Reykjavík University
Spring 2016

T-404-LOKA, Final Project

Final report

Arnar Kári Ágústsson
Árni Benedikt Árnason
Daníel Benediktsson
Jón Freysteinn Jónsson
Jörundur Jörundsson

May 13, 2016

Supervisor: Hannes Pétursson

Contents

1 Introduction

This report describes our final project at Reykjavik University, how successful we were in implementing the required features as originally stated, the highs and lows of the work process as a whole and the future of our project.

2 About the project

Our project was to create a logging system and a customer support system for a company called **SolidClouds**, more specifically for their game **Starborne**. As described on the official Starborne website, Starborne is a take on the grand-strategy 4X genre with the additions of being multiplayer and in real-time. A single game of Starborne will last six months and culminate in a decisive endgame. The game is projected to be released around the beginning of next year.

The logging system's purpose is to gather data about player behavior in the game in order to detect anomalies, ease bug-finding, provide better customer support and visualize the behavior of the game players and servers. The Starborne game servers post data from the game (player operations, server errors, messages, etc.) to an API we created. The API handles moving the data along to both a MongoDB database and the ELK stack (Elasticsearch + Logstash + Kibana). In layman's terms, the ELK stack gives us another way to store our data in addition to providing lightning speed search capabilities and a powerful way to visualize the data. Data visualizations are of utmost importance, e.g. when determining what features users seem to make use of too much and what features they rarely use. The logging system handles dynamic data so as to work with various types of data since the structuring of it can change during development of the game.

The customer support system is a petition (or ticket) system for customer support, where tickets can be tracked, stored, answered and have their status updated. This is important when striving for user satisfaction as this provides the means for users reporting issues and receiving help whenever errors and other unexpected events occur within the game. We created an API that handles all core ticket functionality (creating, updating, deleting tickets, etc.), a web client written in the React framework and using the Redux architecture and an in-game form for creating a ticket. Players can create a ticket via our web client or in the game itself. In the web client both players and customer support staff can monitor tickets, chat with each other about the issue, and add more information or attachments to the ticket. The importance of a good customer support system is even greater due to the fact that Starborne is still in development and many bugs have yet to come to light. For a customer support representative to resolve a player's request he must be able to verify that the information from the user is correct by looking into the logs created by our logging system. Customer support staff can see logs by users inside their view of the web client, thereby tying the logging system to the customer support system in an important way.

3 Results

3.1 What went well?

There are many positive things that we can take from this final project. The group was very united beforehand so all group work was very successful. We were a five member team and we knew that it would be a challenge to keep everybody focused and on the same page. We were aware of the fact that efficiently dividing work and issues between group members would be crucial for us and we did that very well from the start. This made all work much easier and faster which was important for us.

We were also very happy about the amount of work we got done in the project. We finished all issues from the initial project plan and even added two major features to our project, the in-game client and a connection between the customer support system and the logging system. By implementing those two features we glued the two projects (or systems) together and accomplished making them a solid system and we are very happy about the final result. We think that SolidClouds can easily start using the systems right away without doing much work themselves and we have already showcased our work for them.

We are also very satisfied with the technology we learned during the course. We challenged ourselves by using technology that we had little experience with and we were pretty quick getting used to the new environments. We had never used the ELK stack before and we also had little experience in React/Redux. Knowledge in those technologies is very valuable and we are happy about going through the learning curve that accompanies them. We think we used a very broad technology environment in the project: C# .NET, React/Redux, Unity, ELK stack, MongoDB and Amazon Web Services.

There was never a moment in the process of the project that we were afraid about the result of the project as a whole. We were focused and determined to do well in this final project at our university and finish our degree on a positive note. We think that the group has become even more unified by going through this process together and we are confident that this is just the beginning of the group cooperation.

3.2 What did not go well?

Even though we find the result of the project very positive there is always something that could have been done better. The first thing that we thought of was the technology stack that we chose for the Logging System. We were working with very dynamic data with C# .NET which is a strongly typed environment. After reconsideration it would have been better to use an environment similar to NodeJS for the server work for that system. We were also very unsure how to structure the Logging System and found it difficult to get a result that we were satisfied with. Logs are very complicated things to work with because the amount of them can be enormous. This uncertainty resulted in difficulty finding the right work rhythm to begin with.

Even though we were very happy about the learning curve we had to go through with new technology as discussed in the previous chapter, we think that our lack of experience of those technologies was a bottleneck to begin with. This applied especially to the Logging System where the ELK stack took a lot of our time and resulted in longer working hours and frustration at times. We think that it would have been a lot better to seek more help from a person with experience in the ELK stack.

A final thing we would like to mention is that the facilities at SolidClouds were a bit of a disappointment for the group. The room we got was very small and affected our work sometimes. We had to move the facilities to one of the group members home during the three weeks period to keep the focus at highest. When looking back at the project, we should have asked for better facilities from the SolidClouds team to begin with.

4 Visualization of Work Flow

4.1 Release Burndown Chart

Jira gives us the possibility of creating a release burndown chart for all issues (stories) in a version. At the beginning of our work we created a project in Jira, but we did not specify a version for the project as we though that was not applicable at the time. Changing the version for all stories afterwards in order to create the release burndown chart resulted in an incorrect chart and in the end we opted to import our data to Excel and manually create a release burndown chart.

The blue line in the chart represents the story points of our original backlog and an estimation of story points needed to finish each sprint. However, as work progressed we naturally added more stories (and as previously stated two more features) and in the end a total of 637 story points were created. The orange line thus represents the total story points and the story points needed to finish each sprint with regards to the total story points of the final project. Finally, the grey line represents the remaining story points after each sprint.

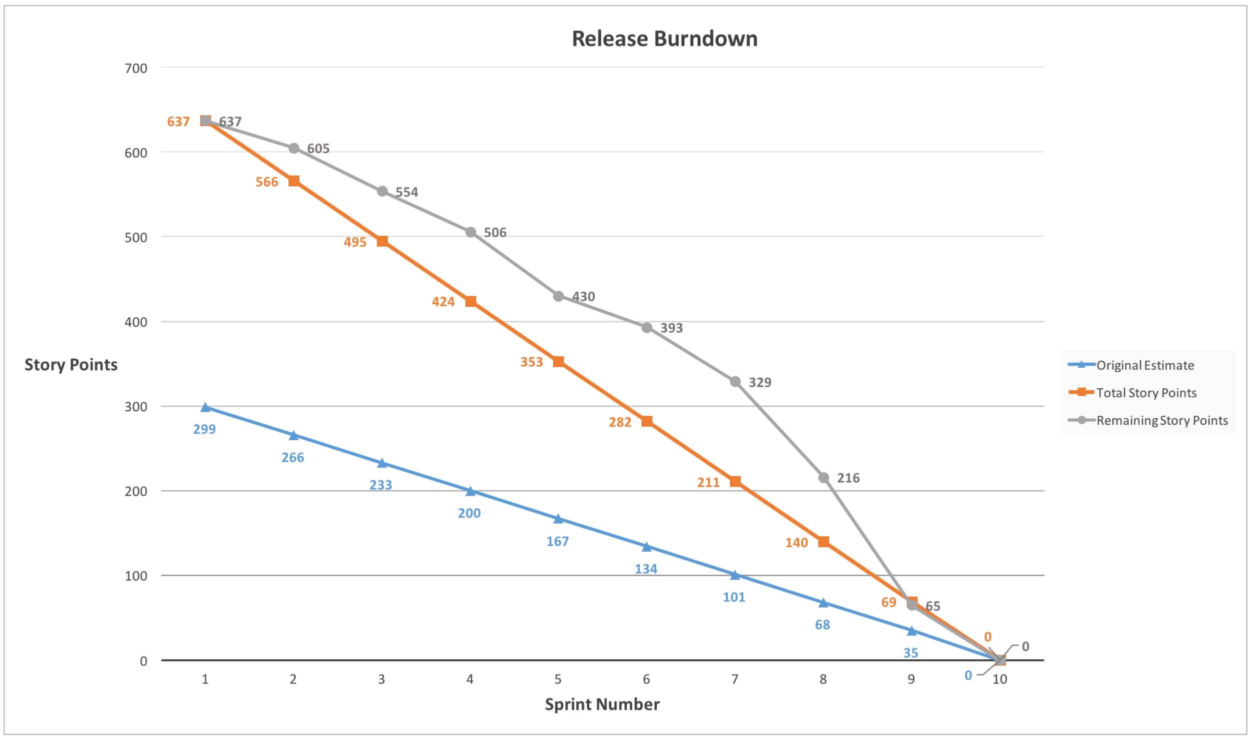
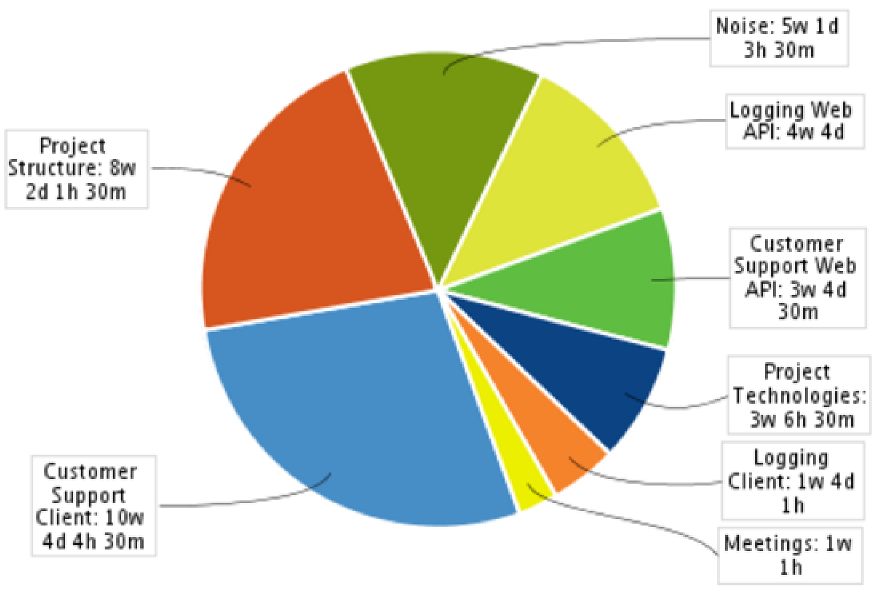
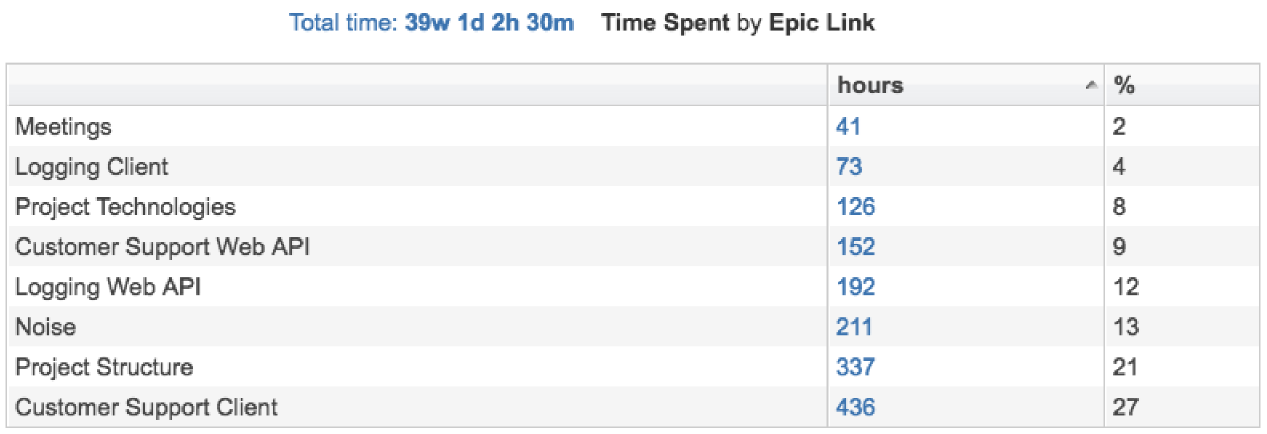


Table 1: Release Burndown

4.2 Time Spent

In the pie chart here below we can see how much time each epic took. The chart is rather self explanatory aside from the Project Structure and Project Technologies. Project Structure concerns all documentation and reports for the project and the Project Technologies concerns all investigation of technologies we went through along with build steps and deployment. We took our time getting into and familiarizing ourselves with the technical environment we worked in.





Total time: **39w 1d 2h 30m** **Time Spent** by **Epic Link**

	hours	%
Meetings	41	2
Logging Client	73	4
Project Technologies	126	8
Customer Support Web API	152	9
Logging Web API	192	12
Noise	211	13
Project Structure	337	21
Customer Support Client	436	27

Table 2: Time Spent

4.3 Burnup Chart

In the Burnup Chart here below we can see the number of issues that are *To do*, *In progress* and *Done* over time. The axis and labels explain the chart very well but the orange area are number of issues that are *To do* at any given time. The blue area are the number of issues that are *In progress* and finally the purple area are number of issues that are *Done*. Note however that some stories were only for time reports like meetings etc. and this result in some issues that still have the status *To do*.

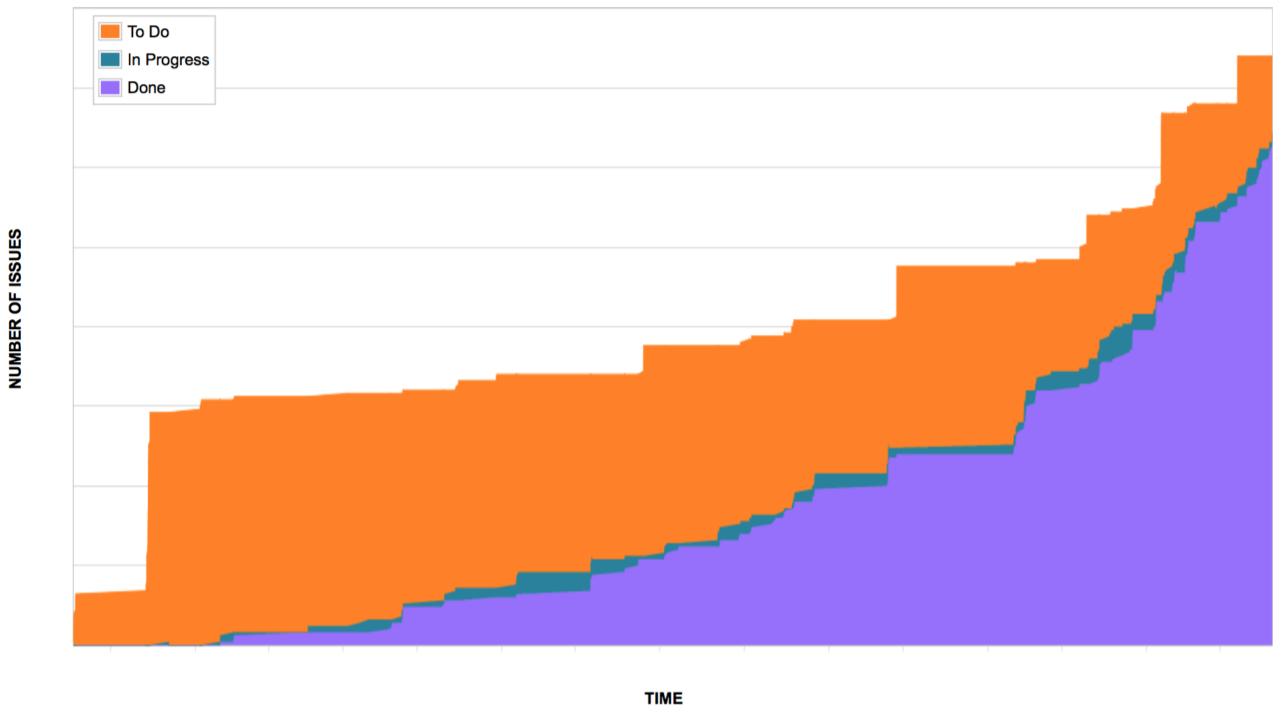


Table 3: Burnup Chart

5 Future

The SolidClouds team intends to use the system we have built in production. They plan on fully utilising our systems when the next Alpha Test for the game begins next autumn. We expect that the system will live up to all their expectations. We chose to create both backend systems in C# .NET because the SolidClouds team has more experience in that environment. All backend code is well documented so it should not be a problem for the team to modify or change logic in the systems.

The in-game client was created using the Unity game engine which is the same engine that the SolidClouds team uses. Our client will then be added to the game with help from the SolidClouds team. This cooperation will familiarise them with our client code and make it easier for them to make changes in the future.

All the features we mapped out in our original plan as well as additional features and changes requested by the SolidClouds team have been implemented into the web client. If any requirements change in the future then the front-end web client will be the biggest hurdle for the Solid Clouds team to overcome since the team does not specialise in web development and does not have experience with the latest frameworks we selected, React and Redux.

We knew all along that other programmers would both modify and add features to our system so we made all code as maintainable and thought out as possible. We are satisfied with the code quality in all our systems and expect that handoff to the SolidClouds team will be smooth. We also created

Developer Manuals which help future developers to our systems. Those documents should make the process of getting into the projects for developers much easier and provide them with resources to familiarise them with the technology stack of each system. We also created a Operation Manual for all future staff that should to be able to setup the system. In that document we go step by step on how to configure, build and deploy the systems.

We have our ideas on what the next steps in development of the system should be. First of all we think that the Logging Systems should handle more load because the number of players will increase by time. This could be done by scaling the system horizontally by adding more servers. We would also like to add processes to our APIs that handles database cleanup. It serves no purpose to keep data that is outdated in the database. Player behaviour changes with the game and therefore very old logs could be misinterpreted by the team and should be stores elsewhere. Moreover, the customer support staff have nothing to do with old tickets and then it is better to erase them from database or store them somewhere else. The final step that we would like to see in the system is to add a notification system inside the in-game client. For example when a tickets status changes the players should get a notification inside the game and should not be forced to login to another web client. The same goes for new messages from support staff etc.

We think that our system has a bright future in the Starborne community and will be of great use for the Solid Clouds staff.

6 Conclusion

We are very pleased with the final result of our project. As previously stated, we finished all the features in our original backlog, and at the behest of our instructor towards the end of this semester we decided we had to take some risks and added two features. These features, injecting the logs into the customer support client and creating an in-game form for posting tickets, proved to be essential in giving our final project a more complete picture and making it more usable in real-life scenarios.

We are also very happy with the work process as a whole, the teamwork and the division of labor in the group. In fact, the group quickly became interested in the prospect of working together in the future on other software projects and towards the end of the semester received an invitation to work on such a project.