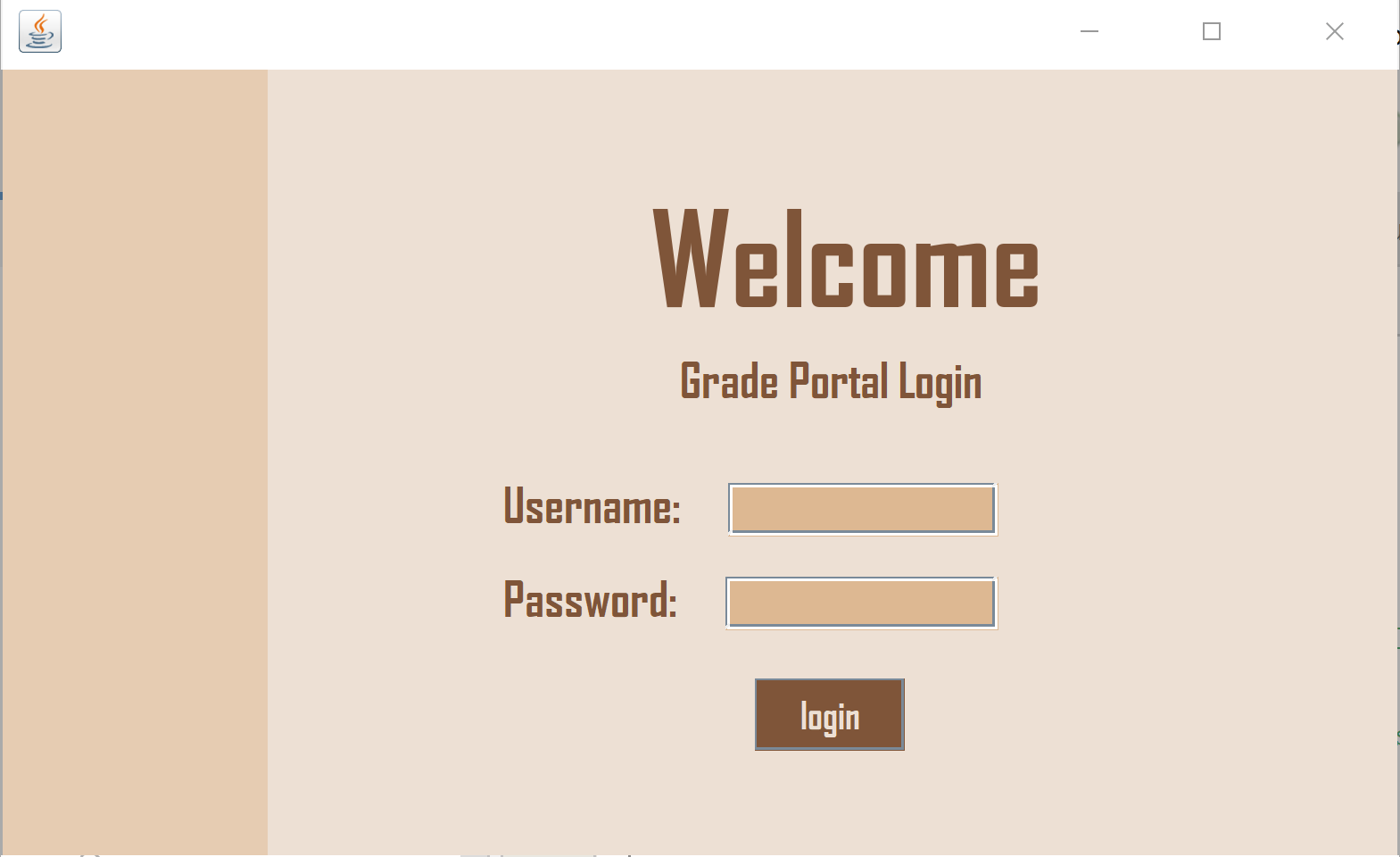
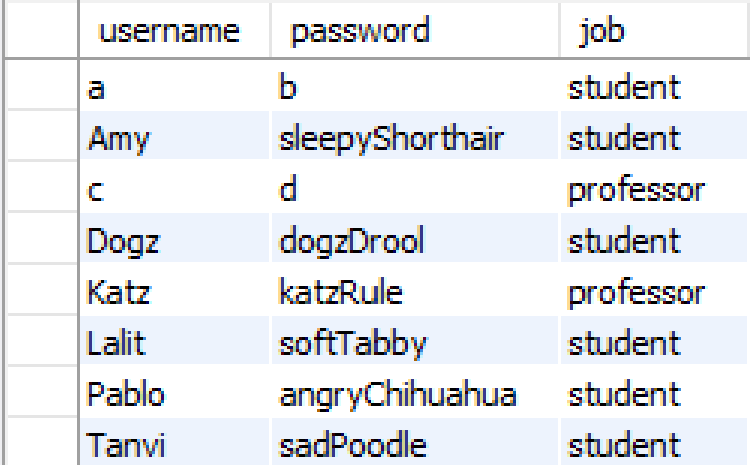
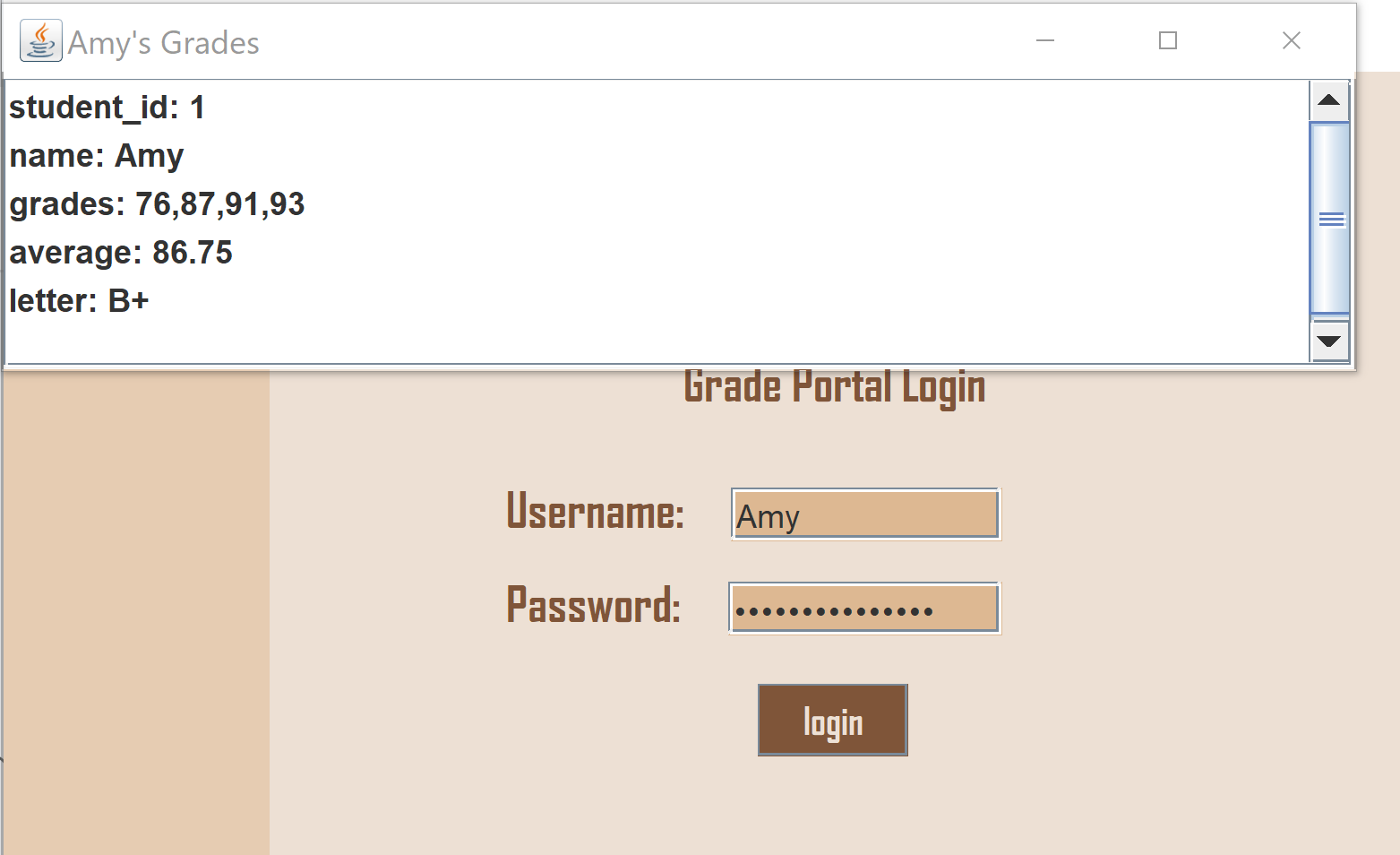
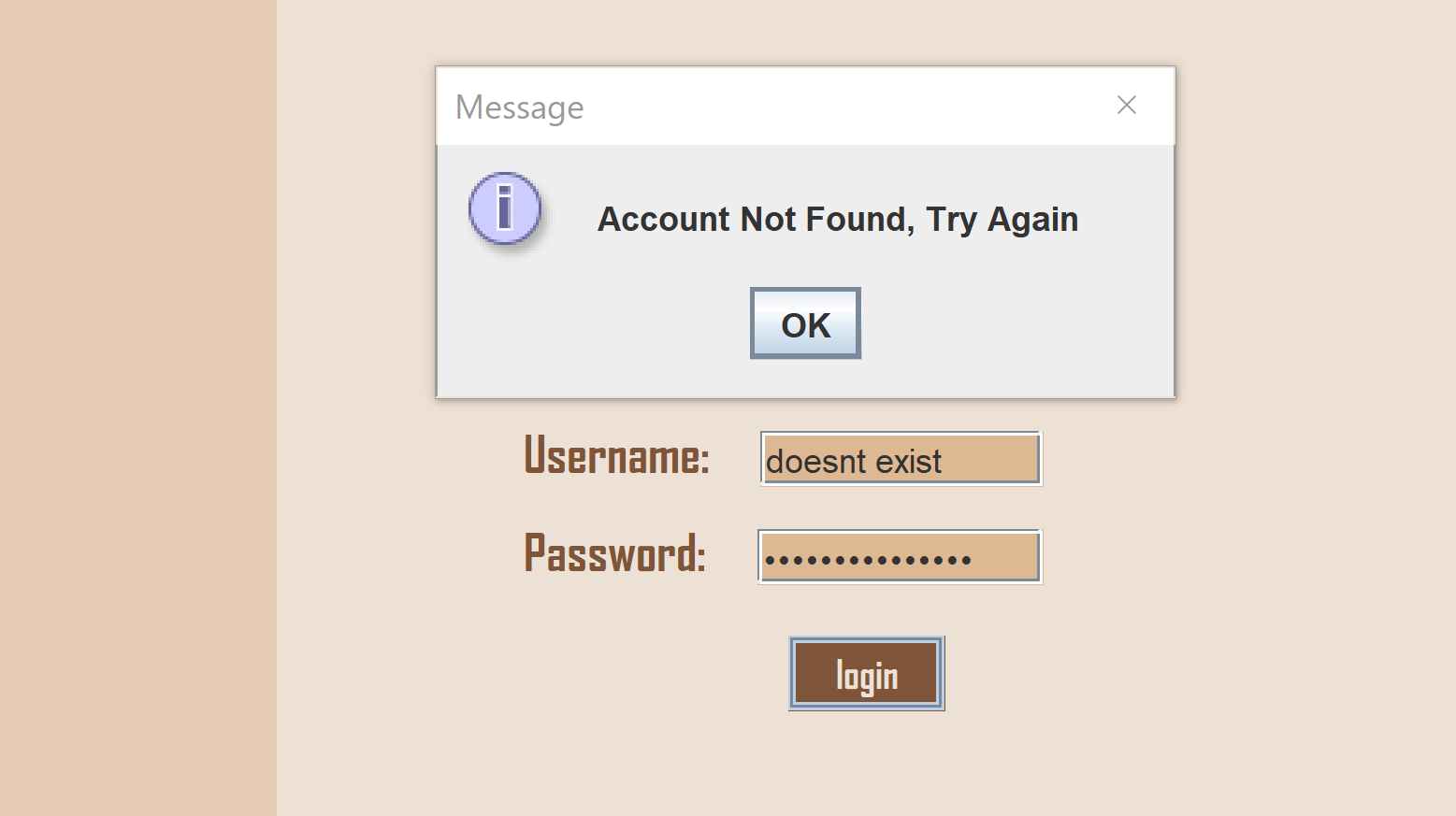
My Final Project implements File IO, Thread Concurrency, and SQL/JDBC. I used Netbeans and MySQL Workbench

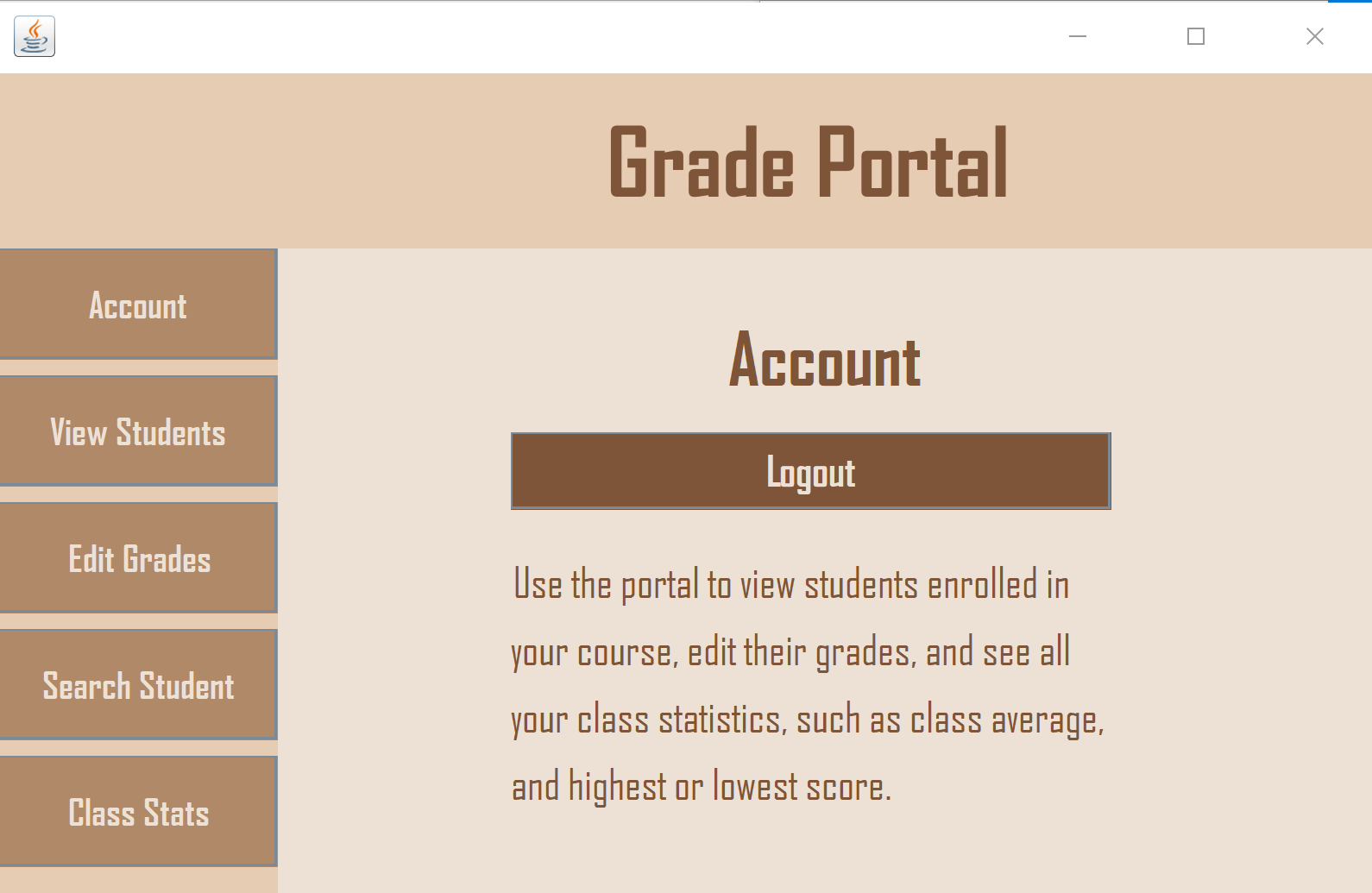
It is a student grade portal that supports both student and professor log-ins. If a student logs in, it will show them their grades. If a professor logs in, it will open the portal where they can view all student data, edit student grades, search for specific students, or see current class statistics.

Opening Page: User Table:

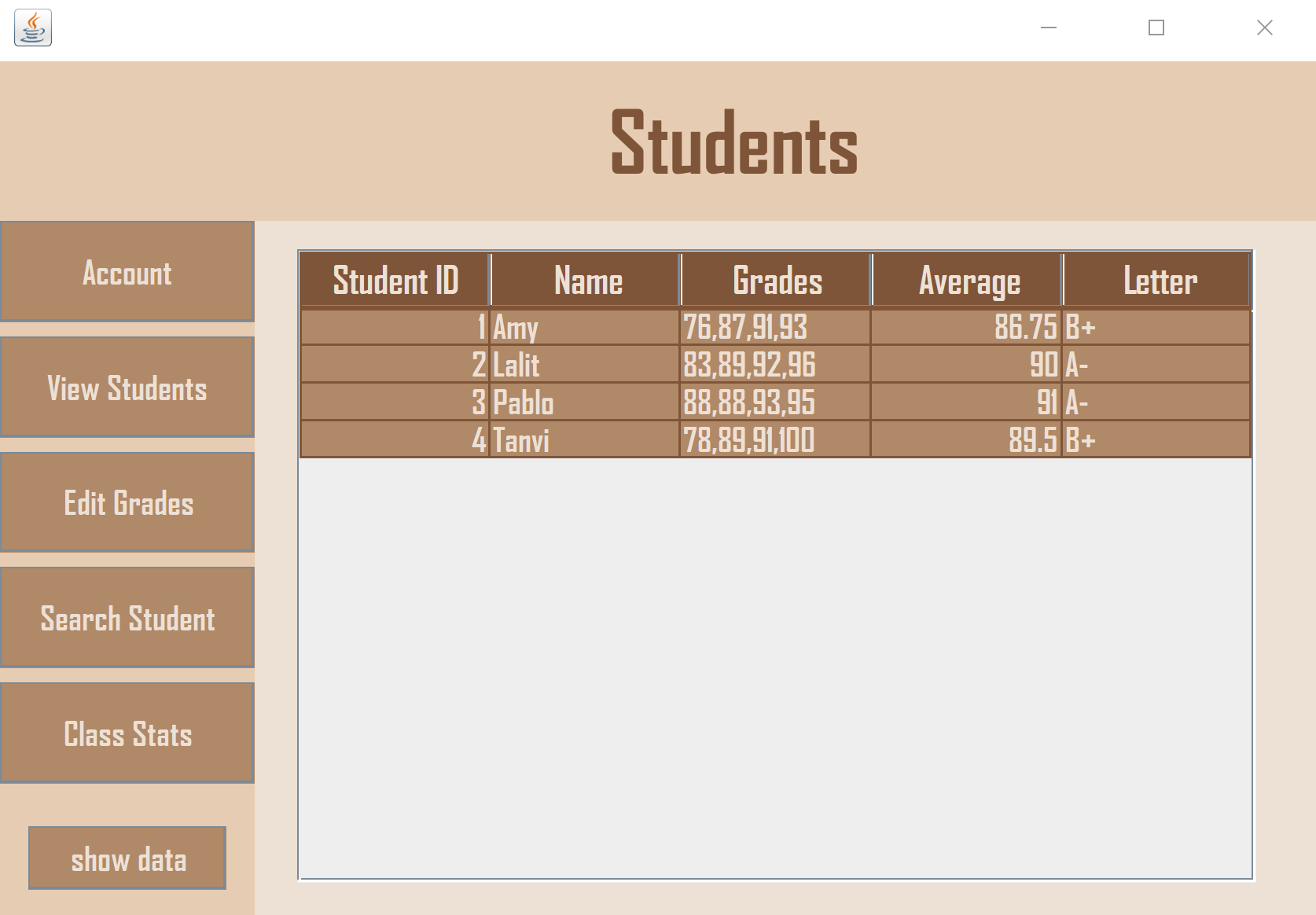


If auth fails (user/pass doesn’t match user table): If student logs in:

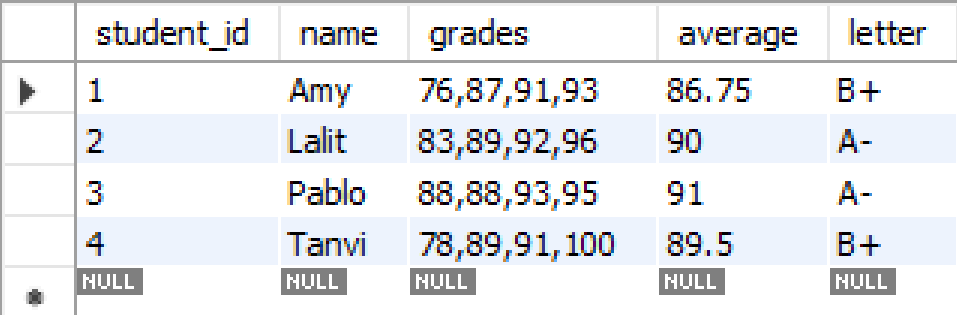


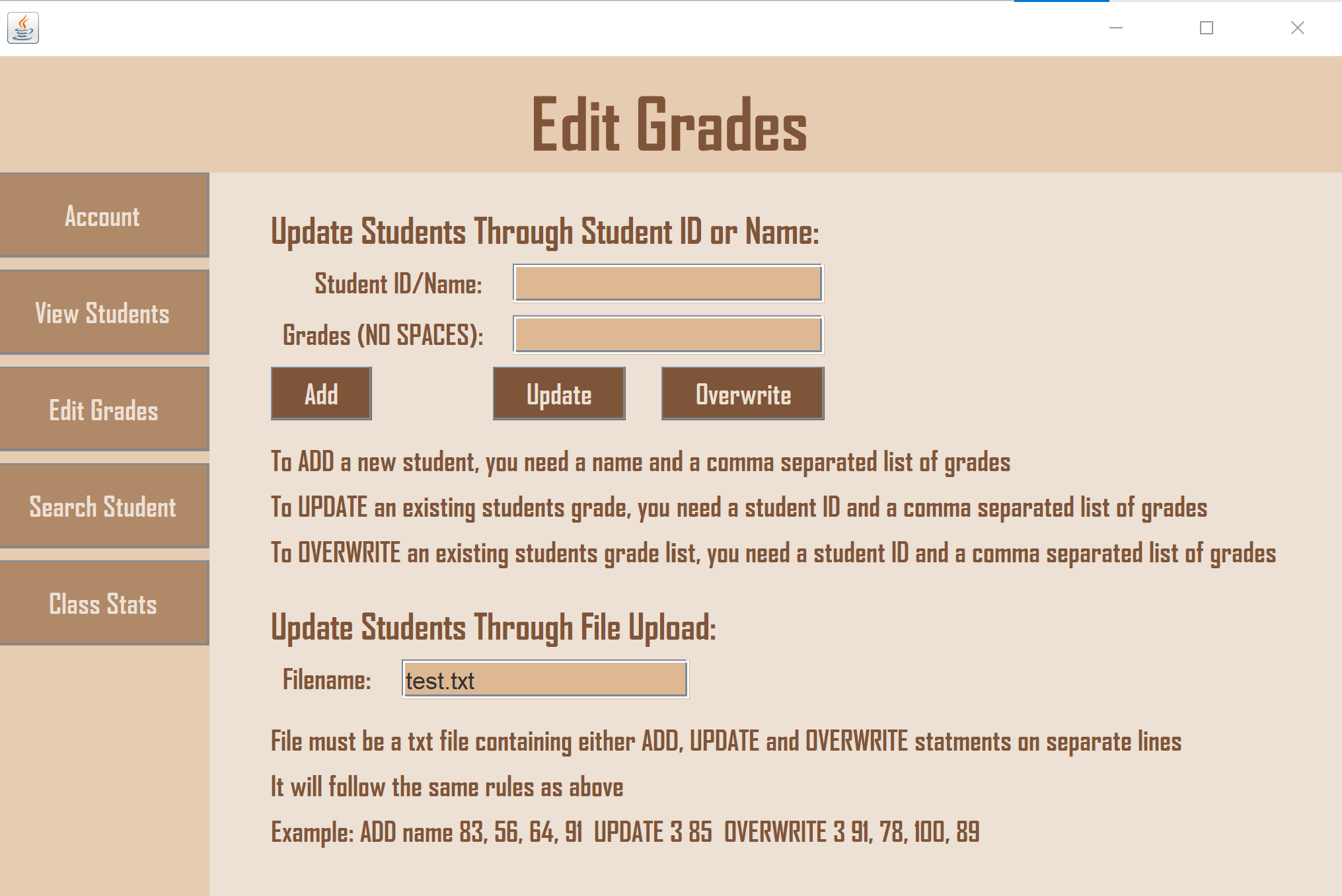


This is what opens if a professor logs in. They can log back out with the logout button or try some of the other features with the buttons on the side panel.

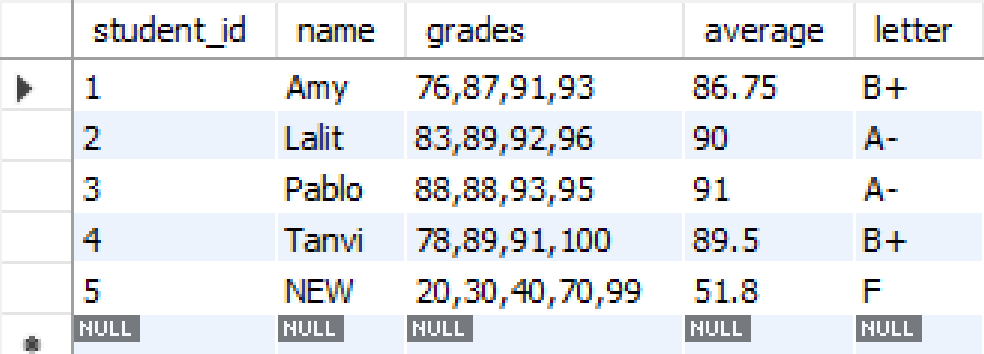
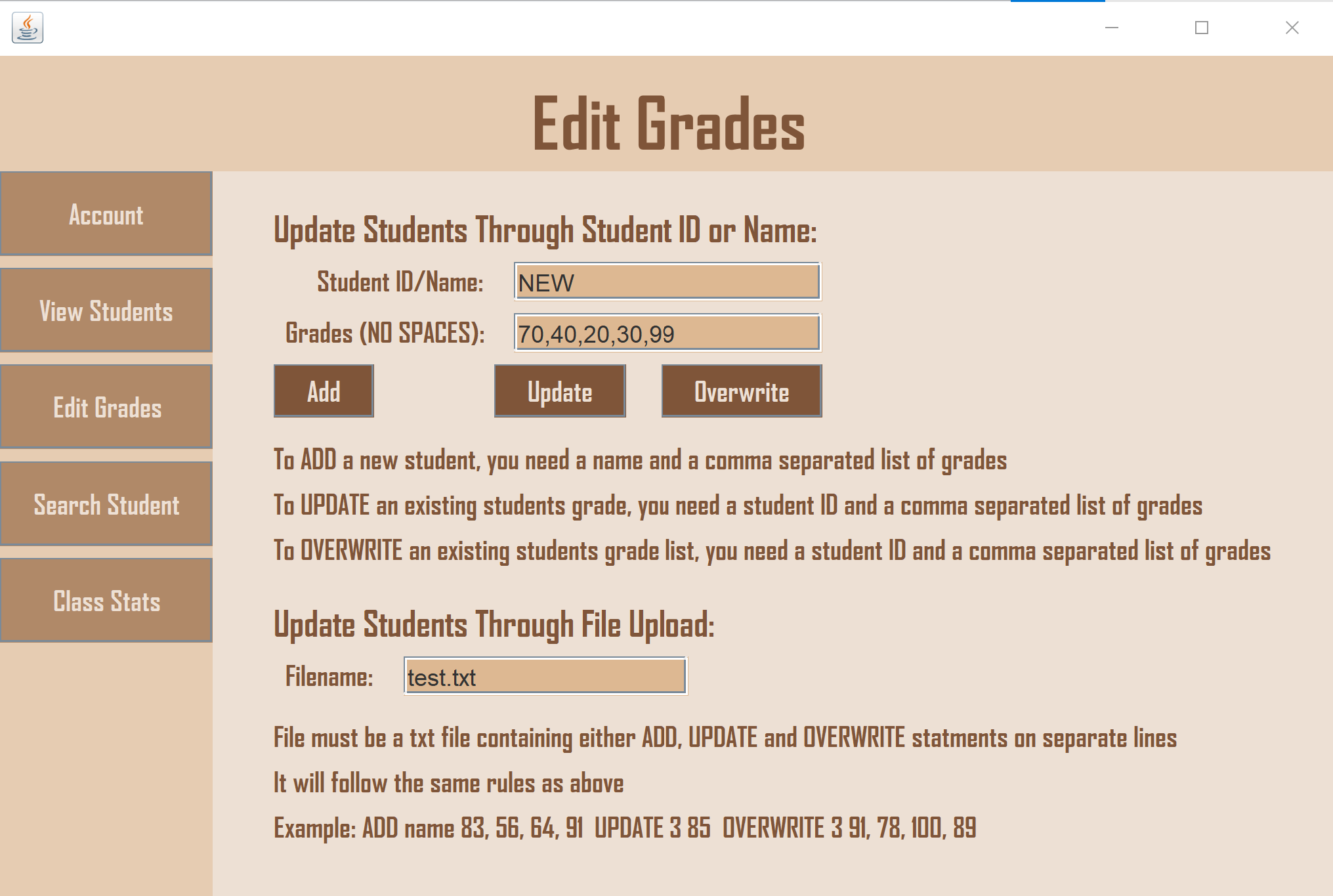


If you hit View Students, it will take you to an empty table. If you hit the “show data” button on the side panel, it will pull up all current student data. Here’s the student table below.



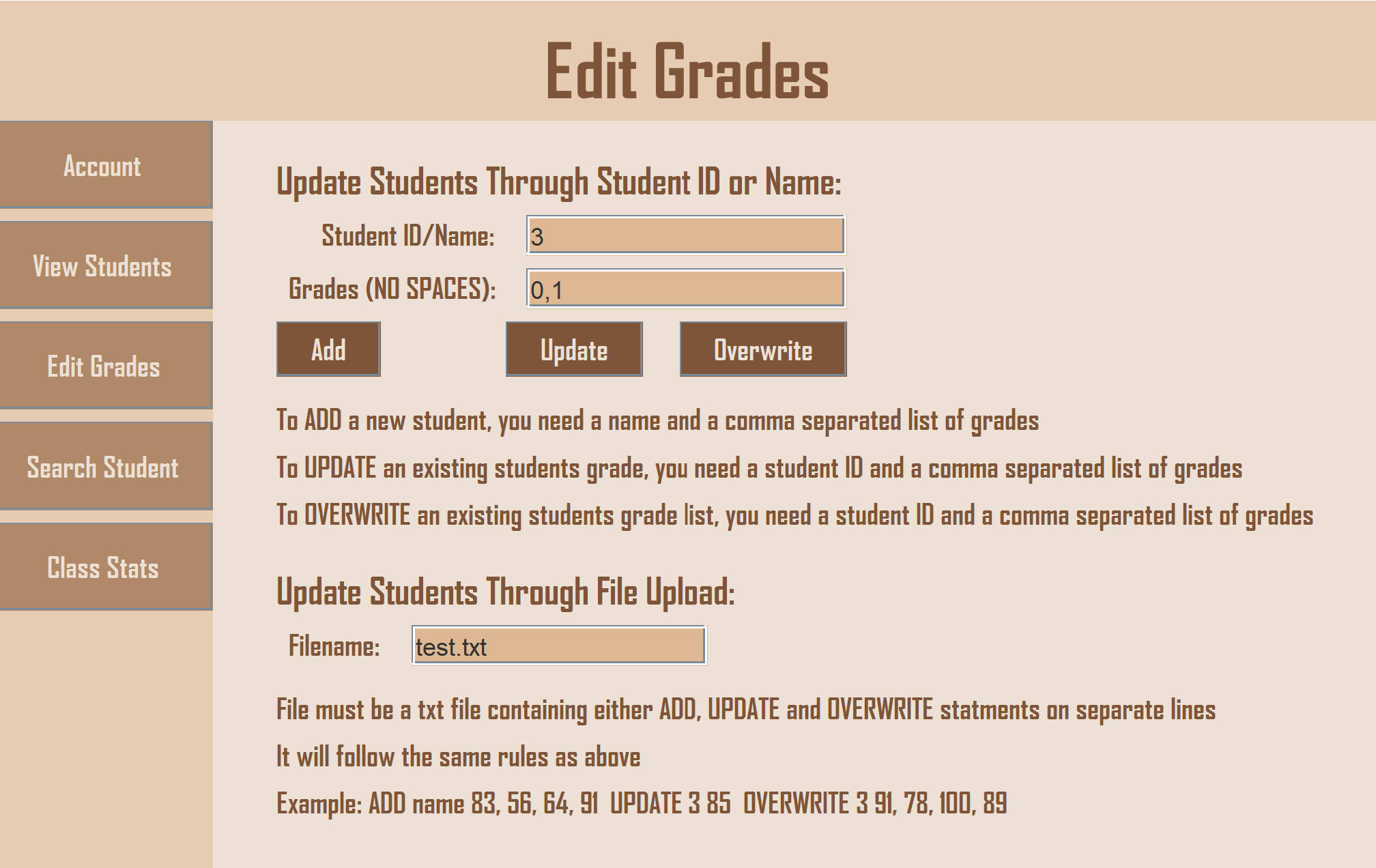


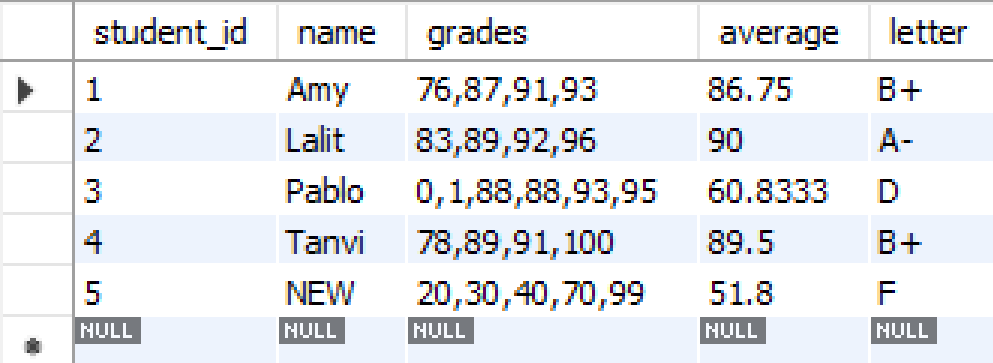
On the edit grades page, there are multiple ways to edit student grades. You can edit through direct user input or through a text file. ADDing a new student record requires a name and comma separated list of grades with NO SPACES. To UPDATE or OVERWRITE a student’s grades, it needs an existing student’s ID and a list of grades.



Example of ADD command

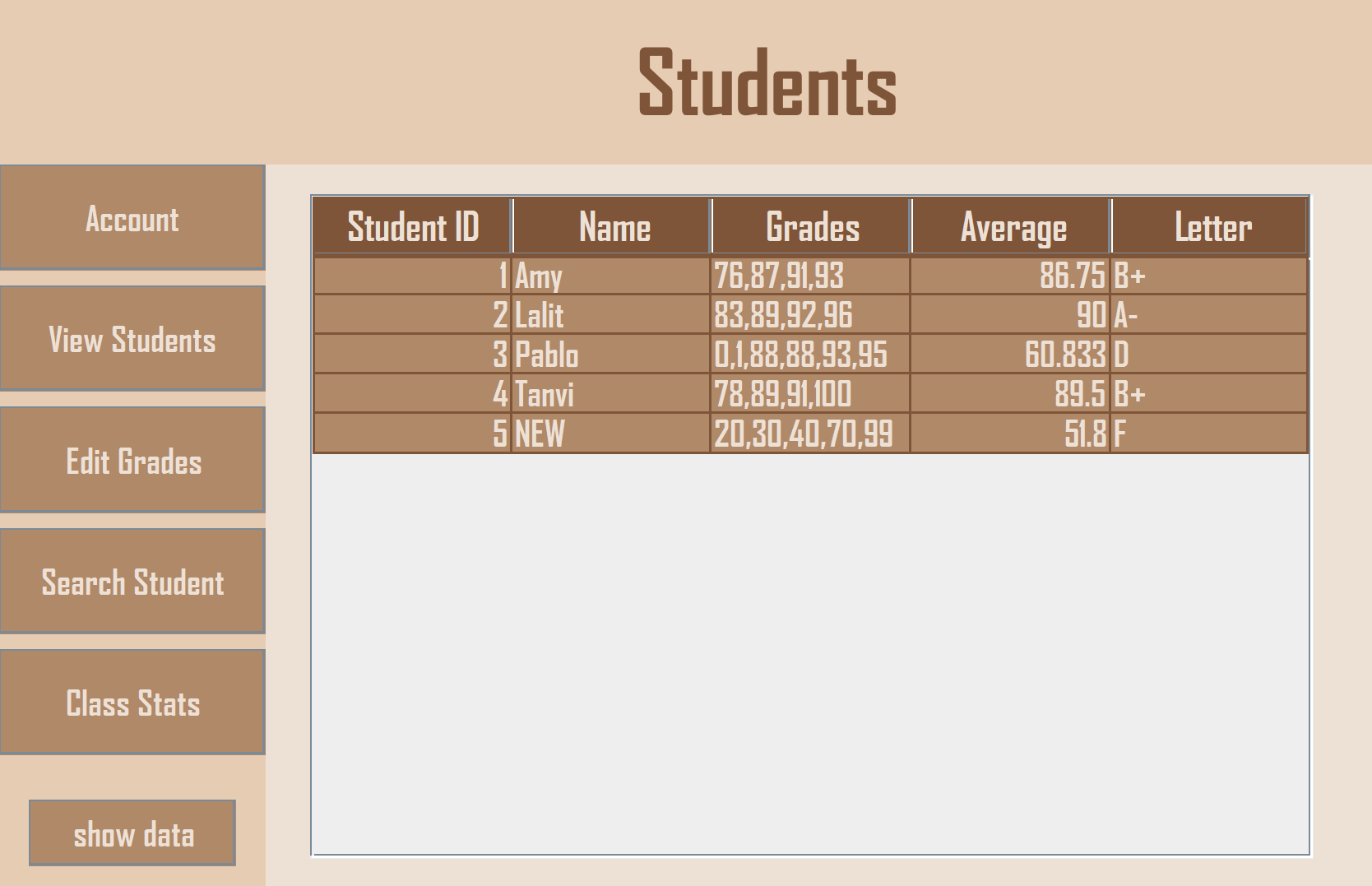
The change is reflected in the SQL table



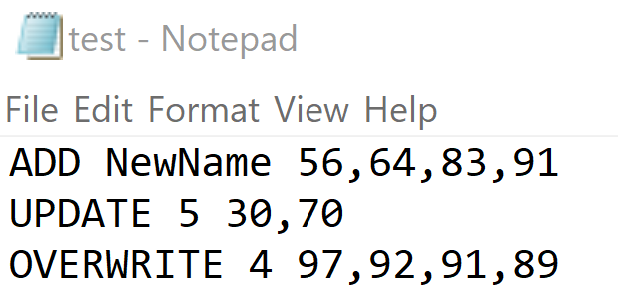
Example of UPDATE command on student 3

The change is reflected in the SQL table

It also updated the average and letter grade accordingly

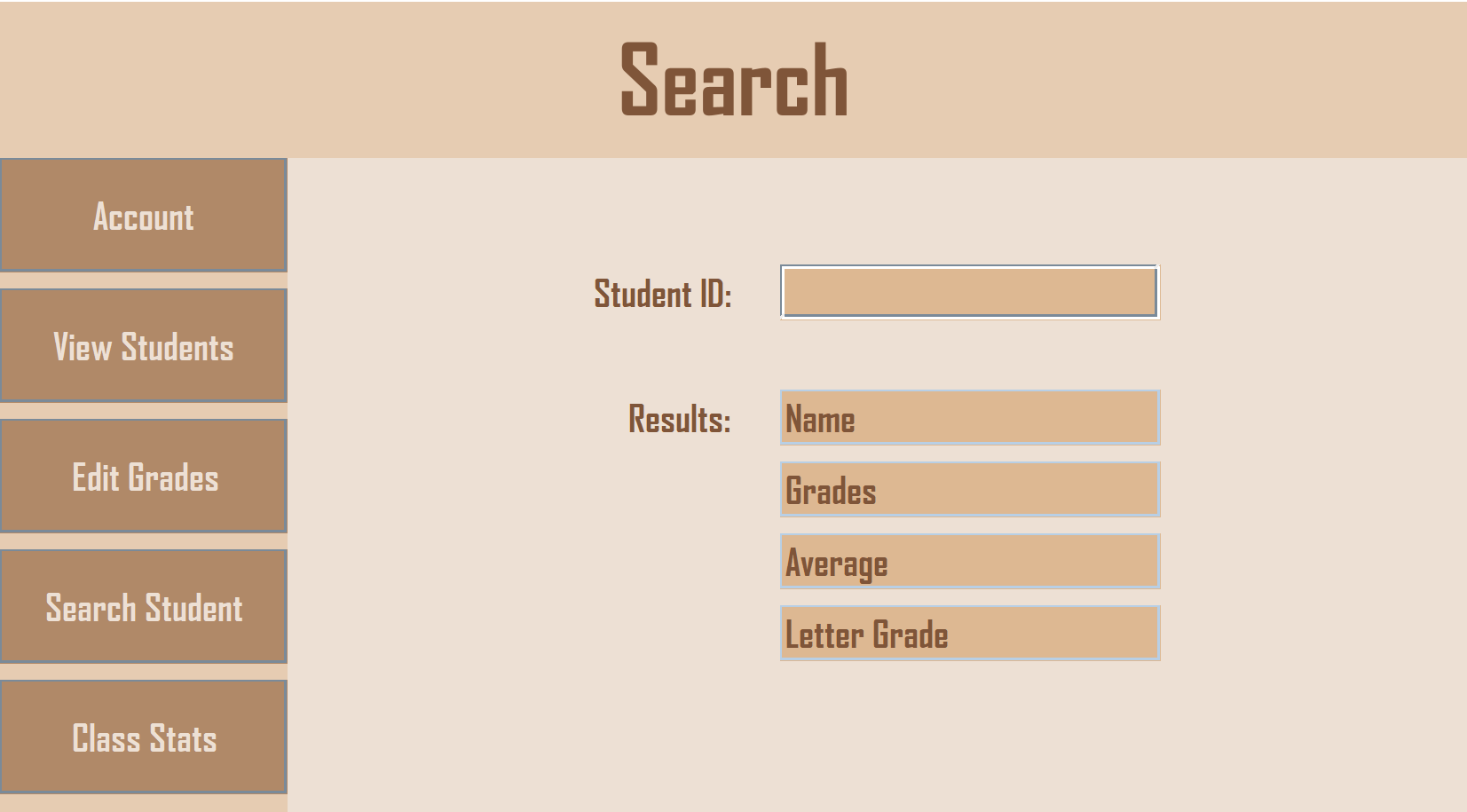


The view table also reflects these changes



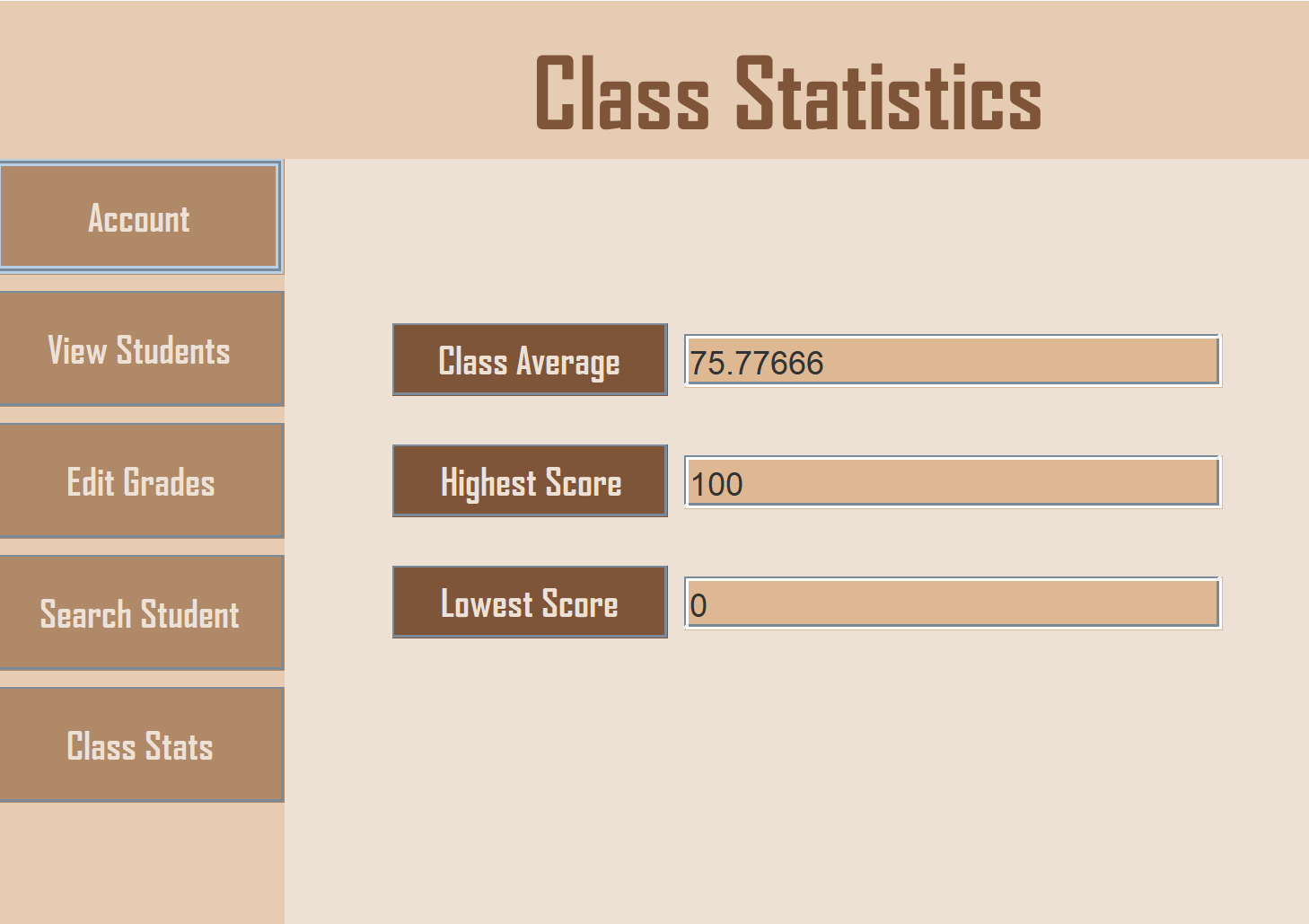
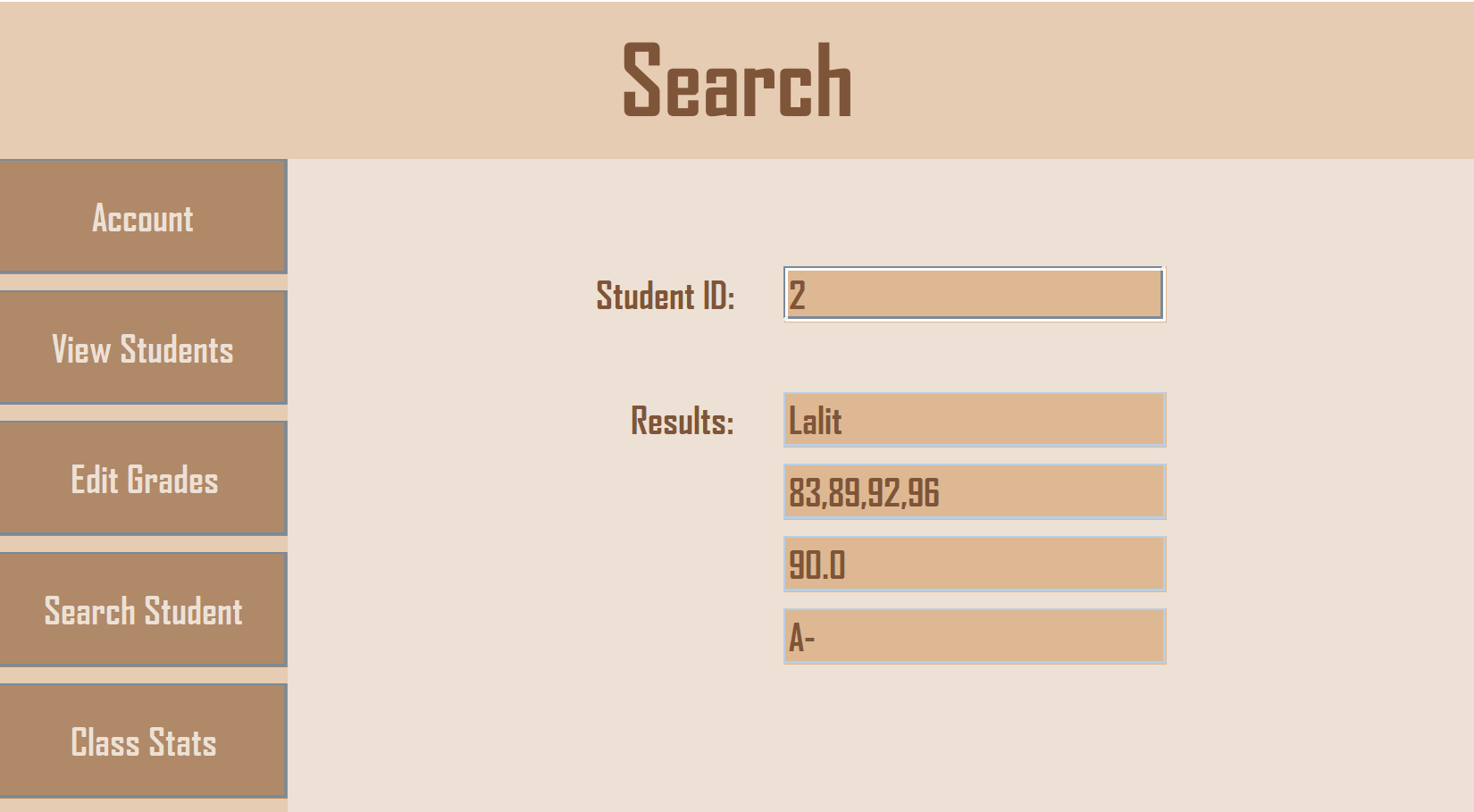
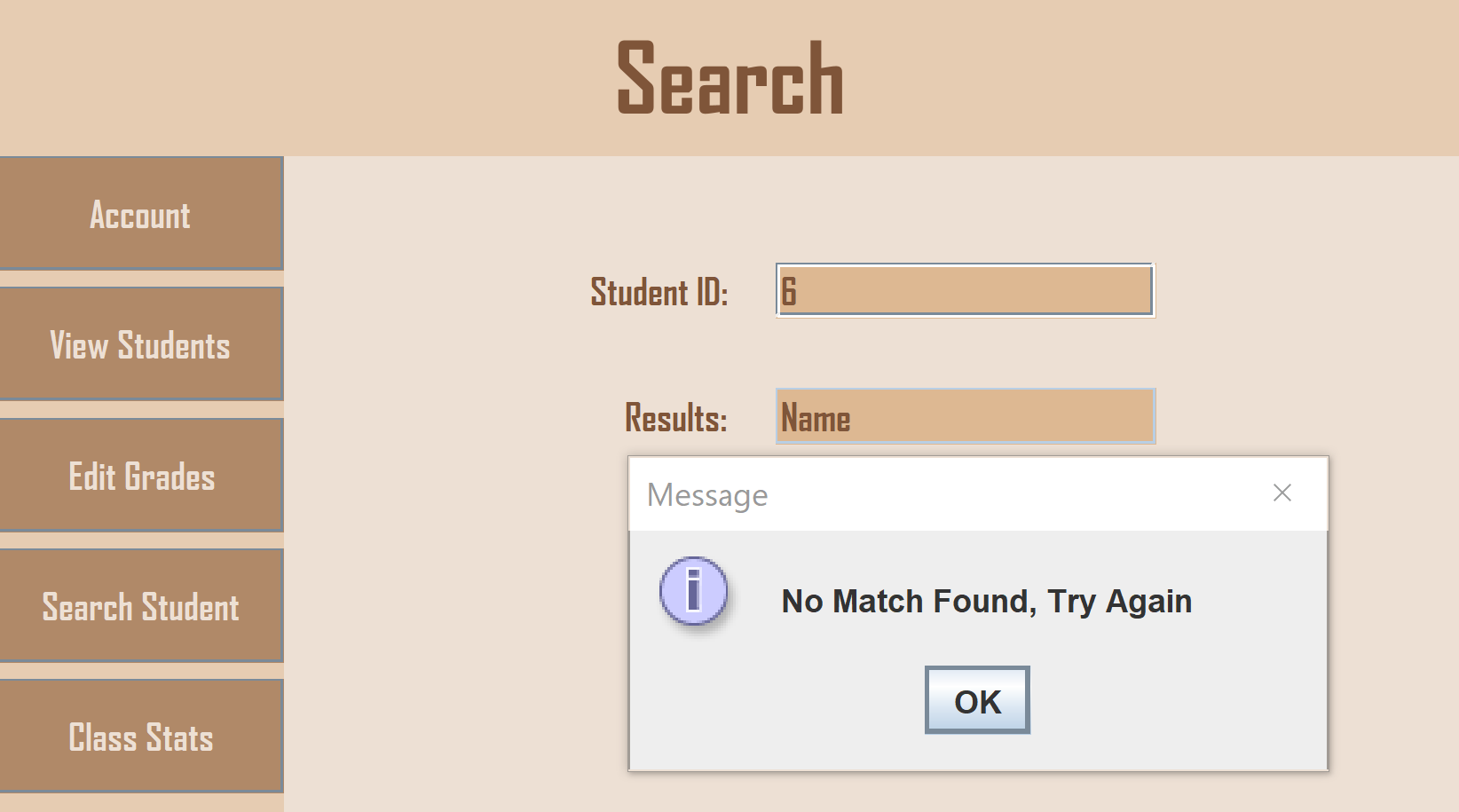
There is a test.txt file in the project folder that also contains these commands and follows the same rules. If you hit enter on the filename text field, it will parse through the file and implement all of these commands.

Everytime there is an edit made, the grades are sorted using a parallel merge sort and implements the use of up to 4 threads. Since the sorting happens so often, the threaded Merge Sort makes it so that it all happens as quickly as possible.



Next is the Search Page. Here you input a specific student ID and hit enter, it will display all their information.

If student ID doesn’t exist: If student ID does exist:



Lastly is the class statistics page. If you hit the buttons it will calculate and display the numbers.

Since the grades are always sorted, stats such as the lowest and highest score don’t require parsing through every single grade. It just compares to the last or first number in each list. This speeds up the calculation process.

Also, since each student’s average is already calculated, instead of going through every grade list individually, it can just take the average of all the averages.

Project Notes:

