SENG 300 Winter 2020

UCID: 30063099

Tutorial Number: 4

Retrospective Report

By

Haohu Shen

April 11, 2020

What went well in this project?

- Since all of team members agreed that this project is a good chance to learn new frameworks for web development, we all decided to make a webpage for our UI instead of using JavaFX even though web UI may bring us more challenges.
- We used **Mybatis** in our persistent level and abandoned the traditional way to access our databases, that is, we put all our SQL statements in multiple XML files instead of creating CRUD prepare-statements, which brought us more flexibility and robustness.
- We separated back-end and front-end by using different frameworks:
  - For the back-end part, we used **mySQL** to create tables and initialized them with some dummy data.
  - To connect Java and mySQL, we used **myBatis** in our persistent layer.
  - To connect the front-end and the back-end part, we used **SpringMVC**.
  - For the front-end part, we used **Vue.js** with **element UI** for CSS.

What were the difficulties you faced and how did you solve that?

- I spent a lot of time on helping other group members setting environments since the steps of installation and configuration of the same toolchain on different OS platforms are different, including conform the decoding setting.
- Some important unit tests about some SQL statements I wrote are not covered, which caused some serious logic issues during our test on the front-end, thus I had to re-check my repo again to fix these problems. These problems are basically about logic of the relational algebra.
- We have two types of end-users (clients and administrators) for our webpages and I was confused how to store their accounts' passwords in our database. After discussion with other team members, we decided using md5 to encrypt them.
- I was also confused on how to define the primary key for some entities between using auto-incremental primary key or any other method, after the discussion with other team members, I decided to use UUID since both of mySQL and Java have built-in methods to generate UUID.
- Another difficulty is that using Git control in **IDEA Jetbrain** is a kind of different from that using Git terminal or operate on Github webpages, for example, sometimes when I merged files from other branches, some OS-specific config files are merged accidentally as well, which may cause some issues hard to find, thus I learnt how to use ‘git rebase’ to turn to some specific version instead of redo the ‘git clone’ thing.
- It is our first time to apply the agile software principles in the project so actually we didn’t do pretty well on it, but we did learn something important from it, the most difficulty is that we cannot balance the workload for every member perfectly since each of us has different schedules on our daily lives.

What did you learn?

- How to use **Maven** to manage all packages we need in **IDEA Jetbrain**.
- How to install and initialize **Tomcat** and integrate it into **IDEA**.
- How to integrate **SpringMVC**, **Vue.js** and **element UI** into **IDEA**.
- Some basic operations and how to set up the configuration for **myBatis**.
- How to write SQL statements using mybatis under SpringMVC, especially on how to manage the files with different types.
- How to use logs in Java for debugging.
- Some best practices of the agile development.

Something else I prefer to mention

- I decide to re-construct this project by replacing **Vue.js** with **React** since the latter framework is much popular in Canadian IT industry and thus much useful in this summer vacation.
- I will try to find some plugins that will help me to generate SQL statements automatically because I still think writing SQL statements one by one is inefficient and not practical.
- I decide to re-construct this project by replacing **Tomcat+SpringMVC** with **Springboot**, which is more efficient.
- The project we made so far does not prevent from **DDos** such as Challenge Collapsar attack, thus I decide use **Redis** in my re-construction to address this issue.
- I still think simply using **md5** is not safe enough, so I decide to replace it with an implementation of **Bcrypt** in Java for the re-construction since MySQL does not have a built-in function to support this feature.