.2.1, we outline our class structure.

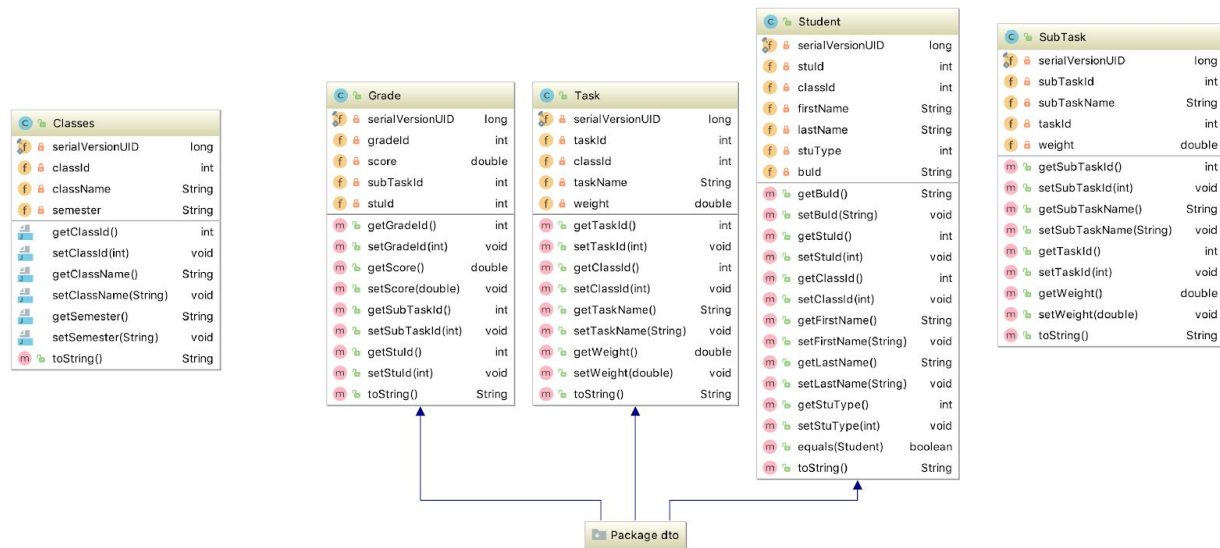
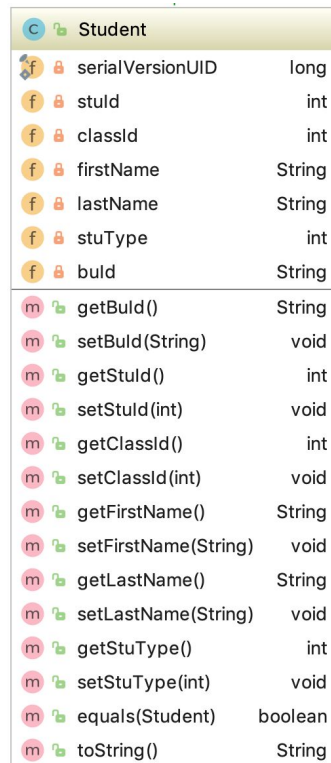


Figure 2.2.1

2.2.2. The student entity/class is holds all relevant information about a student such as name, ID, and the class that they belong in. Student's currently don't inherit or get inherited from because we currently do not have a distinction between undergraduate

and graduate student so there's no need for any type of inheritance. We also avoid any type of object composition here due to the fact that we did not want to have our database have tables nested in tables.



Student		
f	serialVersionUID	long
f	stuld	int
f	classId	int
f	firstName	String
f	lastName	String
f	stuType	int
f	buld	String
m	getBuld()	String
m	setBuld(String)	void
m	getStuld()	int
m	setStuld(int)	void
m	getClassId()	int
m	setClassId(int)	void
m	getFirstName()	String
m	setFirstName(String)	void
m	getLastName()	String
m	setLastName(String)	void
m	getStuType()	int
m	setStuType(int)	void
m	equals(Student)	boolean
m	toString()	String

Figure 2.2.2

- 2.2.3. The assignment entity/class is our way of taking care of a given classes' grading criteria. Our assignment entity is broken up into two main components, Tasks and SubTasks. We realized that every grade has one overarching category, and a lot of underlying sub categories. For example a homework category can have multiple homework assignments under it, so we took that idea and represented it in the simplest way possible. In our task and subtasks classes we use this idea of fake composition to avoid having nested tables, where each task has an ID and any subtask spring from that main task has an attribute called "task_id" which allows for us to easily figure out which task a subtask relates to. Moving away from the assignment perspective, for any grade we gave it a subtask ID since grades cannot be under main tasks, they have to be attached to a

subtask. This way of managing grades is something we believe to be very easy, and insures that we do not make any mistakes because all of the sub entities in the assignment entity are attached by a unique ID so there's no way of mixing them up. This ensures that our grading system is reliable because there is no way to easily make a mistake when grades get entered. Refer to figure 2.2.3

Grade	Task	SubTask
<div>serialVersionUID</div> <div>long</div>	<div>serialVersionUID</div> <div>long</div>	<div>serialVersionUID</div> <div>long</div>
<div>gradelId</div> <div>int</div>	<div>classId</div> <div>int</div>	<div>subTaskName</div> <div>String</div>
<div>stulId</div> <div>int</div>	<div>taskName</div> <div>String</div>	<div>subTaskId</div> <div>int</div>
<div>score</div> <div>double</div>	<div>taskId</div> <div>int</div>	<div>taskId</div> <div>int</div>
<div>subTaskId</div> <div>int</div>	<div>weight</div> <div>double</div>	<div>weight</div> <div>double</div>

Figure 2.2.3

- 2.2.4. Class management/entity similar to all our other classes is designed to be simple to insure that we do not introduce any complex errors into the system. This also ensures that we have a system that is reliable for our client. The class entity is only responsible for holding a classID, name and semester. We use this base information to make sure all our other entities have the same information if they need to be the same and different otherwise.
- 2.3. Our database model like explained above resembles our class structure very closely because we wanted to make sure that all the layers of our application worked together in a cohesive fashion. This way of design ensures that our classes resemble our database, and our database resembles our services. Refer to figure 2.3.1 to see the database model.

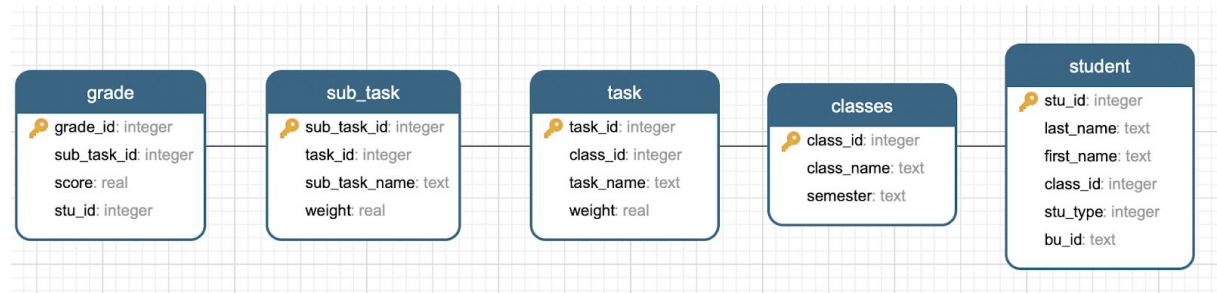


Figure 2.3.1

- 2.4. Our user interface along with experience is something that has gone through a lot of changes throughout the development process. We started of with a lot of screens, in hopes of helping the user avoid making any type of mistake. This included making sure that we made it difficult to find buttons that could have catastrophic effects on the system such as delete, or edit. In doing this we had screens that were like displayed in figure 2.4.1 where the user needed to click a lot just to get to an option.

Figure 2.4.1

This was a problem we realized because while we made it safer to use, we drastically reduced our user's experience. We made it very difficult to do something that shouldn't be difficult and this was not something that fit in our initial mission statement. So instead of hiding buttons in our actual system implementation, we used a combination of click and destroy and or edit. This means that if a user wanted to remove or edit an entry they would need to click on that specific entry as well as a

button like “remove” or “edit” for that change to even happen. For example of this look at figure 2.4.2.

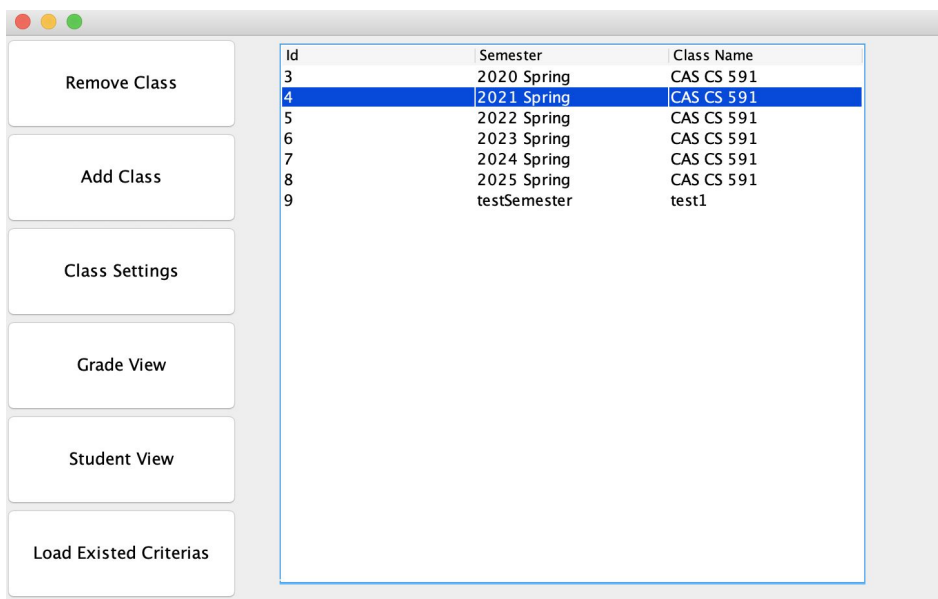


Figure 2.4.2

In figure 2.4.2, in order for a user to remove a class, they’ll first need to click the class such that it is highlighted, then after that click the “Remove Class” button. We found that this was the best way to design our UI to ensure the best user experience. Instead of forcing our users to go jump through hoops to accomplish simple tasks, we made it so that any type of change was easy to make. But the user would have to do so consciously unlike excel where any button click could result in a major error.

Another major UI decision that we chose not make because of the design constraints of the Java Swing/FX library was a card view for the students. Initially we wanted to make it so that the student view looked like the model represented in figure 2.4.3.

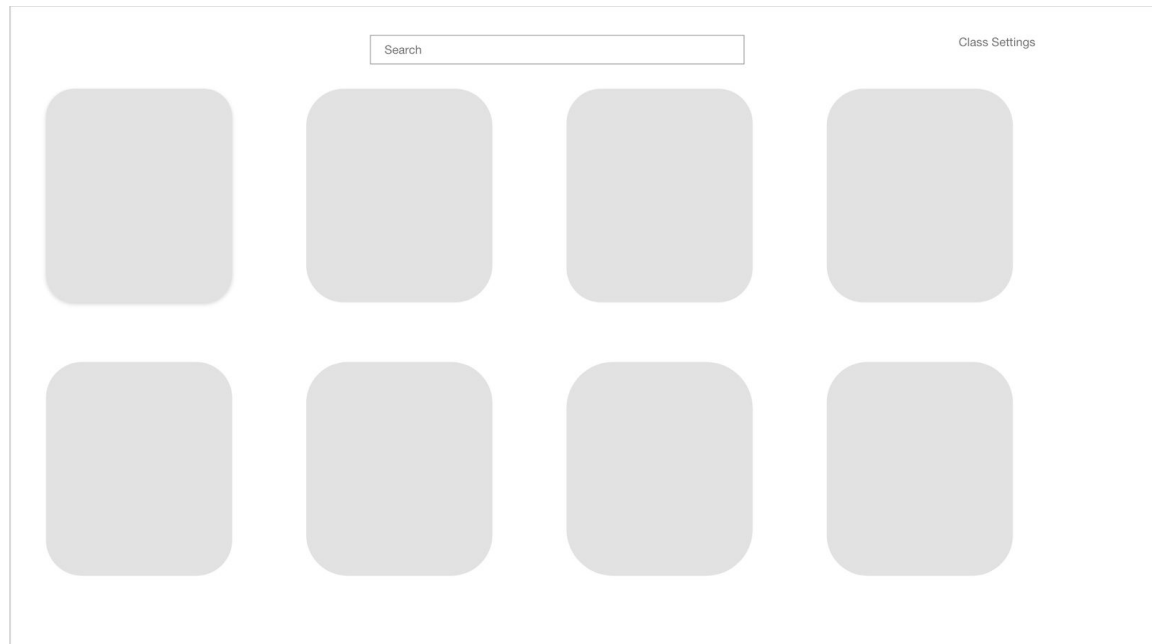


Figure 2.4.3

Instead of this we did something more closely aligned with the design of figure 2.4.4. This decision was made because we realized that we wanted our client to be familiar with the new system, and to do this we had to ensure that we kept the familiarity of the previous system. The card view for the student also introduced a lot of issues with the user experience that we wanted to avoid. The card view would have brought back the issue we initially had with making the user jump through hoops to accomplish simple tasks.

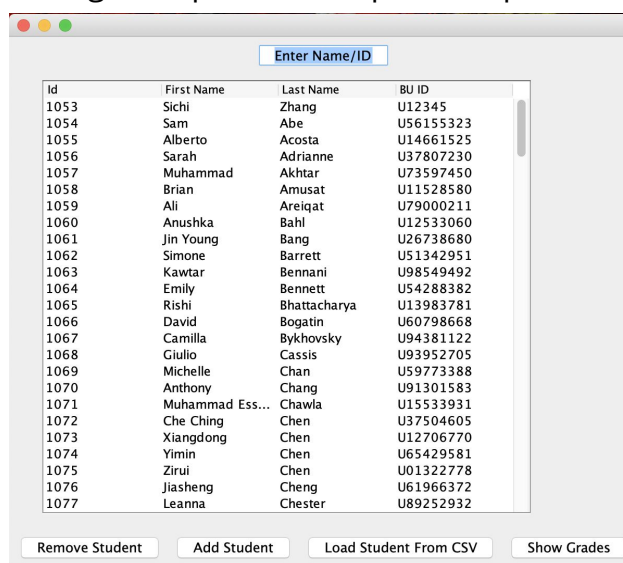


Figure 2.4.4

This model (figure 2.4.4) makes it easier for the user to manage their students, so this is why we went with that instead of the card view.

- 2.5. In terms of constraints, the only ones we had were the ones imposed by the java UI library, along with the ones imposed by our client's old system. We say this because we did not want to deviate too far from something that is familiar with our client.
- 2.6. Services are how we communicated to our backend through our front end. We used Sqlite as our database of choice. Our services handle the request from the front end and our process it to DAO . In figure 2.6.1 you can see the services we used to create, delete, students, assignments, and classes.

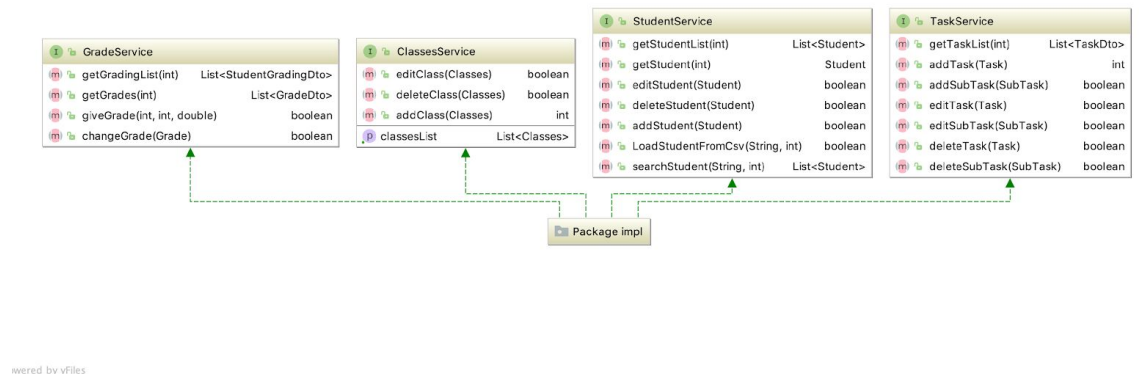


Figure 2.6.1

- 2.7. We decided to customize our GUI by inheriting JFrame or JPanel for GUI components. As a result, we can decide our own field parameters, constructors and corresponding methods for encapsulated variables in order to communicate between different GUI windows.
- 2.8. Our DAO (Data Access Object) is the bridge that connects our backend to our database. The DAO is responsible for all the operations that we invoke in services. All of our DAOs use data transfer objects (dto) or entity objects which inherit from the entities (discussed in sections 2.2.2 - 2.2.4) to encapsulate the data. We made this decision again just

because of how closely coupled all our models are. Figure 2.8.1 shows our DAO model, as well as our DTOs.

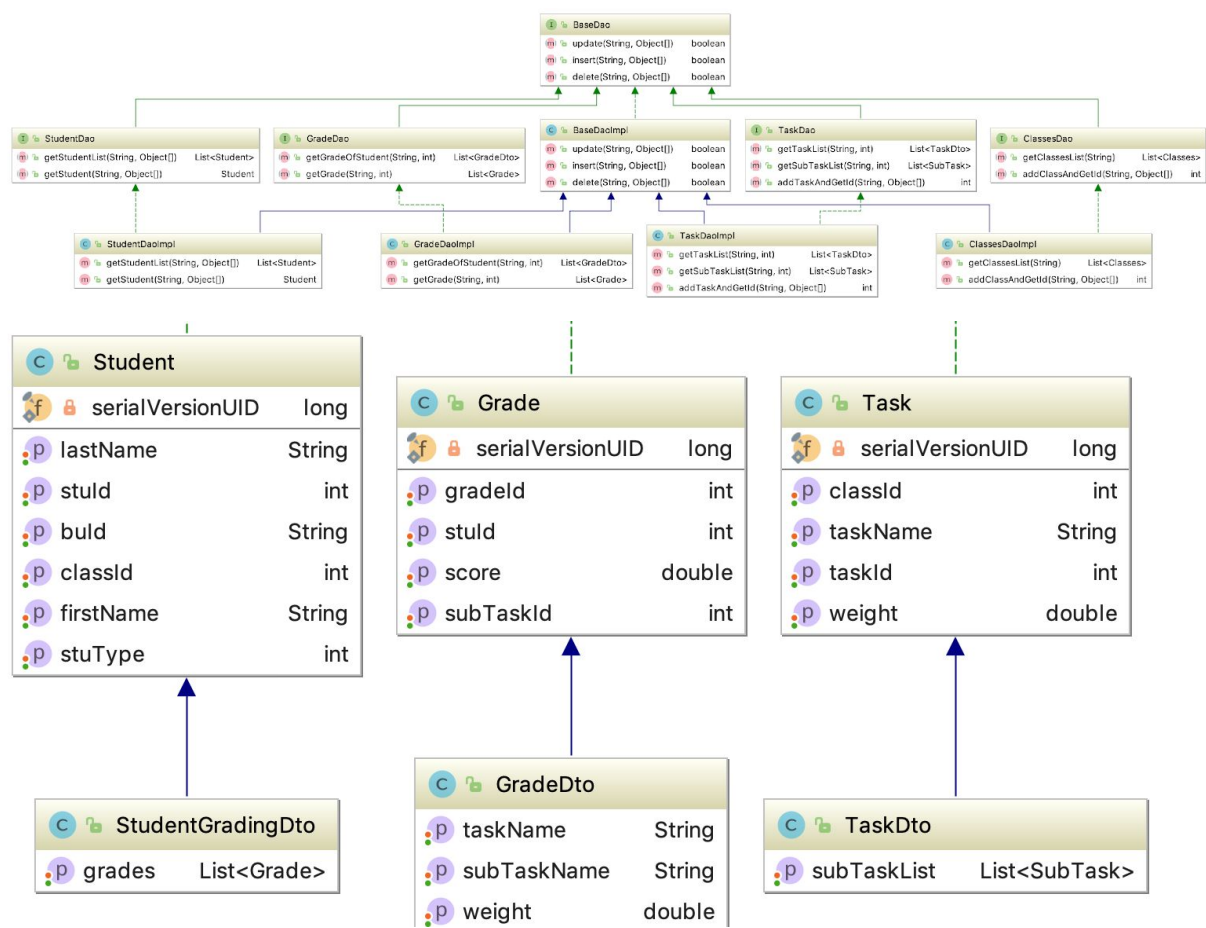


Figure 2.8.1

3. User Document

3.1. The user document will provide a high level overview on how to use Easy Grader. From class creation, to grading criteria update and beyond.

3.2. Class Management

3.2.1. To add a class once you are on the class view, click the “Add Class” button as displayed in Figure 3.2.1

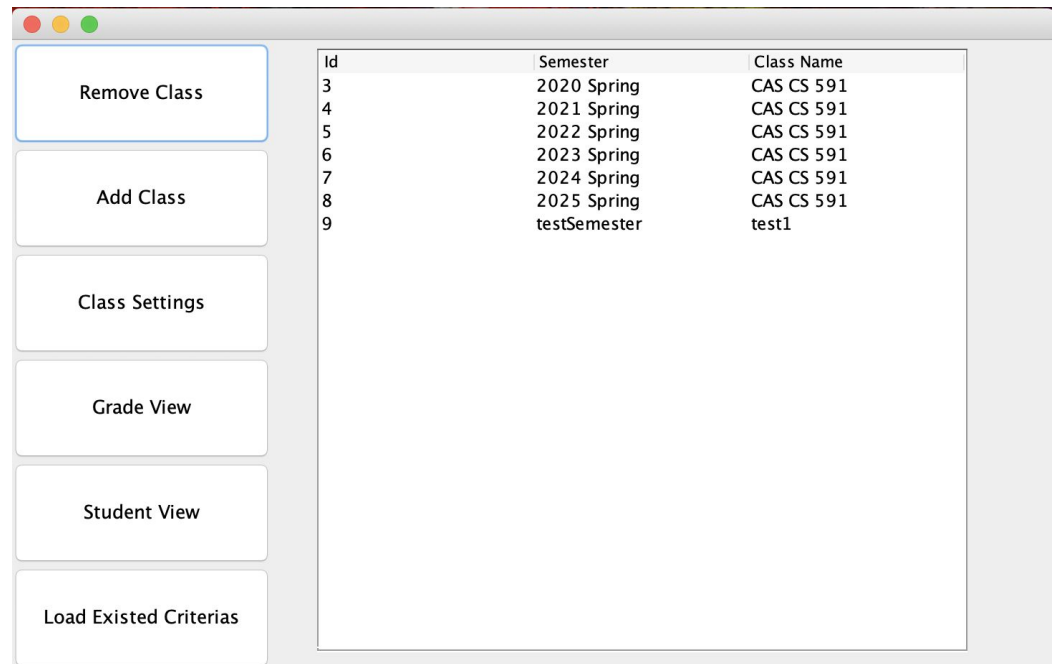


Figure 3.2.1

After this a new popup window should appear as displayed in Figure 3.2.1. Enter the class information and click “Create Class”

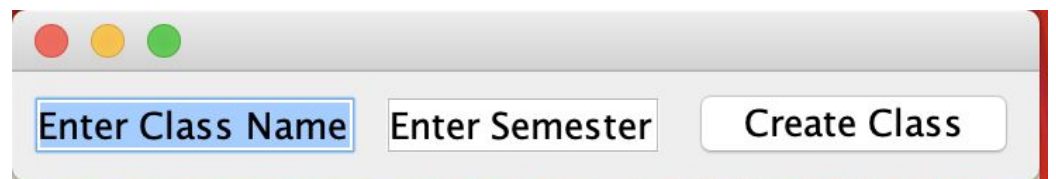


Figure 3.2.2

To remove a class, click on a class name, and then click the “Remove Class Button”

3.2.2 To create a class from an existing class with a specific grading criteria, click the “Load existed criterias” (refer to figure 3.2.1) and then

a new popup window will appear similar to figure 3.2.2 and enter the information of the class you want to clone .

3.2.3. To access the grade view, from the class view (figure 3.2.1) click "grade view"

3.2.4 To access the student view, from the class view (figure 3.2.1) click "student view"

3.2.5 To edit the grading criteria for a class, from the class view (figure 3.2.1) click "class setting" button

3.3. Assignment Management

3.3.1. Once you are on the assignment management view, to add a new assignment click on main category (i.g Exams) then the "Add Sub-category" button as displayed in figure 3.3.1

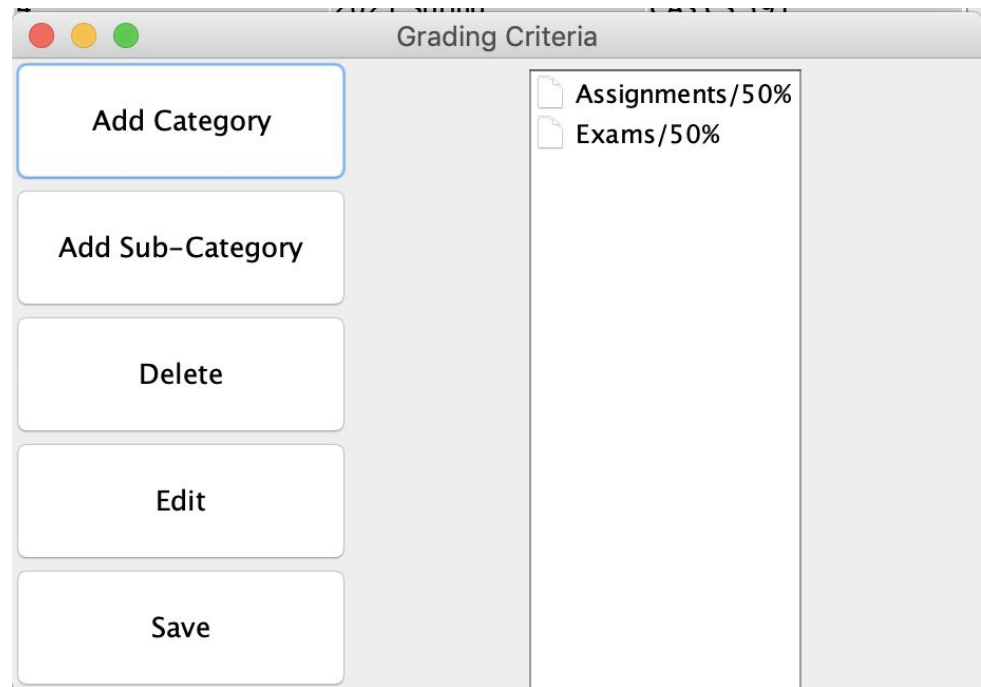


Figure 3.3.1

After this, a new popup window will appear, where you can enter the name of the assignment along with the percentage refer to figure 3.3.2

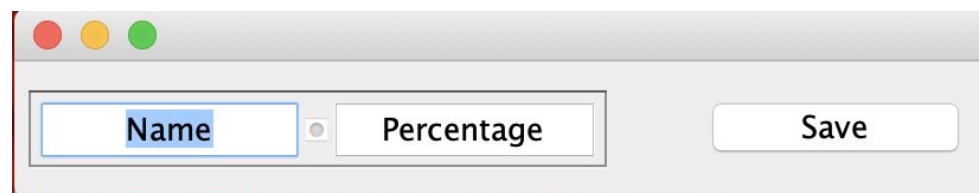
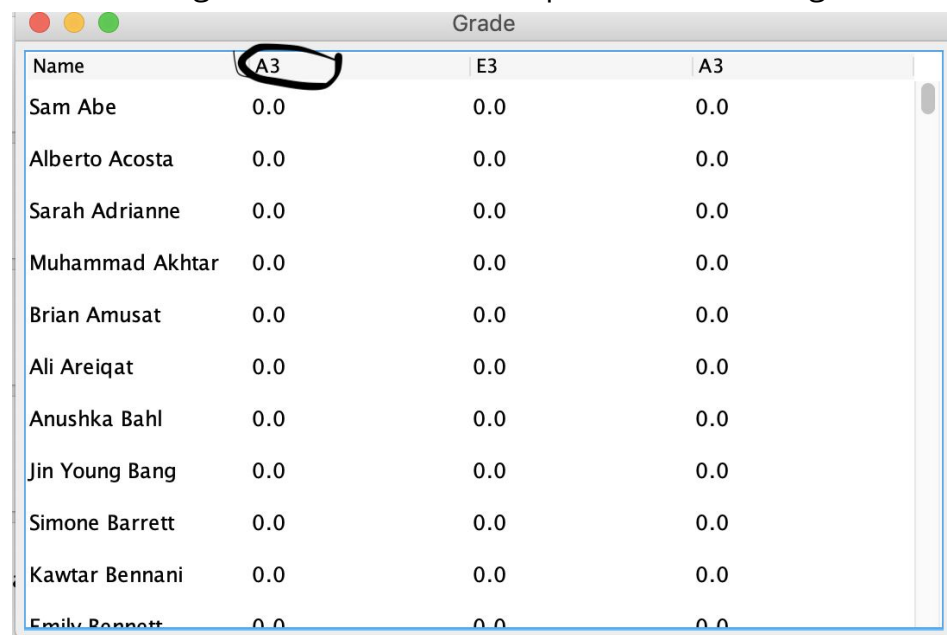


Figure 3.3.2

- 3.3.2. To remove an assignment, click the assignment under the main category for example “Exams” (refer to figure 3.3.1) and then click the “Delete” button.
- 3.3.3. To edit the grading criteria in the assignment view (Figure 3.3.1) click the “Add Category” button, this will allow for you to change the weights along with the major categories.

3.4. Grade Management

- 3.4.1. To add a grade once you are in the grade management view, click the assignment name on the top row (circled in figure 3.4.1)

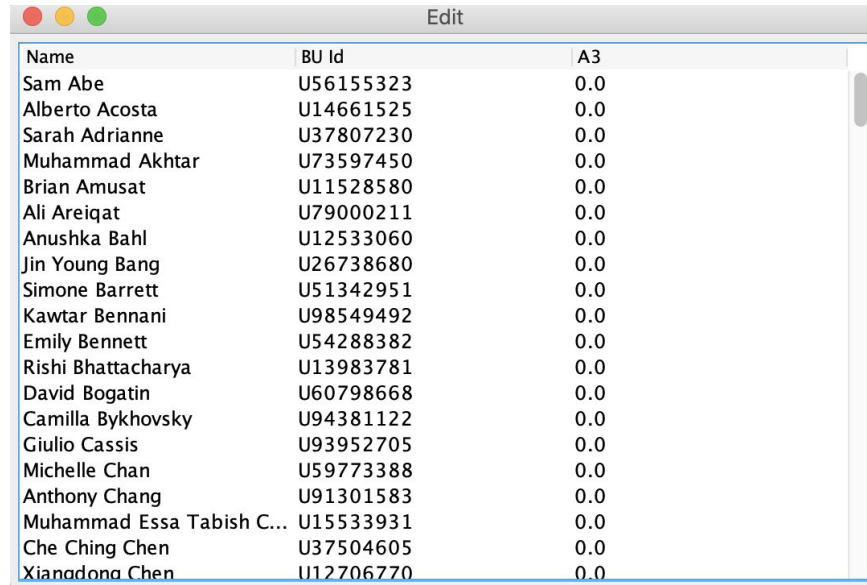


The screenshot shows a window titled "Grade" with a table containing student names and their grades in three categories: A3, E3, and A3. The "A3" header in the first row is circled. The table lists 12 students, all with a grade of 0.0.

Name	A3	E3	A3
Sam Abe	0.0	0.0	0.0
Alberto Acosta	0.0	0.0	0.0
Sarah Adrianne	0.0	0.0	0.0
Muhammad Akhtar	0.0	0.0	0.0
Brian Amusat	0.0	0.0	0.0
Ali Areiqat	0.0	0.0	0.0
Anushka Bahl	0.0	0.0	0.0
Jin Young Bang	0.0	0.0	0.0
Simone Barrett	0.0	0.0	0.0
Kawtar Bennani	0.0	0.0	0.0
Emily Bennett	0.0	0.0	0.0

Figure 3.4.1

Once you click that, you will get a view where you can enter the student grades, either in percentage form (100), or negative format (-5) as displayed in figure 3.4.2



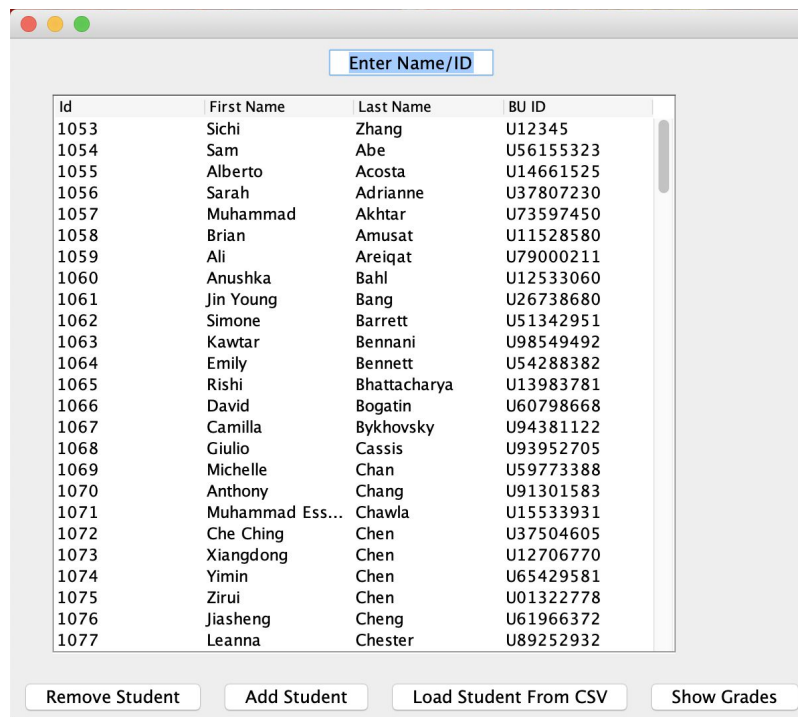
Name	BU Id	A3
Sam Abe	U56155323	0.0
Alberto Acosta	U14661525	0.0
Sarah Adrianne	U37807230	0.0
Muhammad Akhtar	U73597450	0.0
Brian Amusat	U11528580	0.0
Ali Areiqat	U79000211	0.0
Anushka Bahl	U12533060	0.0
Jin Young Bang	U26738680	0.0
Simone Barrett	U51342951	0.0
Kawtar Bennani	U98549492	0.0
Emily Bennett	U54288382	0.0
Rishi Bhattacharya	U13983781	0.0
David Bogatin	U60798668	0.0
Camilla Bykhovsky	U94381122	0.0
Giulio Cassis	U93952705	0.0
Michelle Chan	U59773388	0.0
Anthony Chang	U91301583	0.0
Muhammad Essa Tabish C...	U15533931	0.0
Che Ching Chen	U37504605	0.0
Xianadong Chen	U12706770	0.0

Figure 3.4.2

- 3.4.2. Grade locking is a feature that is automatically built in, so the user will not be able to edit grades without clicking on the name of the assignment and getting to the figure provided in 3.4.2

3.5. Student Management

- 3.5.1. Once you are in the student management view, to add a student click the “Add Student” button as displayed in figure 3.5.1.



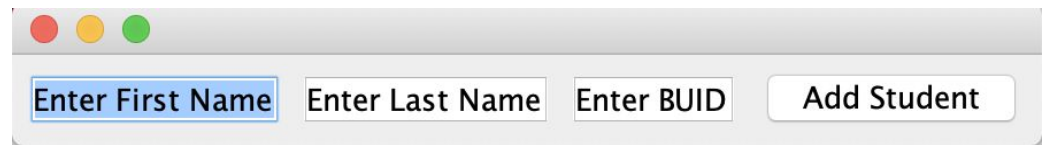
Enter Name/ID

Id	First Name	Last Name	BU ID
1053	Sichi	Zhang	U12345
1054	Sam	Abe	U56155323
1055	Alberto	Acosta	U14661525
1056	Sarah	Adrianne	U37807230
1057	Muhammad	Akhtar	U73597450
1058	Brian	Amusat	U11528580
1059	Ali	Areiqat	U79000211
1060	Anushka	Bahl	U12533060
1061	Jin Young	Bang	U26738680
1062	Simone	Barrett	U51342951
1063	Kawtar	Bennani	U98549492
1064	Emily	Bennett	U54288382
1065	Rishi	Bhattacharya	U13983781
1066	David	Bogatin	U60798668
1067	Camilla	Bykhovsky	U94381122
1068	Giulio	Cassis	U93952705
1069	Michelle	Chan	U59773388
1070	Anthony	Chang	U91301583
1071	Muhammad Ess...	Chawla	U15533931
1072	Che Ching	Chen	U37504605
1073	Xiangdong	Chen	U12706770
1074	Yimin	Chen	U65429581
1075	Zirui	Chen	U01322778
1076	Jiasheng	Cheng	U61966372
1077	Leanna	Chester	U89252932

Remove Student Add Student Load Student From CSV Show Grades

Figure 3.5.1

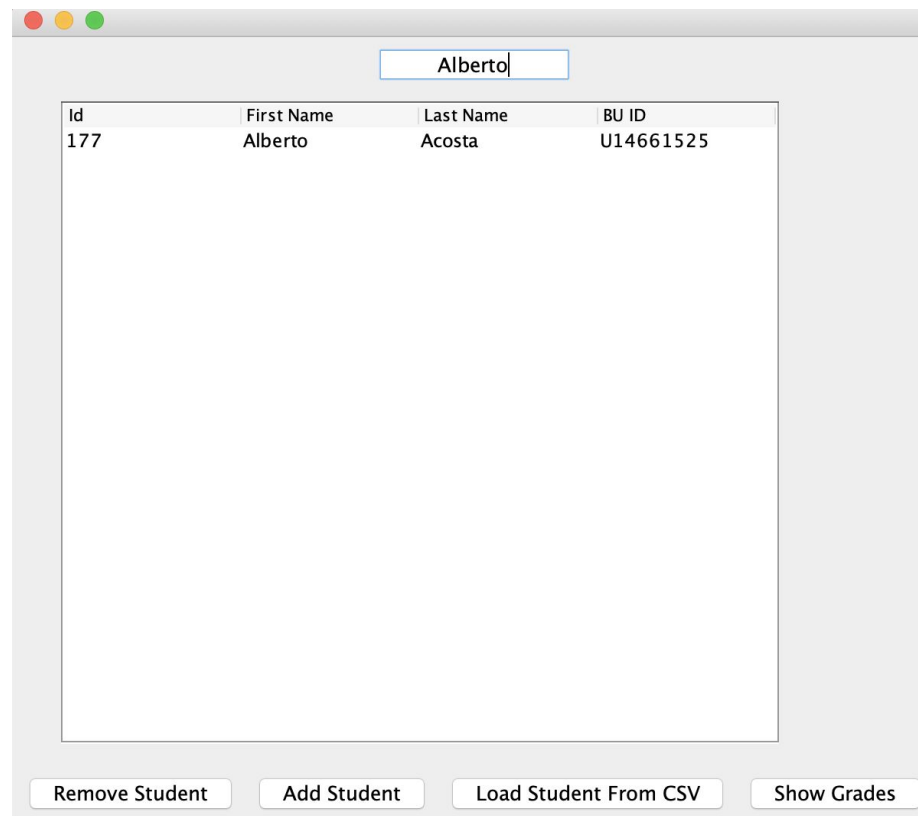
After clicking this a new popup window will appear, where you can enter the student information and the add a student. Refer to figure 3.5.2



A popup window with a light gray background and a title bar with three colored buttons (red, yellow, green). It contains four input fields arranged horizontally: 'Enter First Name' (highlighted with a blue border), 'Enter Last Name', 'Enter BUID', and 'Add Student'.

Figure 3.5.2

- 3.5.2. To load students from a CSV, click “Load Student From CSV” as displayed in figure 3.5.1. This will prompt you to select a file from your file system and then it’ll upload the CSV for you.
- 3.5.3. To search for a student enter their name in the search bar on top of the page and the student’s record should appear



A popup window with a light gray background and a title bar with three colored buttons (red, yellow, green). It features a search bar at the top with the text 'Alberto'. Below the search bar is a table with the following data:

Id	First Name	Last Name	BU ID
177	Alberto	Acosta	U14661525

At the bottom of the window are four buttons: 'Remove Student', 'Add Student', 'Load Student From CSV', and 'Show Grades'.

3.6. Security

- 3.6.1. Instructor login is the first screen any user sees as displayed in figure 3.6.1

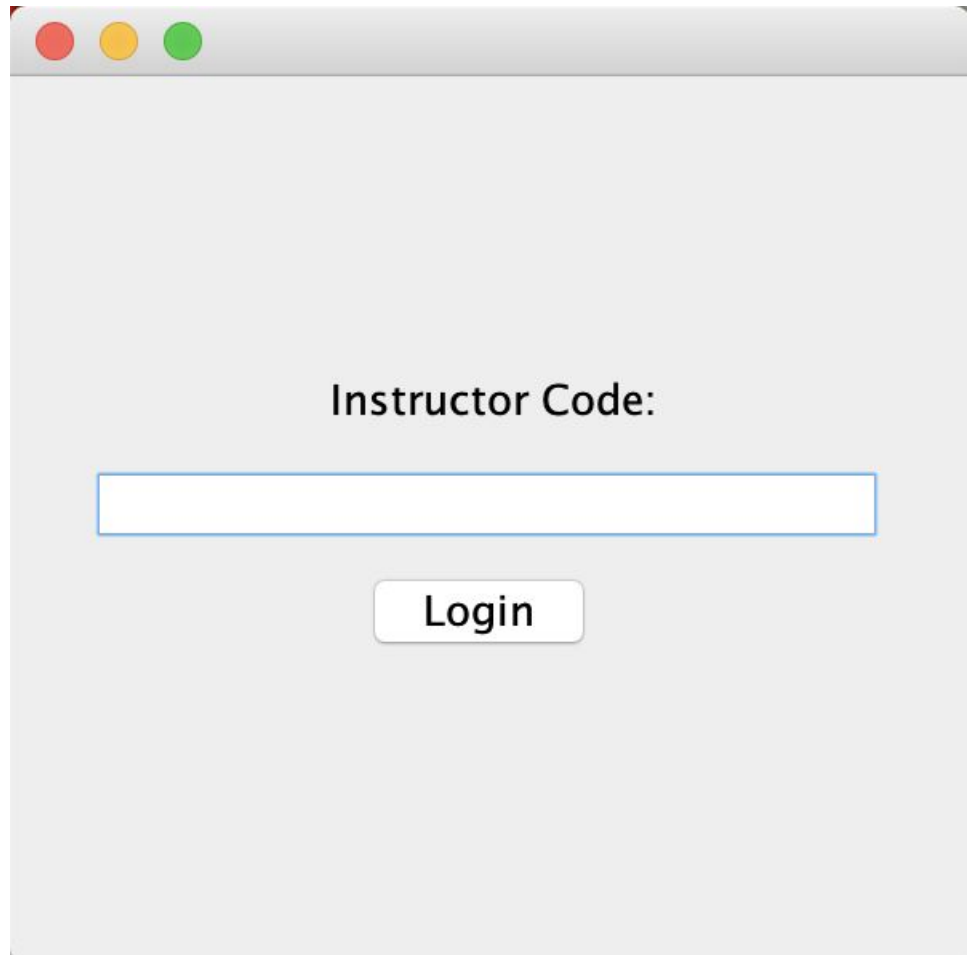


Figure 3.6.1

If the passcode is correct the user should see the popup window as displayed in 3.6.2 or if it's wrong they will see the screen displayed in figure 3.6.3



Figure 3.6.2



Figure 3.6.3

4. Conclusion

In conclusion, the Easy Grader team has completed what we think is a good first step in the right direction for our client CPK. In the future we plan on adding a lot more features such as being able to access statistics, sort grades and students, along with a plethora of other features. We also hope to try and make more daring UI changes to our clients current system to improve the workflow and make it easier to get grades done.