Issue Tracker Express

Sprint Retrospective 3

CPSC 3720

University of Lethbridge

Team Aegir

Lorenzo Conrad

Mathew Richards

Steven Deutekom

December 2, 2019

1 Introduction

We have completed our third and final sprint for Issue Tracker Express. Below we reflect on the experience of this last sprint and the project in general. We will give some details of our initial plan for the third sprint. Then we will show what we actually got done and how close we got to achieving all our proposed objectives. This will be followed with some description of a few of the problems we encountered during the sprint. Then we discuss what things we could do if the project were continuing. Finally, we will reflect on the project and what we learned from it.

2 Sprint Plan

The intention for our third sprint was to finish all of the feature that we proposed for the project. We had implemented most things in our server, but several things were missing from the front end. On the server side we needed to get the filtering of issues and comments

working properly. We also had several unit tests to complete as well as finishing up some doxygen comments. We needed to make sure that the server was relatively robust in handling most of the things we expected it to handle. We had to properly implement saving and loading of the server state. Lastly, we wanted to change our DELETE endpoints to use GET as the DELETE ones were not working properly with our web client.

For the front end we had a lot of work to do. We still had no way to do comments. It was not possible to remove any issues and users or to edit them. We needed to make it possible to filter issues by tag, status, and priority so that users could narrow the list of issues they wanted to look at. In addition to adding all of these features we wanted to make the web page a bit nicer to look at and more user friendly were possible.

At this point we knew that we were not going to be able to carve out the necessary time to have formal standup meetings. This was something we struggled with during our other sprints. So we planned to keep in touch with each other on slack and to meet in the computer lab daily to see what our schedules were like and try to work together to move the project forward whenever possible.

3 Sprint Results

At the end of our third we were able to provide basically all of the stuff in our project proposal. The only thing that we did not get working as we proposed was issue filtering/searching. This was originally supposed to include keyword and phrases that would search Issue titles and possibly even descriptions. In the end for simplicity sake and time we only implemented filtering by status, priority, and a single tag. We feel that this still gives a great deal of power when trying to narrow down what issues are listed. And with the addition of some input fields and selectors on the list issues page it is relatively easy to update the issue list.

On the server side we got everything working. We tweaked our endpoints to do removal

and filtering with GET. The different parameters that could be added changed the behaviour in a nice way. This meant we could use fewer endpoints to get all of our service working. Our unit tests were finished and allowed us to confirm the proper functioning of every part of our service. Even though the handler code was not tested because of g++ issues, we were able to put all the really hard work into the model classes where it could be tested well.

We also managed to handle some of the errors we thought might come up. However, we know that we did not get all of them. Some of these things we tried to make sure would not be possible with the endpoints, but there is no code in place to guarantee they won't happen. An example is adding two issues with the same id. It is not possible to do this with our endpoints. The POST request adds an issue if there is no id, but will update an issue if an id is given. So under normal use two issues cannot have the same id. However, if the save file were altered or a developer made a mistake coding the system does not care if two issues have the same id. This would need to be fixed in the future.

On the web page we were able to get all functionality working. It is possible to create, edit, and remove issues, comments, and users. The user page and profiles properly display pictures for the users like we proposed. We managed to get our tables that show the lists of users and issues to display cleanly. There is no longer any need to refresh the pages to display changes. Unfortunately, we did not really know how to deal with some of the JavaScript ways to update pages without refreshing them until we were basically done. This meant that all of our forms open on new pages and are rather plain. They get the job done for now, but if we were to continue with the process we could do a much better job integrating these components.

As for working as a team we did manage to communicate well for the third sprint. Unfortunately, our busy schedules left us with less time to actually work together. This meant that sometimes those with time were forced to pick up the slack to meet the deadlines. We feel that if we were not in school full time while working on this project that we could

do a better job with this. We did use our Kanban board consistently to track work and add issues and features that needed to be dealt with. Overall our team functioned well together and there was no friction. We all stepped up when required to try and deliver a quality product in spite of the sometimes difficult circumstances.

4 Problems Encountered

Our main problem on this sprint was not having as much time as we wanted. It was difficult to get together to work on the code. As Steve did a lot of the more complicated work during sprint two it was also difficult for Mathew and Lorenzo to know where to start when they did have time. We managed to get everyone working somewhere, but had time allowed each member of the team would have liked to be more involved in each aspect of the project. Code reviews would have been a nice way to do this, or just more in person sessions where the work could be divided up nicely and everyone could ask questions.

We ran into some small technical problems here and there. We did not realize that JavaScript would not even display errors if we didn't explicitly catch them. Pages would just not function properly. We spent some time trying to figure out why responses were not coming back properly only to realize that some of our JavaScript code was misspelled. The responses were not the problem, but instead the code that handled them. More familiarity with JavaScript, and possibly a development environment designed to identify and debug these errors, would have been useful.

We also ran into some issues with the JSON library. Over the project this library saved us a lot of time, but on a couple of occasions we made mistakes with it. The first was with testing. If you directly compare a string to a JSON field in an EXPECT_EQ call it segfaults when they are not equal. This can be confusing and the solution is probably to save the JSON fields as strings and then compare those strings properly. However, we did not figure

this out until it was too late. We were able to get things working and pass the test though. The other issue was that originally we did not nest our JSON objects properly. This made the JSON we dumped to our save file have extra characters it didn't need. When we tried to load the files contents the JSON library could not parse the contents. Once we discovered what could be parsed we were able to adjust our saving code to dump the JSON properly and everything was fine. It did cost us some time.

5 Future Improvements

If we were to work on this project further there are a few things we would like to improve. The website still shows a user id by a comment instead of a name or picture. We managed to fix it in other places, but did not want to make all the API calls necessary to get each user name for a long list of comments. Probably the best change would be to send more user info with comment responses instead of just ids.

We also would like to make the web client and server more robust. The restbed server is pretty good at handling unexpected errors and returning HTTP errors to compensate. However, we would like to be able to catch these errors ourselves and send more informative responses, or at least document what kind of reasons there could be for certain HTTP errors.

Our IssueSystem class also felt a little big. We did manage to keep all the functionality we could in the objects, but there was still a lot going on in the system. In the future we would try to see if there was a way we could extract a class or two to deal with certain behaviours. Perhaps things like searching, filtering, and sorting issues. This way the issue system could fit more with SRP.

6 What we learned

We learned many things from this project. The most obvious thing is all of the new tools and concepts we were able to work with. Using frameworks with our code made the development of our code much easier. The back end was really not too hard to get setup once we understood what to do. It would have been even easier if we had not had compatibility issue between our server framework and testing framework. But this too is a lesson in choosing the right technologies. It also forced us to try and encapsulate as much of the business logic as we could in classes that we could test. This led to a better design in the end, so it was a good thing.

We learned some basic JavaScript and HTML. While we did get more familiar with these technologies we also learned that it is better to work with them one piece at a time. Learning a new language when you need to do complicated things is difficult. It takes away the ability to learn simple things and build on them. It also make it much harder to follow any best practices or come up with good designs. We are all familiar with good design at this point, and we did not succeed at it with our web client. Everything works and we didn't anything really dangerous, but our web client is not clean or very well organized. In the future if presented with the need to learn new technologies we would all like to spend time learning before diving in too deep.

We learned a lot about trying to organize time and keep everyone up to speed on the project. It was mentioned above that sometimes a lot of work was done by one person without anyone else being involved. This makes it really hard for others to work independently. When someone wants to get involved they have to ask a lot of questions before they can decide where to work on the code. We would all hope in the future to find ways to mitigate this issue. It probably would help most to have more frequent meetings and do things like code review. However, it could also help to better document the work to be done with our Kanban

board. Giving more details in the description of an issue might help another developer know where to focus their attention when starting a new feature or fixing a bug.

Lastly, we all learned a lot from the general chaos of this project. There were lots of little things that went wrong. Frameworks not playing well together, GitLab crashing often, and poor time management. In the future we will try to set ourselves up for larger projects keeping this in mind. One has to expect that things will not go as expected. We all know that our code will never be working as quickly as expected, but there are many more factors to consider.

7 Conclusions

This project has been a great learning experience. We got to build a piece of software that could actually be useful in the real world. I am pretty sure we are not going to compete with commercial issue tracking software, but what we built does have the core functionality of a usable issue tracking system. Also, RESTful services are common place and micro service architecture is very popular today. Getting experience working with these technologies and processes is great experience for our futures. With the knowledge we have gained this semester we could easily start developing our own services, if we knew what they should be. We could even have an easier time fitting into a company that uses these kinds of technologies and architectures much easier. In spite of the increased workload for a semester it has been very rewarding.