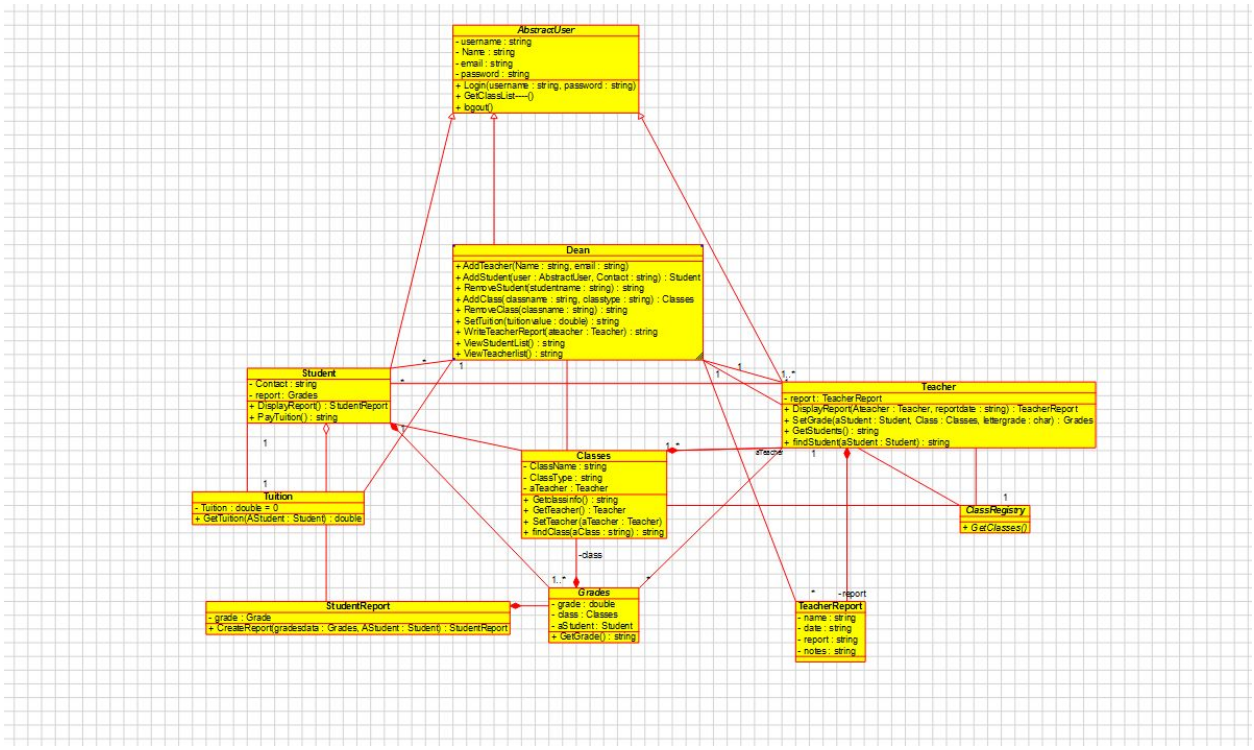
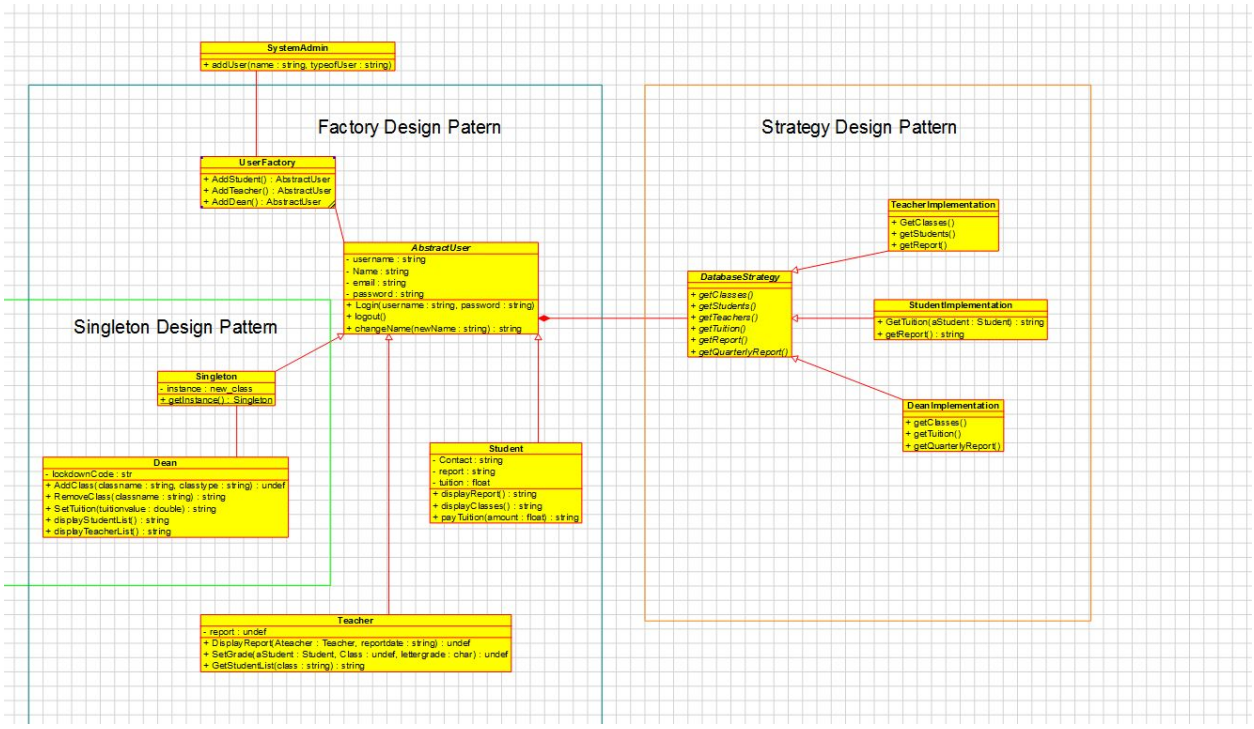
The old class diagram:



The new class diagram:



During this phase of the project we Implement 3 design patterns: Factory, Singleton, and Strategy.

The factory can be seen with the UserFactory class, which connect to the client (the system admin) to create different types of users. The three users the factory can create are: a dean, a teacher and a student.

The Strategy design pattern is used in regard to our database, and is used to create different algorithms depending on what type of user wants to access the data. For instance, if a dean wants to connect to the database, and use the method getClasses(), he should get a list of all the classes that the school offers. But if a student were to call the method getClasses(), that student should only get a list of classes that he/she is enrolled in. Because different users should have different algorithms for the same operation, we used the strategy method for this.

We also used a singleton pattern to specify that there can only be one Dean at a time -- this is fairly straightforward.

Lastly, we also refactored our code so that it better follows the principles of OOD -- specifically encapsulation. Operations related to an object were moved to that object's class, instead of trying to put methods in classes based on permissions. The dean class originally also was very bloated, and we were using it as a controller for the whole system, so we moved most of the operations out of this class.