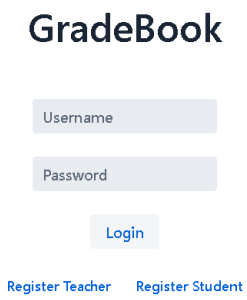
Will Loughlin and Sophia Chaltas
OOAD Project 7

Semester Project: Gradebook

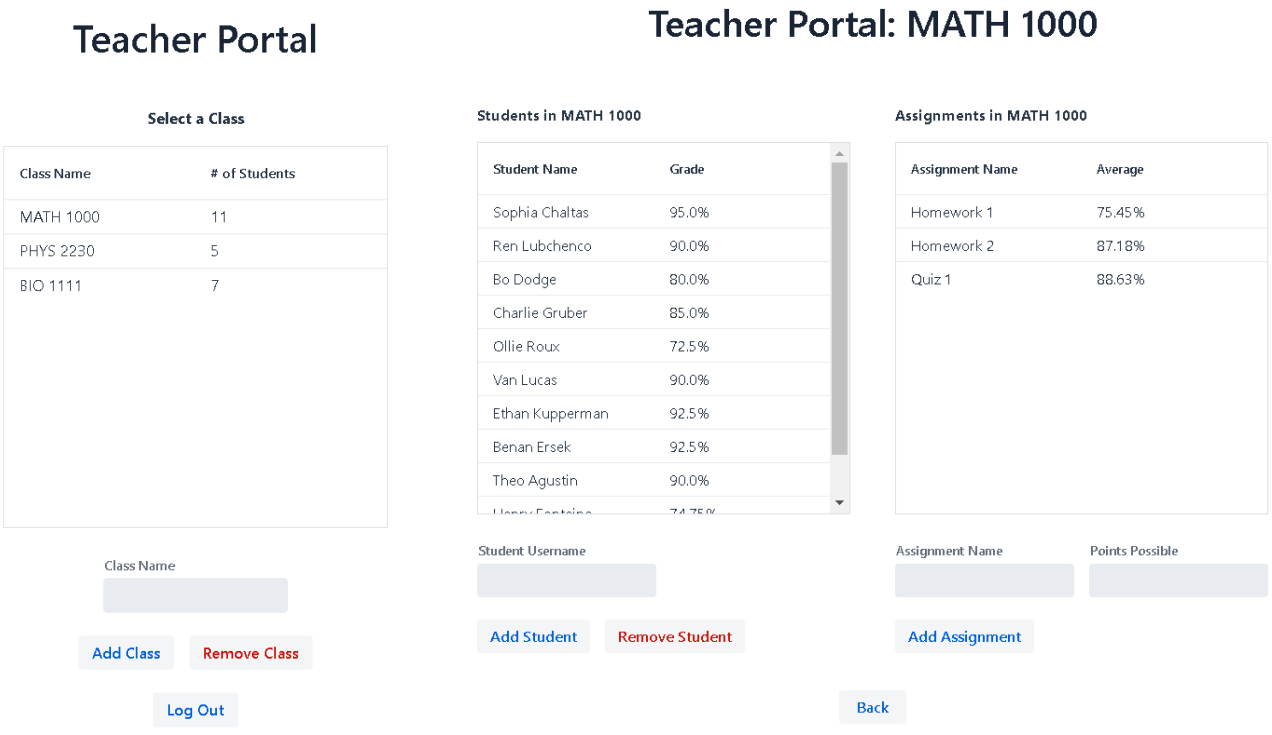
Final State of System Statement



Our project is called GradeBook and it is fully complete. We have a working login system that can be accessed as a teacher or a student. There is also a registration option for both teachers and students to create an account. If you try to login with the wrong credentials or create an account with a username already in use you are prompted with a notification telling you what went wrong.

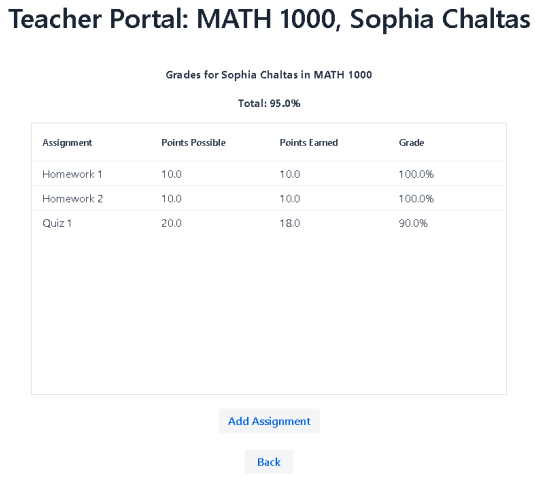
When a teacher logs in they are brought to their teacher portal where they can see a list of their current classes (with number of students listed), create classes, and remove classes. The teacher must type the exact class name to remove it, removing a class deletes all assignments and removes all students from the class. You cannot create a class with a name that is already taken and you cannot remove a class that does not belong to you or does not exist.

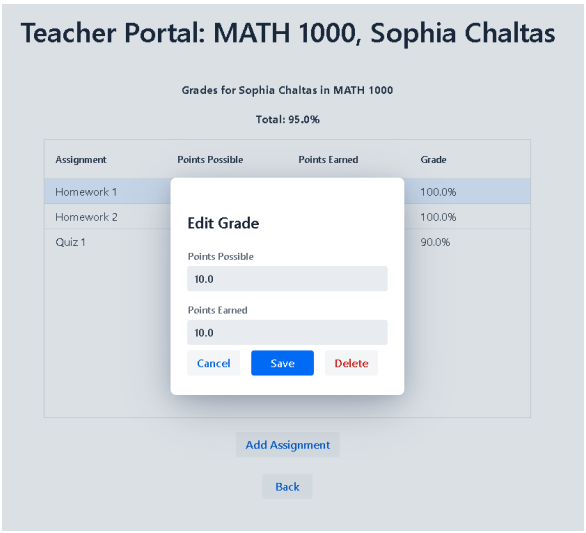
Selecting a class brings the teacher into the teacher-class view where they can see a list of students in the class with their individual grades and a list of assignments with their respective class averages. The teacher can add and remove students (this will delete assignments associated with that student), and add and remove assignments from this page. An assignment added to this



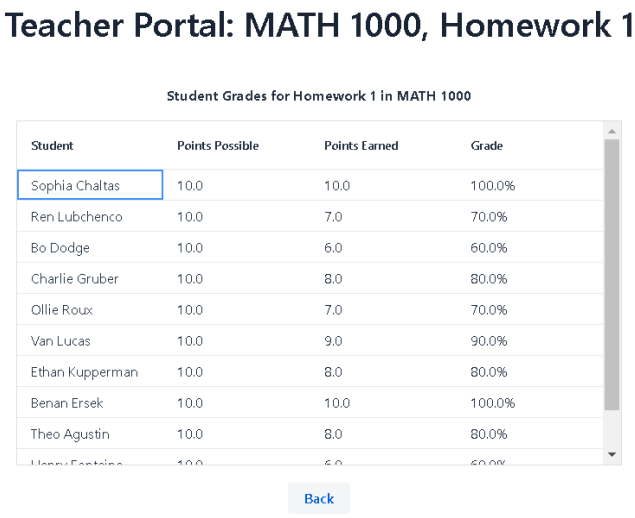
page will be added to all students in the class and will be initially graded as a zero until a grade is entered by the teacher. Each assignment is only displayed once on the assignments grid even though they have individual grades for each student in the class.

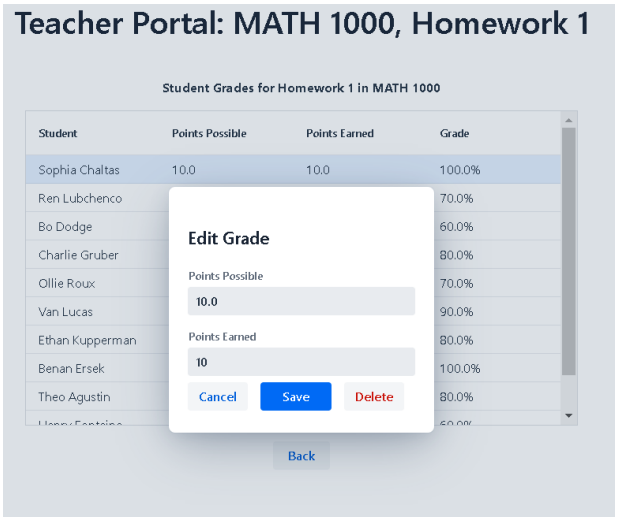
From the teacher-class view the teacher can select individual students to enter the teacher-class-student view. In this view the teacher can see every assignment for that student, edit assignments, add assignments, and remove assignments. Selecting a specific assignment will bring up a popup window allowing the teacher to edit or delete the assignment.



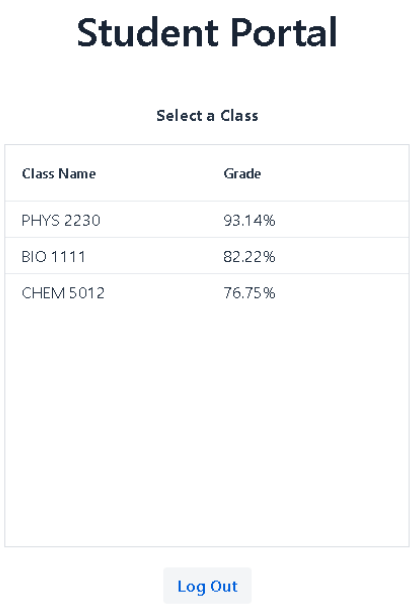


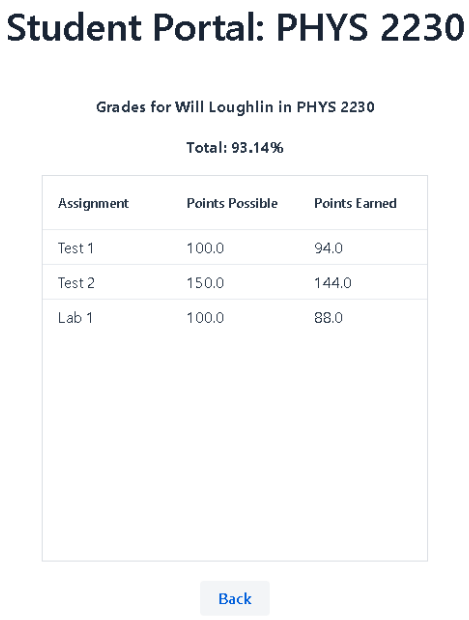
From the teacher-class view the teacher can also select individual assignments to see each student's grade. From this teacher-class-assignment view the teacher can edit and delete instances of the selected assignment.





Logging in as a student will bring you to the student view where students can see a list of their current classes and grades in each class. Students are not able to add or remove classes, they must be put into or removed from classes by a teacher. Selecting a class will bring the student to the student-class view where they can see their overall grade and each individual grade.



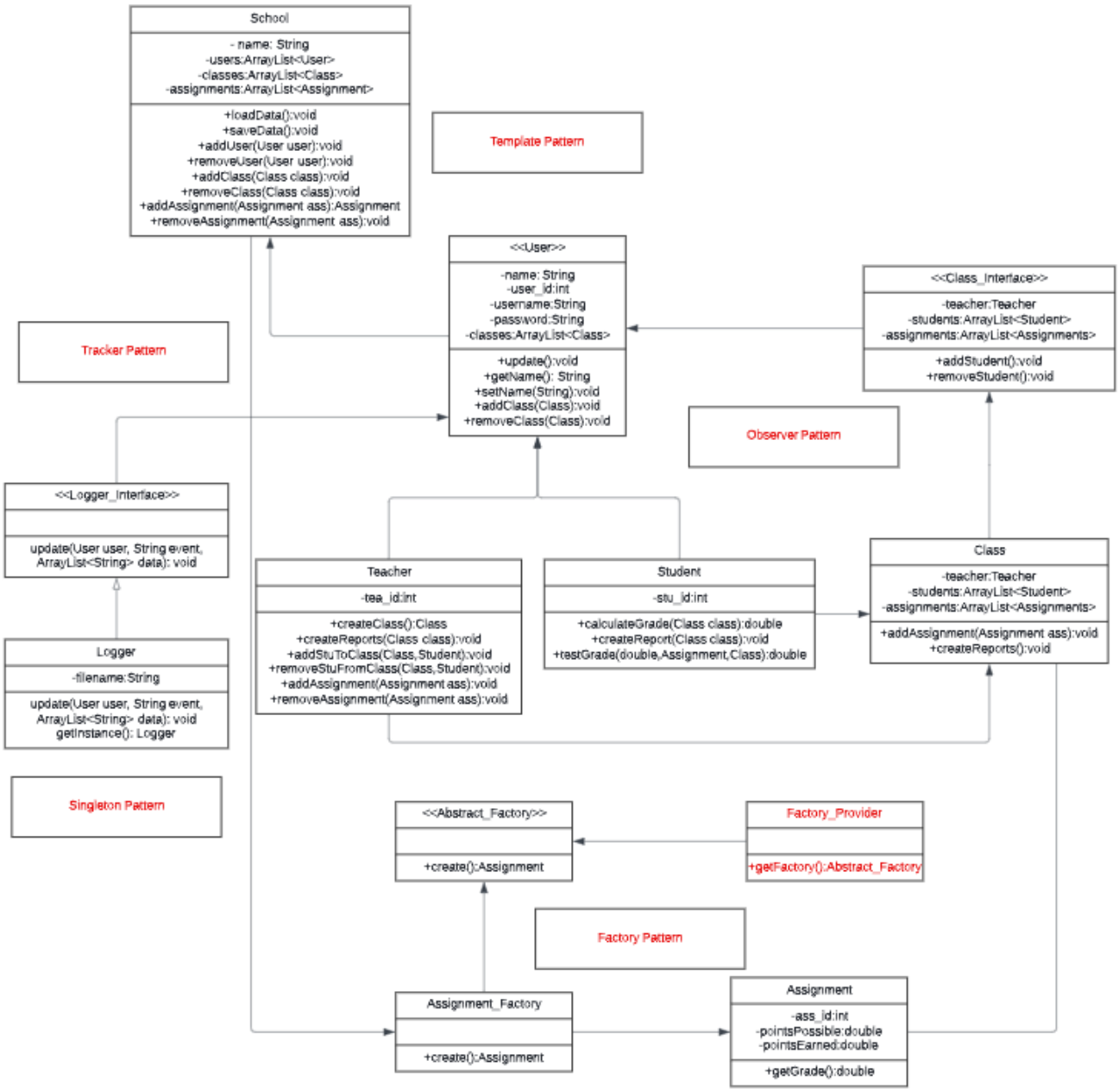


We use 5 design patterns in the backend of our project: Template, Tracker, Singleton, Abstract Factory, and Observer. The implementation of these patterns is discussed in the final section of this document. For data storage we use three csv files: Users.csv, Classes.csv, and Assignments.csv. These files store all system state data and allow a school object to be saved and loaded with the same state. The only functionality we didn't implement was a "generate grade report" ability because these reports would be stored on the server and would not be easily accessible to any regular user. Aside from the grade report functionality we completed everything that we proposed in our initial project proposal. Our project can be found on github with the link: https://github.com/WillLoughlin/GradeBook and can be run with the command 'mvnw' while in a cloned version of the repository.

There is example information stored on the server. To use an example teacher login, use the username 'Nate Poulter' and the password 'password'. To use an example student login, use the login 'Will Loughlin' and the password 'password'. You can also create your own teachers, students, classes, and assignments and populate the classes however you would like.

Final Class Diagram and Comparison Statement

Our UML diagram barely changed from project 5 to 6, and also did not change after completing project 7 since almost all of the backend code was completed in project 6. Below is the diagram that we implemented in project 6.



https://lucid.app/lucidchart/b0ba452c-46df-4b30-b79f-27d96fa941c0/edit?invitationId=inv_4a6ed4b0-c859-4f3a-876e-66634cfc10e8&page=0_0#

Third-Party code vs. Original code Statement

We use a significant amount of third party code for the server portion of our project. Specifically we use a Maven server that implements Vaadin for user interaction. The code that we wrote for this project is all contained in a single folder that can be found at the path:

Gradebook/src/main/java/com/example/application/views/main

Everything not contained in this folder is third-party code that is used to facilitate user-server interaction. This starter code for the server that we used is from a Vaadin YouTube tutorial that can be found here: https://www.youtube.com/watch?v=qUBt8k4pQgQ&ab_channel=vaadinofficial. The code can be found on Github here: https://github.com/marcushellberg/spring-boot-todo.

Statement on the OOAD process for your overall Semester Project

1. The Role of Patterns

In the formation of this project, we started thinking about the roles of patterns as we brainstormed a topic. We thought of ideas that had clear class separation and would be easy to implement with some design patterns. A grading application clearly had students, teachers, and assignments and we could already conceptualize how we could implement some patterns. We found the easiest pattern to think of was the template, as we would need to create students and teachers. Next we thought about observer, tracker, and singleton since they all go hand in hand. Then it was also logical to identify that we could use a factory pattern for assignment creation. Because the patterns were included in our brainstorming phase, it was easy to start the writeup for project 5.

2. Identify Relationships

Once we had identified these patterns we started thinking about which classes they would relate to. Obviously we knew that template would apply to the user class because we had two different types of users. Students and Teachers have similar needs but with slightly different implementations. The template pattern is effective here because it is helpful for them to have the same function names but they cannot be implemented in the same way. The tracker pattern is implemented with the logger class that tracks actions taken on the school, which is helpful to understand what has been happening to the school with multiple users. We use the singleton

pattern along with the tracker to ensure that there is only one tracker active at a time. The observer pattern is used to model the relationship between students and classes. Students are added to classes as observers that are updated with changes to the class. The abstract factory pattern is used to create assignments in our project. Assignment creation is very important as it stores the main information used in our GradeBook.

3. Thinking in Patterns Approach

This approach has 3 steps: 1: identify the patterns, 2: analyze and apply the patterns, 3: add detail. Since step 1 was completed in project 5 when we had already identified the patterns, our job in project 6 was to do step 2 and analyze them. We had thought that the factory for assignment creation would be related to the teacher class, but during the coding of project 6, it turns out that it was better suited to be used in the school class. We made this change to the school class because it would allow the school to access the teacher and student that would need to be linked to the assignment. Step 3 was implemented all throughout the completion of project 7 as we added lots of detail. This time was used to create the front end application and implement the functionality we had proposed.