Sprint 2 Retrospective

Team 19 – Sentinel Data Vault

Jiho Choi, Zhaoji Jiang, Adam Petty, Dingfu Sun, Thomas Worns

What Went Well?

1) **Sprint Task Planning**
The progression of our sprint adhered quite well to the original plan established in our Sprint 2 Planning Document, more so than Sprint 1. Unlike our first go-around, we more realistically planned tasks that were possible to accomplish in the time allowed. Over the course of the sprint, we made very few adjustments to the Sprint 2 Planning Document and user stories set to be implemented. We were able to better prioritize tasks and did not need to remove any as we progressed through the sprint. The benefit of hindsight and experience helped this process go much more smoothly.

2) **Database Implementation**
After we found out that most of the team members had compatibility issues with MySQL database, we decided to switch to SQLite database immediately. We regained a huge amount of time that was lost due to the fatal error caused by the database during last sprint. Our team members quickly became familiar with interacting with the SQL database from Java using JDBC and completed the setup early on, which allowed us to work on the main functionalities of the project smoothly.

3) **GitHub Organization**
All of the team members began using a .gitignore file which allowed us to keep our repository clean and organized for the whole sprint. The team utilized and took advantage of the branching/reverting functions of GitHub in order to test and experiment with new features without the risk of losing any progress on the whole project. We cleanly organized our source files and project documents using an understandable file hierarchy. At all stages of development, files were consistently synced with the repository to prevent loss of progress.

4) **Consistent Workflow**
We better understood how each team member works individually and how we collaborate as a team from the previous sprint, which allowed us to plan more effectively and get more done during this sprint. This planning allowed us to work more consistently over this sprint. This consistency allowed us to get all of the things we had planned to do done. We also had more time to work on this sprint because we did not have external factors (like exams and other projects) interfering with our progress. Tools like GitHub and Trello allowed us to keep track of exactly what was going on and being updated as it happened, as well as who was currently working on what part of the project.

What Did Not Go Well?

1) **Coordination / Communication**
While we synergized exceptionally well in Sprint 1, we lost some of our coordination this time around, and our communication changed. We shared ideas, suggestions, and bug fixes like before, but we did not communicate progress on our tasks to other team members qutie as efficiently as last sprint. This was primarily caused by the range of tasks being worked on. Last sprint was all about the user interfaces and that was where everyone was focused and working. This sprint work was getting done in varied places whose implementations had little in common. We also had several independently developed fixes and minor redesigns introduced by a single person with the most experience in their respective areas of the system, leading to fragmentation of team focus. An example of this was failing to realize that while encryption and decryption functionality had been completed, it was not fully integrated into the data vault system.

2) **Making Cohesive Code Decisions**
Upon completion of individual work, we spent more time than preferred integrating different classes and functions, not having a coordination plan for these items. For instance, confusing or conflicting variables and methods. Some confusion also arose over what responsibilities specific classes would have in terms of their functionality and how those would be used system-wide. In some cases, the control flow would be carried out by one class (or set of classes), but then run through a different set in another case even though they were closely related. A common example of this that occurred was the flow of data from the UI to the database – sometimes going through VaultController and DatabaseManager, and other times going directly from UI to database.

How Should We Improve?

1) **Coordination / Communication**
To improve in this area, we need to ensure that all progress towards the completion of any of our planned items is tracked carefully through our Trello board. Any miscellaneous fixes or design tweaks that need to be made are made known to the entire team. Commenting code with TODOs and noting any significant progress in the Git commits would go a long way to fixing our coordination as well. We also need to ensure that one person's work does not interfere or conflict with another person's code development. It's also important that each team member is aware of the tasks that are being completed by the other members.

2) **Making Cohesive Code Decisions**
Before starting actual coding, we will need to better make team consensus decisions on design, implementation, and code style. It is imperative that all members are on the same page when it comes to making these decisions and has a complete understanding of them. This will save a lot of time when it comes to integrating code. Also ensure that everyone has a solid understanding of the correct program control flow, and everyone understands how to utilize the proper classes.