Deliverable #5 Report

Team #2 SQL Injector

CSCC01H3 INTRODUCTION OF SOFTWARE ENGINEERING

2015-11-24

Nadeem Haidar
Junaid Patel
Andres Moreno
Minsoo Park
Yujian Chen

Deliverable #5 Report

Team #2 SQL Injector

Sprint / Product Backlog, Burndown Chart, and Taskboard Snapshots

- This information can be found from GitHub repository
- For Burndown Chart Snapshots, these can be found on Trello board

Code inspection

- Reports can be found in GitHub repository

Sprint Plan

- *Initial plan*

  Same as the previous iteration, each team member has planned 4 hours per week to work on the project. That totals to 40 developer hours for this two-week iteration. The chart below is the amount each member planned to work each day. The number in the bracket indicates the task that each team member should work on.

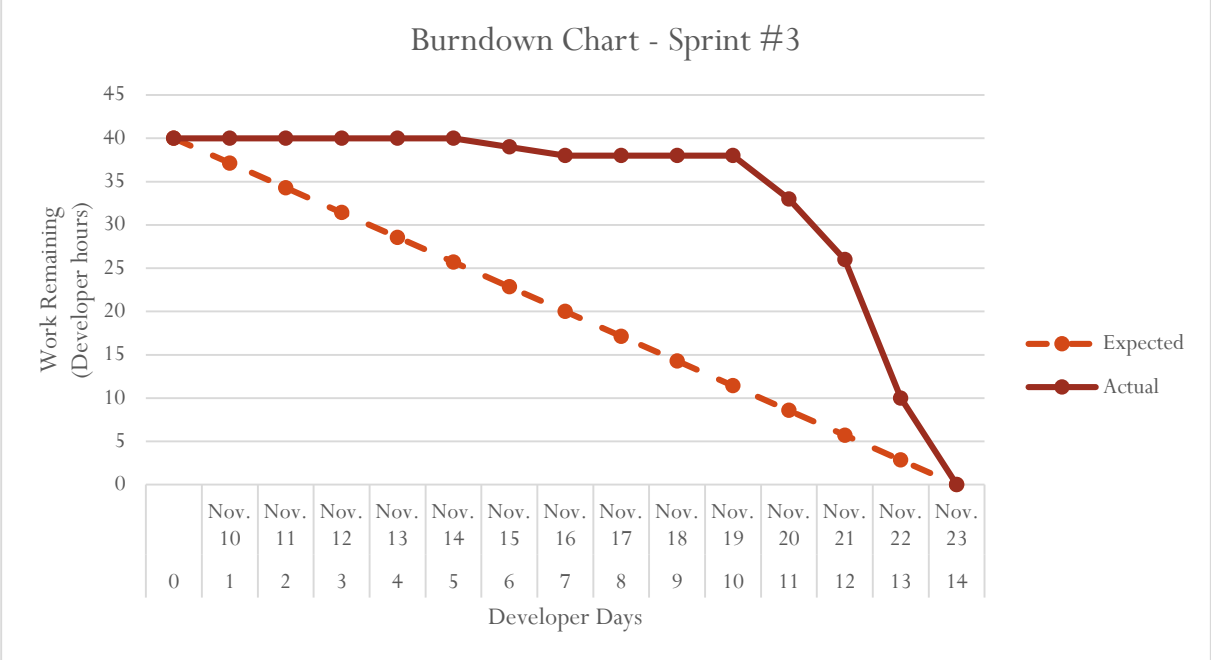
Date	Nadeem	Junaid	Andres	Minsoo	Alex	Total
	0	0	0	0	0	0
Nov. 10	0	0	0	1 (t2)	1 (t2)	2
Nov. 11	0	0	0	0	0	0
Nov. 12	0	0	0	0	2 (t4)	2
Nov. 13	0	2 (t2+t3)	0	1 (t5)	0	3
Nov. 14	2 (t2+t3)	2 (t5)	1 (t2)	1 (t8)	1 (t9)	7
Nov. 15	2 (t6)	0	1 (t4)	1 (t8)	0	4
Nov. 16	0	0	0	1 (t8)	0	1
Nov. 17	0	0	0	1 (t8)	0	1
Nov. 18	0	0	0	0	0	0
Nov. 19	0	0	0	0	0	0
Nov. 20	0	0	1 (t4)	0	0	1
Nov. 21	2 (t9)	0	1 (t11)	1 (t1)	0	4
Nov. 22	2 (t10)	2 (t7)	2 (t11)	1 (t1)	2 (t1)	9
Nov. 23	0	2 (t10)	2 (t11)	0	2 (t1)	6

- ***Actual Logs of the plan***

We were unable to follow this plan. However, we are able to finish all tasks we decided to do in this iteration. This was mainly due to the reason that our implementations had unexpected bugs and there were difficulties using the third party library in our plugin. Our daily log is shown below.

Date	Nadeem	Junaid	Andres	Minsoo	Alex	Total
	0	0	0	0	0	0
Nov. 10	0	0	0	0	0	0
Nov. 11	0	0	0	0	0	0
Nov. 12	0	0	0	0	0	0
Nov. 13	0	0	0	0	0	0
Nov. 14	0	0	0	0	0	0
Nov. 15	0	0	1	0	0	1
Nov. 16	0	0	0	1	0	1
Nov. 17	0	0	0	0	0	0
Nov. 18	0	0	0	0	0	0
Nov. 19	0	0	0	0	0	0
Nov. 20	0	0	3	1	1	5
Nov. 21	2	1	0	3	1	7
Nov. 22	6	4	1	2	3	16
Nov. 23	0	3	3	1	3	10

- ***Burndown Chart***



- ## *Current state of the project*

In this iteration, we finished what we had planned. At the beginning of this iteration, we decided that we will finish the leftover editing tags from the batch-editing feature, and start working on custom output styles feature, which is **B04** and **B05** from our sprint backlog. The taskboard snapshot in GitHub will reflect what we planned. During our researching, we found that on GitHub there is a open source project call csl-editor for creating and designing custom output styles exactly as stated in our **B04** and **B05** user stories. However, there were some modifications that needed to be done in order to make this library work in our plug-in: It requires an in-app browser in order to load the website, a server to host our modified website, and an option to install the style right into Zotero without the user having to downloading the csl file and install it manually.

The largest problem we had in this iteration is in the csl editor, it depended on flash to save the csl file and the user had to download it manually and install it. We had to modify the code to get the csl file as a string and then raise an event. We also had to develop a custom browser inside zotero that listens to the event and gets the string representation of the csl file.

B04: As John, a researcher, I want to be able to add or remove fields from a specific output style and modify the format of the style, so that I can match the requirements requested by my professor.

B05: As John, a grad student, I want to be able to create new output styles without modifying the original ones, so that I can keep the original ones intact.

- ## *System design*

We modified our system design in this iteration since we introduce a third party library and a server in order to make our customized output style work. Batch editing remains the same as stated in last deliverable report: The batch-editing component have a backend logic (`ExtBatch.js`) for removing, renaming, adding, editing, merging tags and pulling data from Zotero. The UI file (`ui.xul`) contains all the layout and design and embeds itself in Zotero's UI file.

The custom output styles is a combination of several components. The plugin has a custom embedded browser that loads our website. This browser loads the website from our server that is running the csl-editor source code and the user is able to modify and edit their style through the website. After the user finishes editing, and saves the style, the server sends an event with the string representation of the style. Our custom browser listens for this event and automatically installs the style to Zotero.