### **Customer Value and Scope**

* the chosen scope of the application under development including the priority of features and for whom you are creating value

The scope frontend set for this week was to create a weekly structure page which we finished to a functional degree with room for improvement. We believe that it did create value for the customer and is necessary to construct for future development. Backend created a database which does not create any value for PO but is necessary for future development. The created value was split evenly between PO and us.

* the success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

During this first sprint week we had a lot of focus on getting started. Our MVP goal this week was to deliver a visual weekly view to show our Product owner, which successfully was done. In the backend our goal was to set up a working server as well as a database that could communicate. This was also achieved.

When coming to learning we wanted to get to know the frontend framework, React, as well as getting a bit comfortable with web development using libraries. The backend team wanted to develop understanding about backend architecture. The overall goal for the project in terms on product delivery is to create a software solution for our Product owner that creates real value for him in the long run. As a team we all want to get experience in working with people having diverse backgrounds, as well as working with scrum. Overall getting more knowledge in teambased software development.

* your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

The user stories used in this sprint where quite broad and sometimes hard to grasp. This can be because of the dependency between different parts of the application to get the basics working. We had a standard pattern looking like: “As an X, I want Y since Z”. It worked well because we could see how our tasks related to the user story and their contribution to customer value. But we want to modularize these stories more until next sprint to make them less dependent. Acceptance criteria has worked quite well but could be better If we standardize the different parts of the criteria. Tasks could be more divided between team members to help the functioning of the remote workflow. Over all these parts of the scrum board could be more standardized and detailed but still got us to work towards a goal together while having a good overview of what was needed to be done. For next time we want to make individual bids on the point estimation of each user story to make better estimation instead of only discussing in the big group.

* + The three KPIs you use for monitoring your progress and how you use them to improve your process

The first one we use is velocity, which means the amount of “points” we finished each week. We determine points for a user story together during our start up meeting each sprint. This week our velocity was 12 points. We are satisfied with this, since we finished all user stories that were supposed to be completed. However, since allot of work this week was to learn frameworks, we guess that we will be able to maintain higher velocity in the future.

Our second KPI is a customer survey where our PO gets to grade our work each week by answering the questions:

How well does the current product suit your needs?

I feel that the project is moving in the correct direction

How much new value has been created since the last week?

The questions are answered by giving a number on a scale from 1 to 6. Where 1 is “not at all” and 6 is “Completely”. This week we scored 18 (maxed out all questions). We might need to tweak questions in the future to get a bit more nuance.

Our last KPI is code coverage, this is all the code that has been tested an therefore completed. This week this is 0, since no code is actually tested.

### **Design decisions and product structure**

* how you ensure code quality and enforce coding standards

By using merge request we make sure that at least two people look at a piece of code before it is merged into the master branch.

This ensures that the code has a higher minimum quality and that the design decisions and workings of each part of the code base are known by at least two people. This also helps with the spotting of bugs and enforces the use of feature branches which modularizes the version control history.

### **Application of Scrum**

* the roles you have used within the team and their impact on your work

We chose to split the team into two subteams working separately with the frontend and the backend. This helped us get a better modularization of work but in turn made us less aware of the progress of the whole project. The scrum master has been helpful in bridging this gap because we have regular meetings and good message flow between teams. Working with a scrum master and two separate teams probably made us have more meetings than normally in the startup phase but we believe this team structure gives us a better effectivity in the long run.

* the agile practices you have used and their impact on your work

We use both a task board and retrospective in the end of a sprint. These tools encourages more discussions, planning and therefore improves our key decision. We have seen that the subteams have been more in sync and key technical decisions could be taken together instead of alone, taking into account the impact of their decision on the other subteam.

* best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

We had to learn a lot of new frameworks, tools and technologies to be able to make the application for our product owner. Because we split our team into two subteams, into frontend and backend, some of the things we had to learned focused on frontend, some focused on the backend and some for both. Git and trello was something everyone had to learn how to use. Some teammates already knew how to use these programs and therefore taught the others in the group. Everyone worked together with git and trello because these were essential for setting up the project and planning it. The frontend subteam also had to learn VS code and frameworks like React that used languages like HTML, CSS and JavaScript. We used Easter to individually learn these through tutorials and examples online, then we used what we learned to teach each other. A similar technique where used for the backend subteam. All teammembers in the backend already had a pretty good understanding of Java and SQL and thus working with it went swimmingly. The only thing that was a little new was to start a Java HTTP server and connect it to a database we had available. There was good information about how to successfully create a server online which we used to our advantage.

* relation to literature and guest lectures (how do your reflections relate to what others have to say?)

We have tried to use literature to know how to work with an agile method instead of a waterfall method. From the Husqvarna guest lecture, we heard that maintaining the same scrum master during the whole project is recommended since changing master between all sprints creates confusion over who is supposed to do what.