Retrospective Document

Failed to implement Adapter Pattern

Initially we plan to implement a log in page for different kinds of users. We attempt to use adapter pattern that we taught in class to adapt different users. However, adapter pattern was not as easy to implement as we thought. Eventually, the log in page for other users are simply copy, paste and override of the first user page. And it took us a long time to figure out which part should be override, and which part should not.

How to improve

Make sure the class graph can be drawn correctly before the actual implementation of project. Don't realize the importance of adapter pattern until the middle of project. It is already too late, and code is already stable. In addition, spend more time on architectural sketch. The choice of pattern shall be obvious if we invest more time on the planning stage.

How to Measure improvement

To measure the improvement, the easiest way here is to compare different user page. If the adapter pattern is implemented successfully, there should be eventually multiple interfaces for different users. But the source code should be the same.

Unideal Distribution of Work

Initially, we assign database to one group member and UI to another one. We plan to integrate everyone's work at the end. However, we realize that it is hard to integrate their work if they don't know well of each other's stuff. Most of the time, UI developer does not understand why the database is created in this way. And database developer does not know which button he should link to.

How to improve

It is better to distribute by the components that can be progressed simultaneously. For this project, we should not divide the project by database and UI. We should divide by main functionalities. For example, one can be responsible for course enrollment, while others can work on user registration.

How to Measure improvement

The success of improvement also means there is no extra work for integration. If the distribution of job is reasonable, the integration shall be done naturally. Instead, if collaboration directly causes mandatory code rewrite, the work distribution is definitely not efficient.