

## Semester Project: Gradebook

---

### Project Summary

Our semester project is a grading application called Gradebook that is designed to help students and teachers track and calculate grades across different classes. Our two team members, Will Loughlin and Sophia Chaltas are both seniors in the computer science department. The goal of our project is to provide a simple and effective platform for teachers to store, edit, and calculate grades for their students that allows students to keep track of their academic standing.

This is meant to be a replacement of physical grading systems like large gradebooks that are complex, time consuming, and can result in computation errors when putting final grades together. We will have a simple user interface created with the open source Java framework Vaadin that can be used as a teacher or student. This interface will allow teachers to perform necessary operations and students to check their grades. A key part of this project is that the system is simple, efficient, and easily understood.

We will implement several object oriented design patterns including the Template, Tracker, Observer, Factory, and Singleton patterns. We will use these patterns and other object oriented development methods to store, manipulate, save, and load objects simulating classes, teachers, students, and assignments to create a system to effectively manage and evaluate grades.

### Project Requirements

#### **Application:**

- Saving/Loading
  - Must be able to save and load teachers, students, classes, assignments, and grades.
  - Load saved state on server startup, automatic saving to prevent loss.
- GUI
  - User interface must be simple and effective.
  - Will use Vaadin (open source java GUI framework).
- Login
  - Login for students and teachers, provides security and tells the program whether the user is a student or teacher.
  - Registration option for users without an account.

**As a Teacher:**

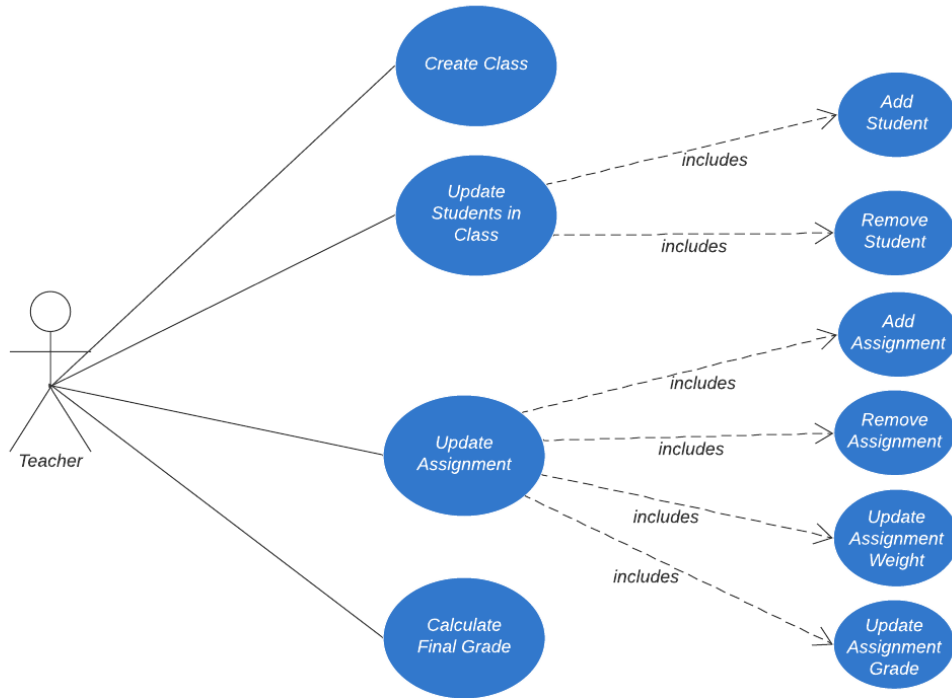
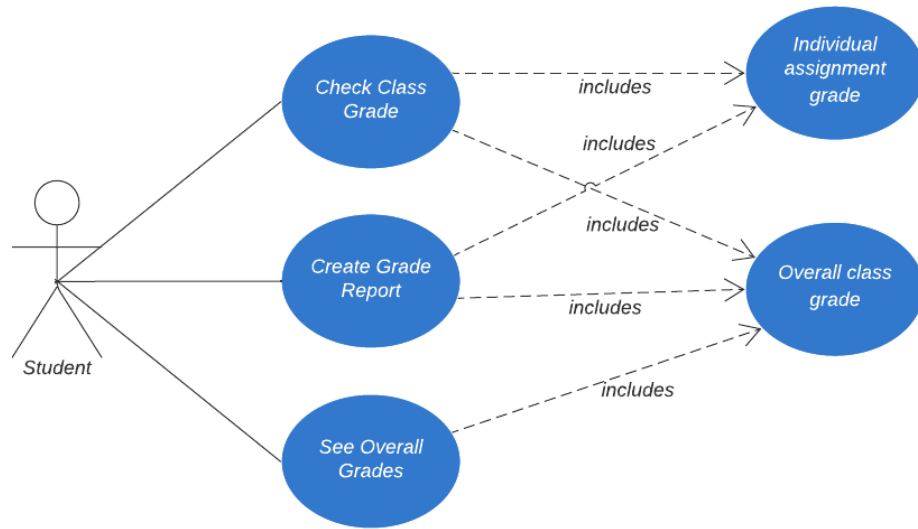
- Create classes and add/remove students
  - Teachers must be able to create a class object for each class they are teaching and add which students are in the class.
  - Must be able to add and remove students from classes after class is created.
- Add assignments with grades
  - Assignments can be added to a class with grades entered later.
  - Assignments must have a weight given by the amount of points assigned.
  - Must be able to add grades for each student on each assignment.
  - Grades can be changed after they are entered.
- Calculate final grades
  - Generate grade reports for each student in the class.
  - Output grade reports as separate .txt files.

**As a Student:**

- Check grades by class
  - Individual assignment grades and total class grade.
- Create grade report
  - Generate .txt file in the same format as generated by the teacher.
  - Includes assignments, grades, and total grade.
- See overall grades
  - See list of current classes with grades.
- Stretch Goals for student
  - Add/Drop classes: Students are able to enroll and drop classes from their portal.
  - Test different grades: Students are able to test the effect of different scores on ungraded assignments on their total grade.

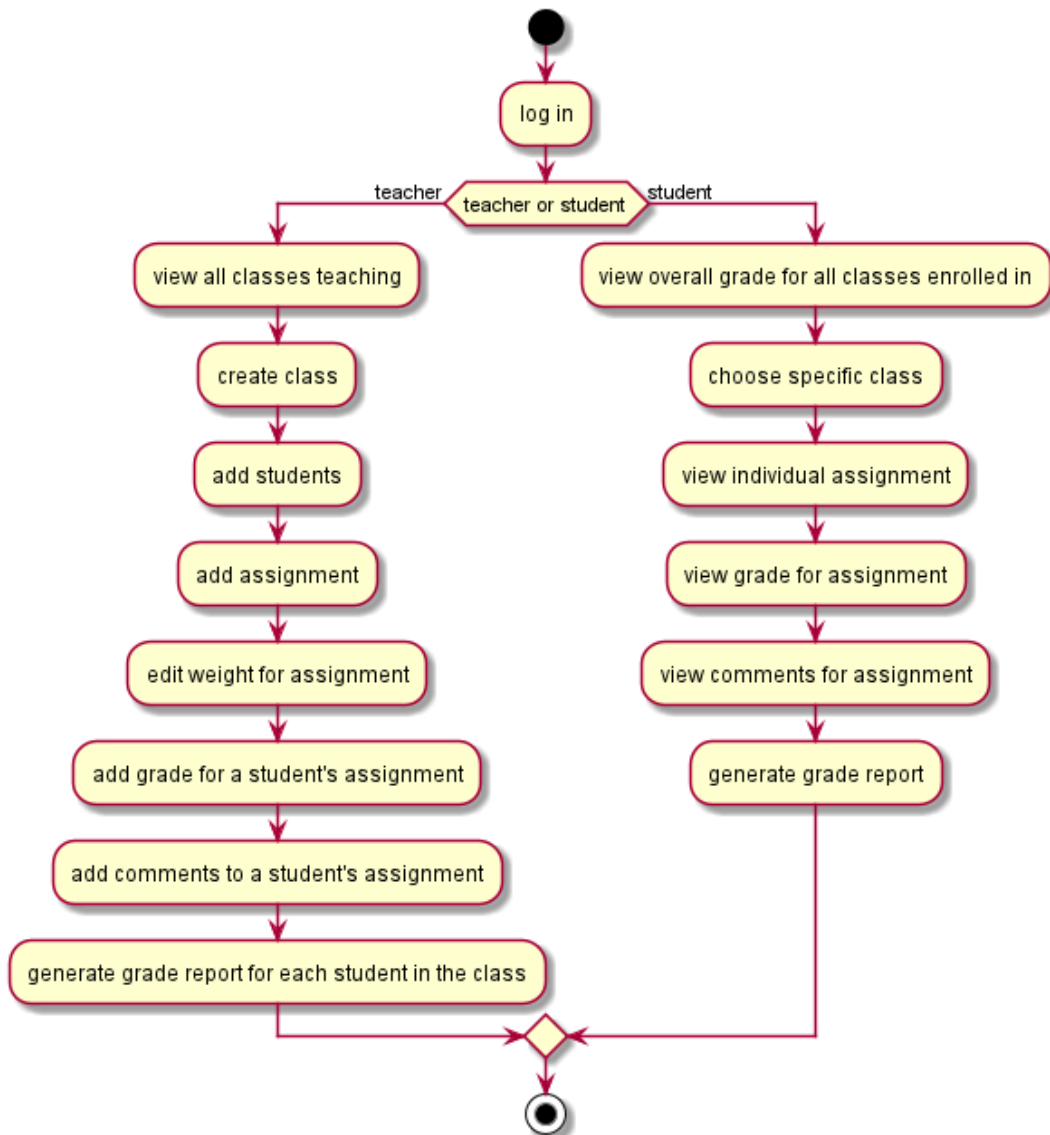
## Users and Tasks: Use Cases

This project will have two users: students and teachers. Teachers should be able to add various types of assignments in the system, upload grades and comments for those assignments, change the weight of assignments, and delete assignments. Students should be able to view these assignments, the grades, and their overall grade. We will outline the use cases for a student and teacher with the following UML use case diagrams.

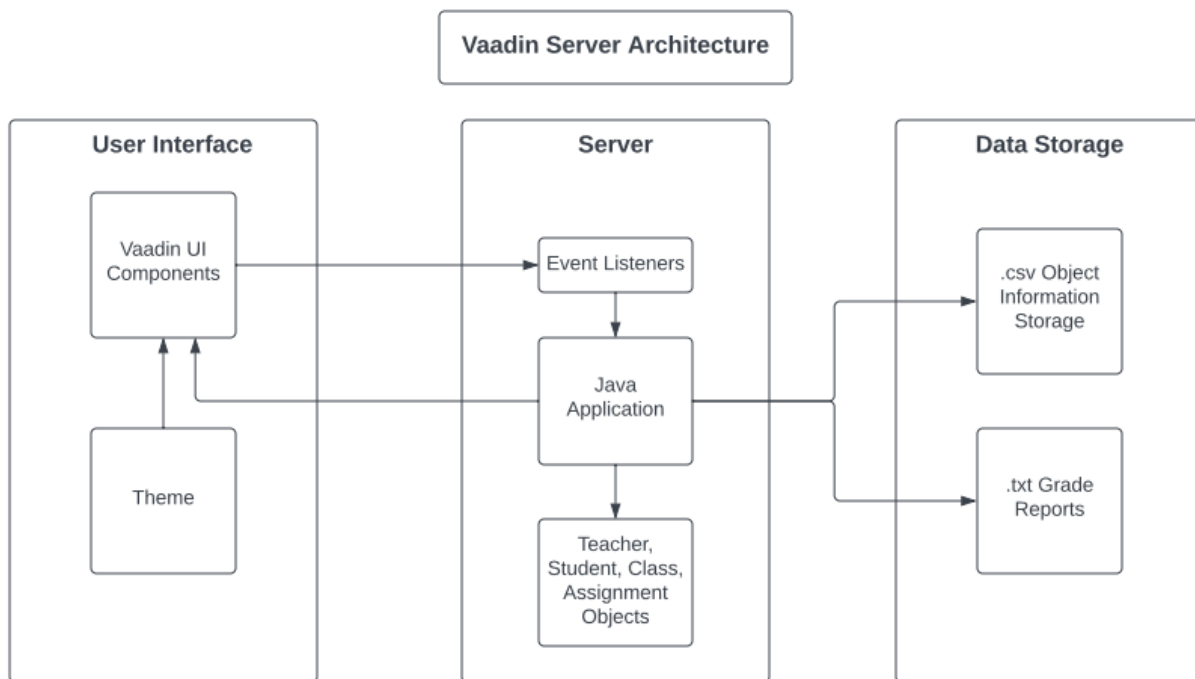


## UML Activity Diagram

The activity diagram below shows the operations that a student or teacher can perform.



## Architecture Diagram

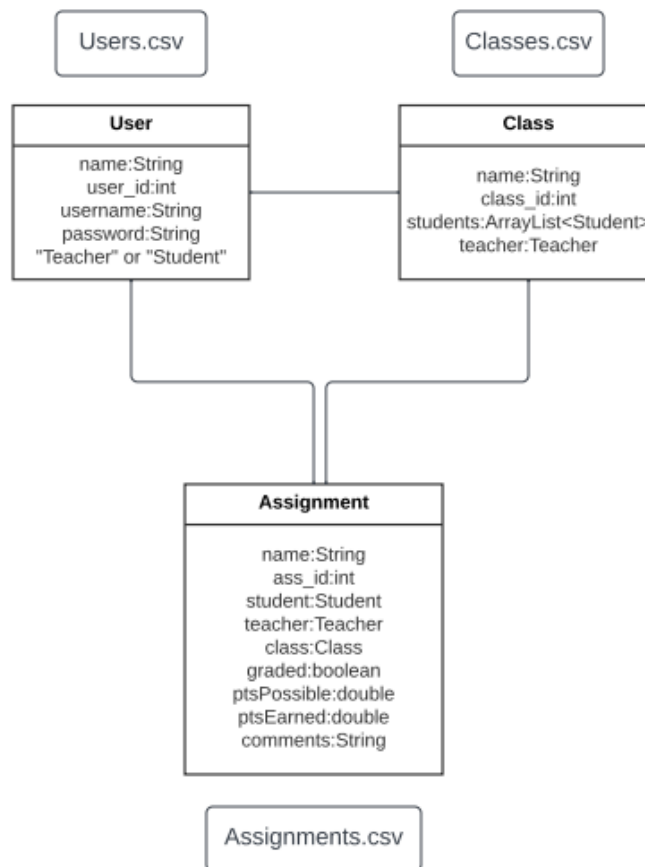


This architecture diagram describes how the components of our project interact with each other. Vaadin is an open-source GUI framework that allows developers to create simple user interfaces with java. Event listeners in the Java application are alerted to actions taken on the user interface and the display is updated accordingly. We are storing data with .csv and .txt files which can be seen in the data storage section of this diagram. The Java application will access these files directly, allowing it to save and load states from memory.

## Data Storage

Data will be stored as .csv files to provide simple management of saving and loading capabilities. Grade reports will be generated and stored as .txt files, but all system state data will be stored as .csv files in a directory within the server's repository. When the server is started it will create objects from the information stored in the files and define relationships where they are needed. There will be three separate .csv files storing information for Users (Teachers and Students), Classes, and Assignments.

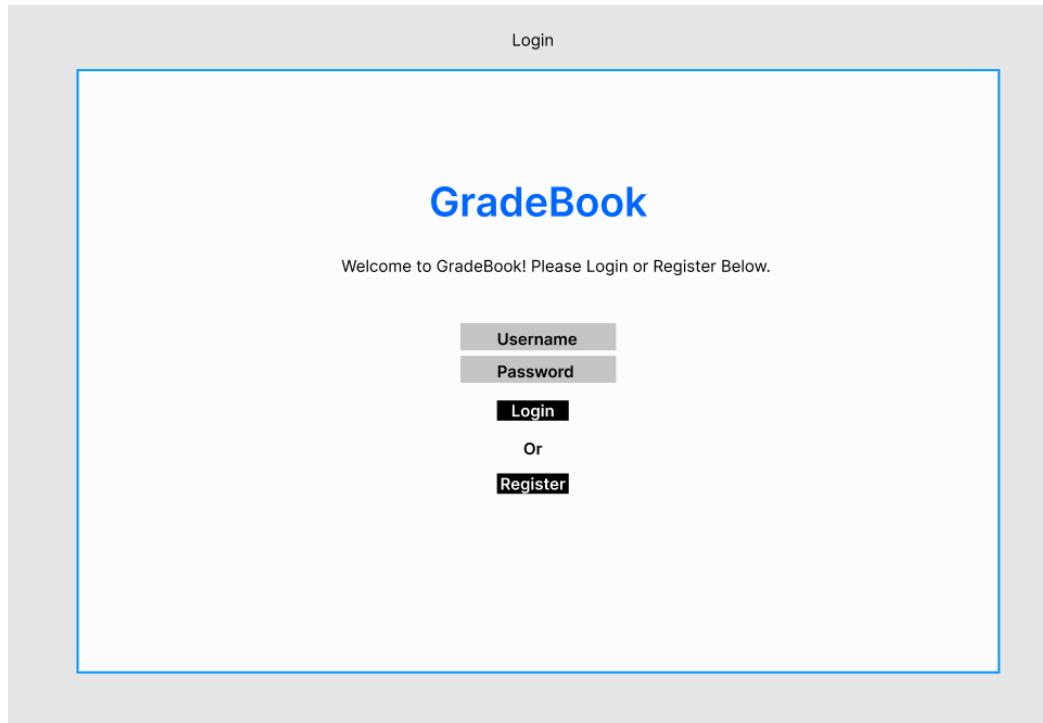
First the server will load the Teachers and Students found in the Users.csv file, creating objects for each person. Then it will load Classes, defining relationships detailing which teacher is responsible for each class and which students are enrolled. Lastly it will load the Assignments file, instantiate an object for each assignment, and define which class and student is responsible for the assignment.



The Users.csv file stores information about all Teachers and Students. These two are identified by the “Teacher” or “Student” keyword associated with their data. Once the users are instantiated, the Classes.csv file is read and the class objects are created. The relationships between teachers, students, and classes are stored in the Classes.csv files as a list of students in each class with the teacher separate. When the classes are instantiated their references are also added to the teachers and students related to the class. Once classes and users are created the Assignment objects are created. Each assignment stores which class, teacher, and student it is related to and these objects are updated with a reference to the newly created Assignment object. When this process is finished the server has a list of all users, classes, and assignments with relationships complete.

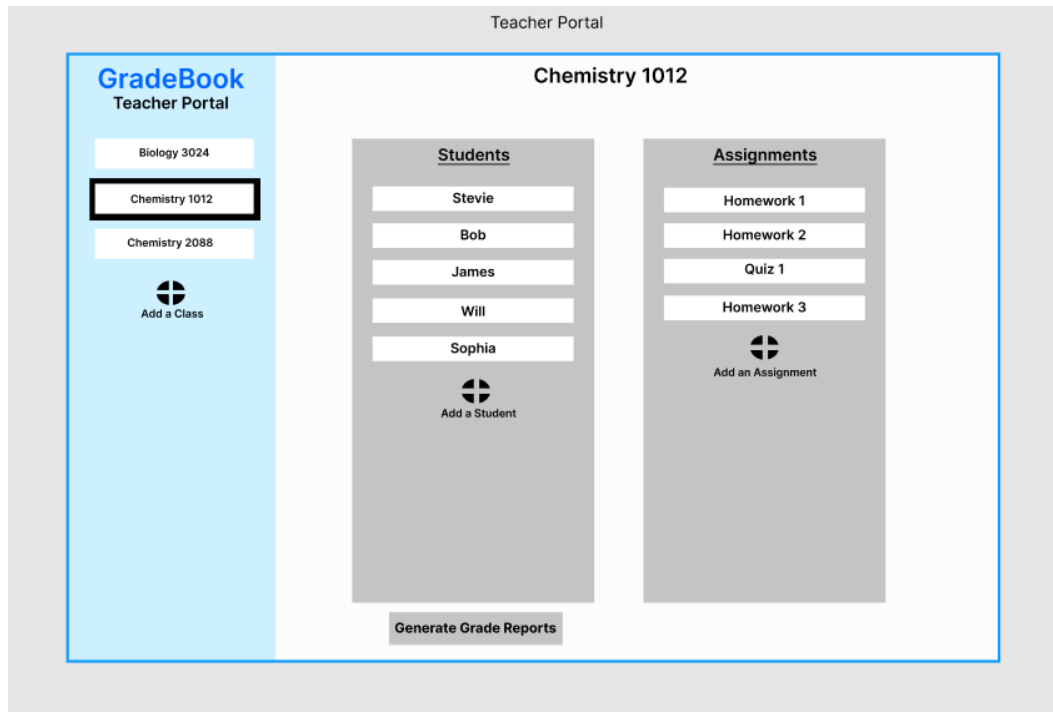
## UI Mockups

Login page, user can either enter credentials or register an account.

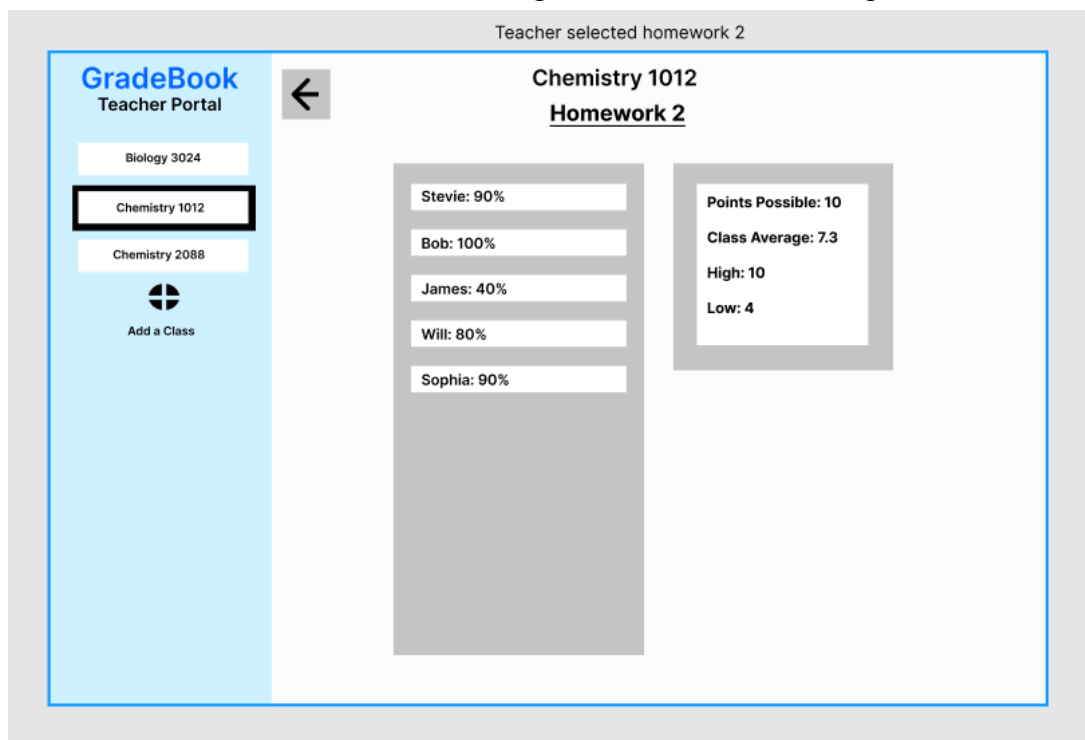


The image shows a UI mockup for a login page. It features a light gray background with a central white rectangular area outlined in blue. At the top of the gray area, the word "Login" is centered in a small, dark font. Inside the white area, the word "GradeBook" is centered in a large, bold, blue font. Below it, the text "Welcome to GradeBook! Please Login or Register Below." is centered in a small, dark font. Further down, there are two stacked input fields with gray borders; the top one is labeled "Username" and the bottom one is labeled "Password". Below these fields are two buttons: a black button with the word "Login" in white, and a white button with the word "Register" in black. Between these two buttons is the word "Or" in a small, dark font.

Teacher Portal: If the user is a teacher their portal will look like the image below, teachers are able to create classes, add students, create assignments, and generate grade reports.

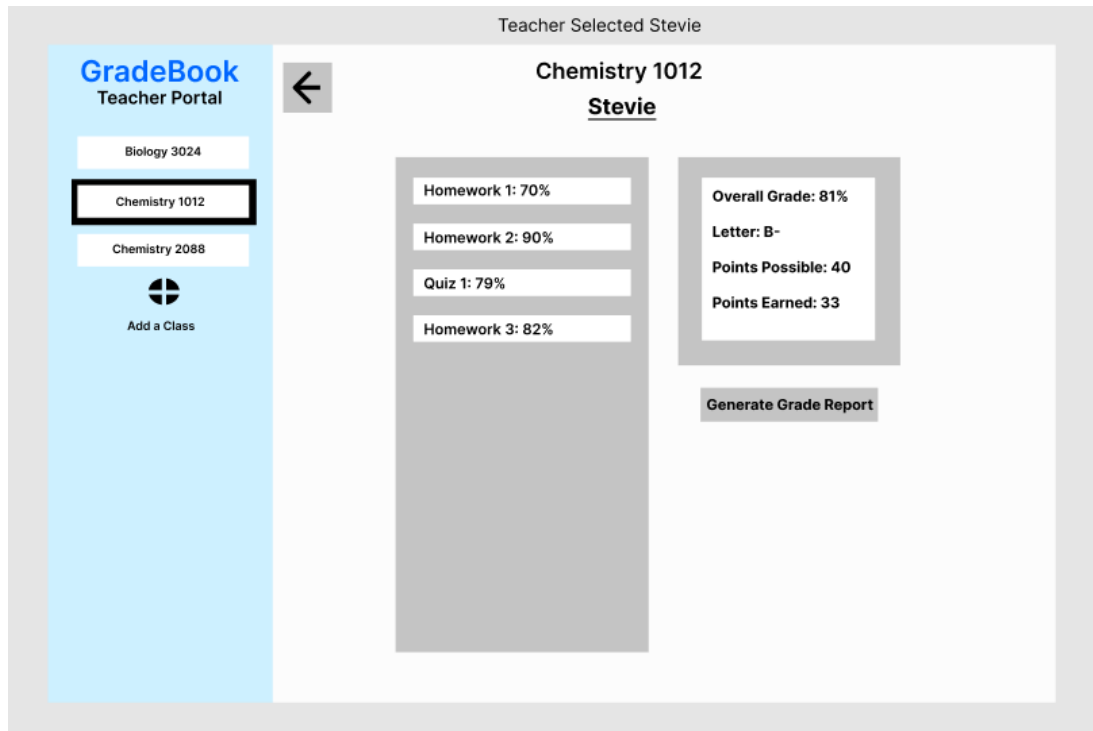


Teachers are able to select individual assignments to enter and edit grades for each student.

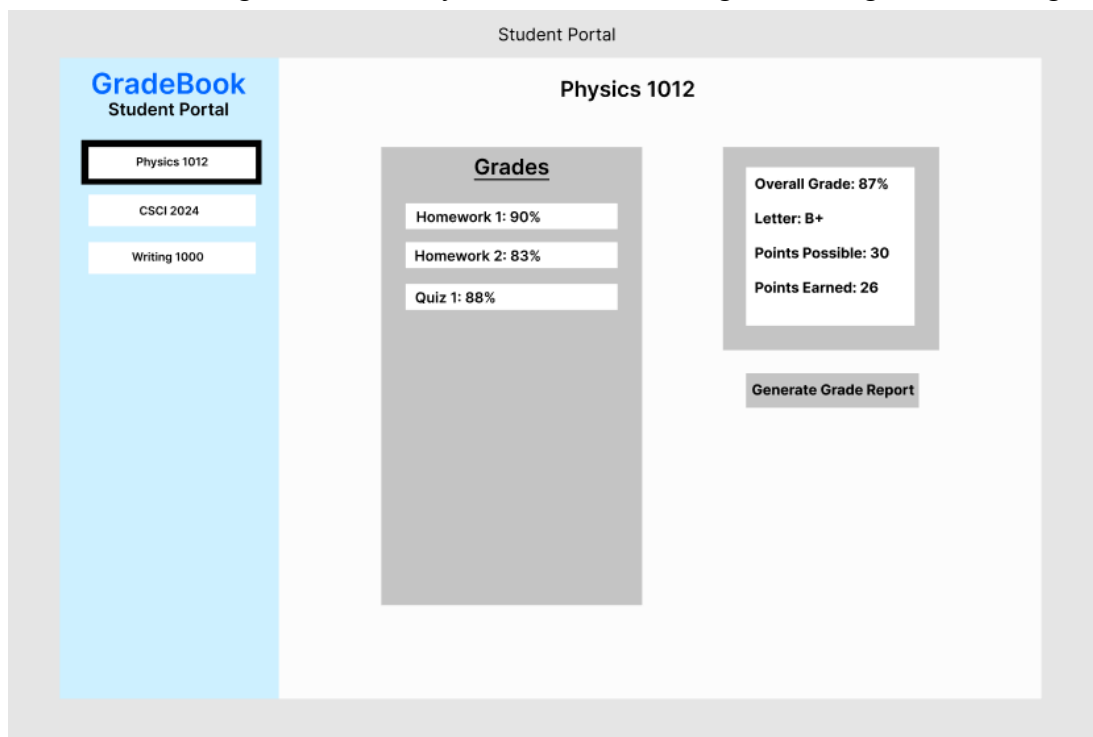


Teachers can also select individual students to see their grades in a class.

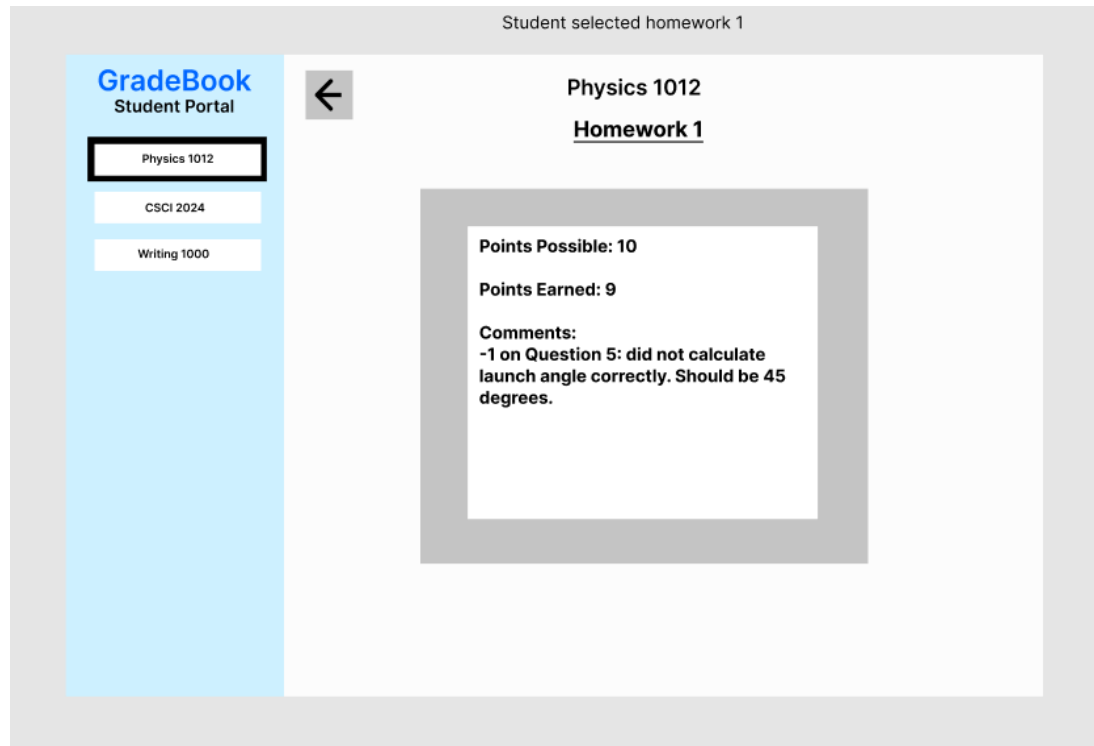




Student Portal: If the user is a student they will see the portal below. Students cannot create classes or add assignments but they can view their own grades and generate their grade reports.



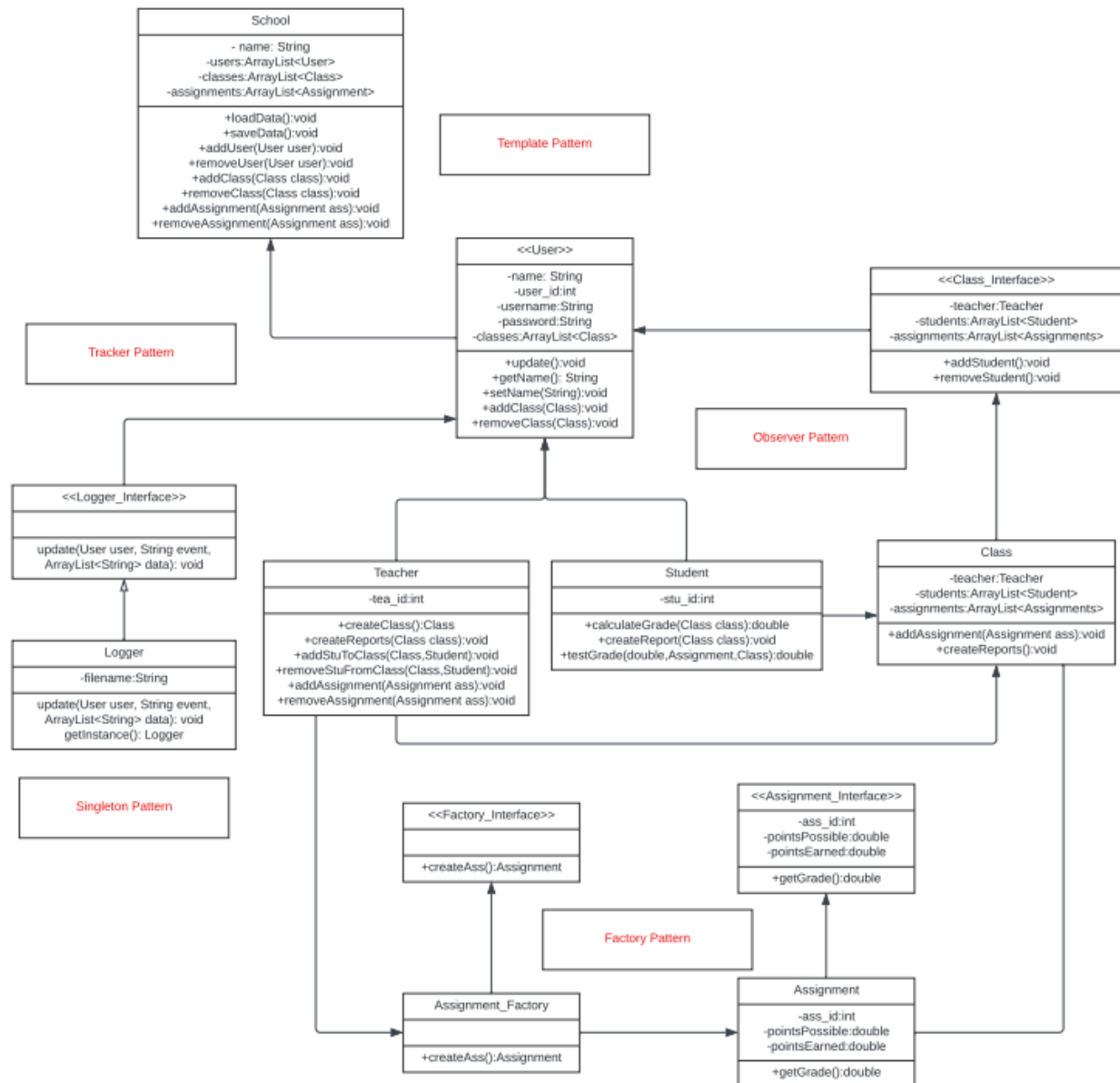
Students can select individual grades to see details and comments on their work from the teacher.



Created with Figma:

<https://www.figma.com/file/U1au0kbUsa6PqJJcrLclRU/Untitled?node-id=0%3A1>

## UML Class Diagram and Pattern Use



[https://lucid.app/lucidchart/b0ba452c-46df-4b30-b79f-27d96fa941c0/edit?invitationId=inv\\_4a6ed4b0-c859-4f3a-876e-66634cfc10e8](https://lucid.app/lucidchart/b0ba452c-46df-4b30-b79f-27d96fa941c0/edit?invitationId=inv_4a6ed4b0-c859-4f3a-876e-66634cfc10e8)

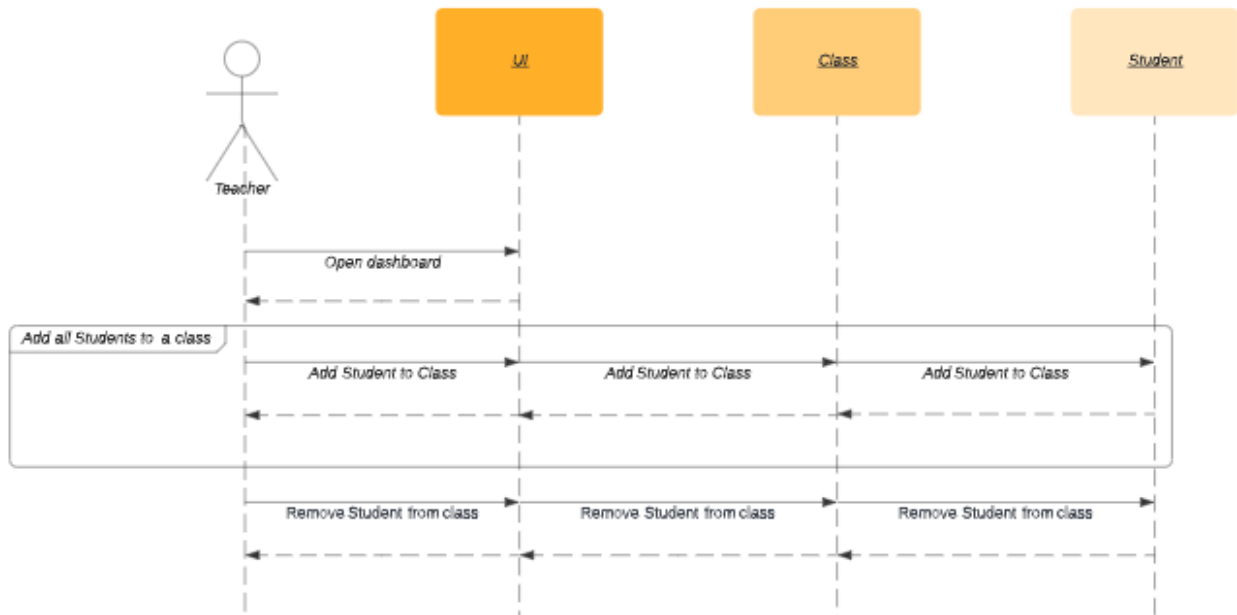
We are using five design patterns in our implementation of this project: Template, Tracker, Singleton, Observer, and Factory. The Template pattern is used with the abstract class `User` and concrete classes `Teacher` and `Student` to reduce repeated code and allow the `Student` and `Teacher` classes to add their own implementation for other methods. The Tracker pattern is used to log the actions of each user to keep track of how the grade data is changed and who is changing it. The singleton pattern is used with the logger to ensure that there is only one logger

active on the server. The Observer pattern is used with the students as the observers and the class as the subject. Student observers can be added and removed from each class, and updates to a class alert all students enrolled in that class. The Factory pattern is used to provide an interface to create assignments with. The teacher of a class will use the Assignment Factory to create assignments for that class.

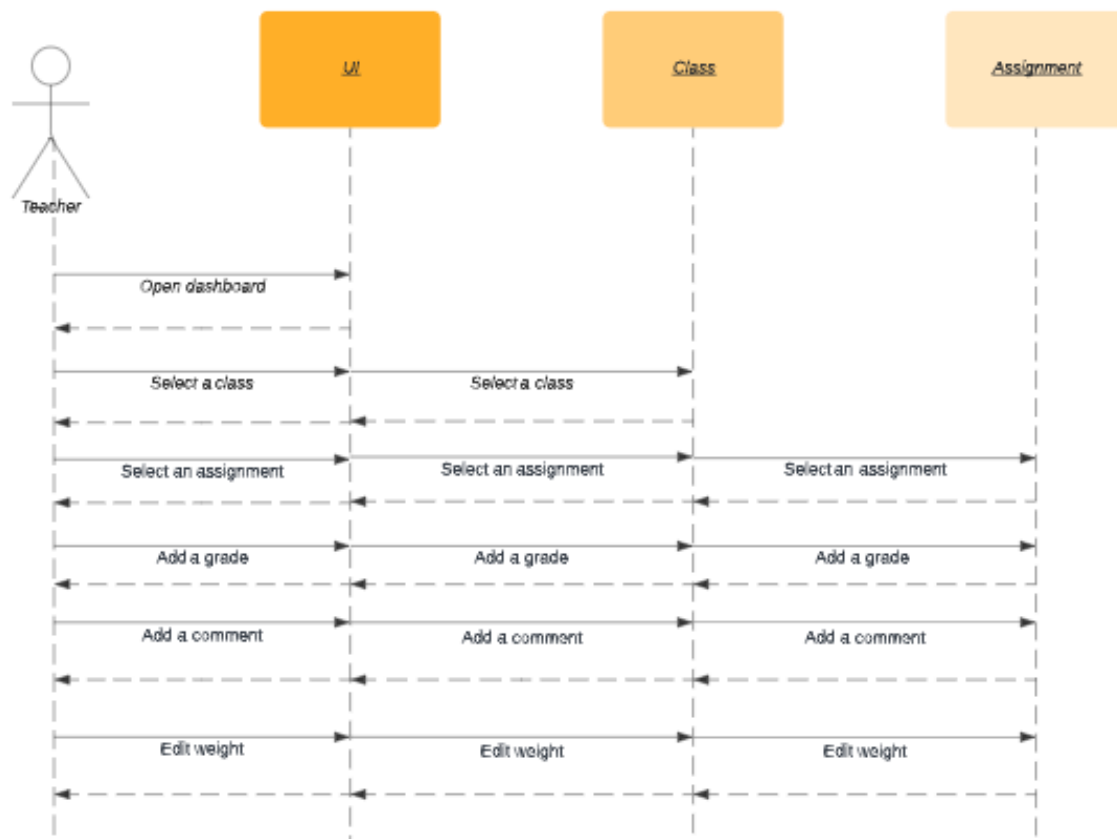
## User Interactions/UML Sequence Diagrams

Three interactions:

- Add and remove students to/from class



- Edit assignment



- Generate grade report

