Post mortem

Overview

Due to current circumstances regarding Covid-19 at the time the restaurant Bryggargatan aimed to minimize the person to person interactions for their customers as well as employees. They asked us to develop a software solution that would enable customers to order and pay for their meal from their table without having to interact with a waiter/waitress as well as to avoid standing in crowded lines.
Our highest priority was to make the product easy to use for the restaurant's customers as we expected most customers to not be particularly tech savvy.

Details

This project had 3 developers who worked for 7 weeks, approximately 10 hours each per week. See appendix for week by week planning.
We used the agile workflow; Scrum. One Scrum meeting per week.
We used Git for version control.
We used Visual Studio, Brackets, Intellij IDEA, and NeoVim for coding.
We wrote our project in C, C#, Java, HTML, Typescript, and CSS.
We used GIMP to create our mockups at the beginning of the project.

What went right

We did pretty well at dividing tasks into proper, manageable sizes. To have a rigid DoD (definition of done) was good because it gave us a clear goal for a given task and helped the other team members to understand our code, it also improved the readability of the code.
The time estimations given to each task improved each week and felt pretty correct by the second half of the project.
Our team worked together in a common workplace at close to all times which meant we all could ask for help and we all had a pretty good idea of what the others were doing.
Using a real corporeal whiteboard and being able to move around our post-its whenever necessary was amazing for all our Scrum needs.

What went wrong

Communication with our client started off decent but as time went it dwindled. We would not really know if our client actually likes the output we made as we did not communicate our design choices as well as we should have. Essentially we could have let our client in on more choices instead of deciding mostly everything within our group.
We never got a hold of the Swish certificate we needed to complete the project. This may be due to poor communication between us and our client or simply due to time constraints.
We never checked if our C# server code would be compatible with our Linux server which caused problems towards the end of the project ultimately making us have to rewrite the code in Java.
Not assigning a specific time to our Monday Scrum meetings impacted our meetings with the course examinator negatively since we often had not discussed what to bring with us to them.

Warning signs

It took too much time to figure out how to use iZettle and Swish to our advantage, about halfway through the project we should have realized this and considered alternate options.

Risk management

Deciding to use Typescript instead of Javascript was a risk because it was new to us.
We used Grunt to compile our Typescript into Javascript, it took quite a while to set it up and if we would not have figured it out those hours would have been wasted.
We took a risk rewriting our server code from C# to Java to be able to run it on the Linux server.

Mid-project changes

We had decided to try to use iZettle in combination with Swish with our client but it turned out it would not work for our purposes so we switched focus to using only Swish.
Rewrote our payment server from C# into Java.

Conclusions

Things like requesting a Swish certificate from the client takes a lot of time. Considering all of the back and forth communication and the bureaucracy regarding online payment services as well as other time constraints for both parties.
Scrum is an awesome workflow.
Working together in the same location improves motivation drastically which in turn improves efficiency.
Keeping a strict weekly schedule and a soft daily schedule has led to less stress and proper workflow.
Assigning each task a category is not required for a small party of 3 but it is worth considering during larger projects.

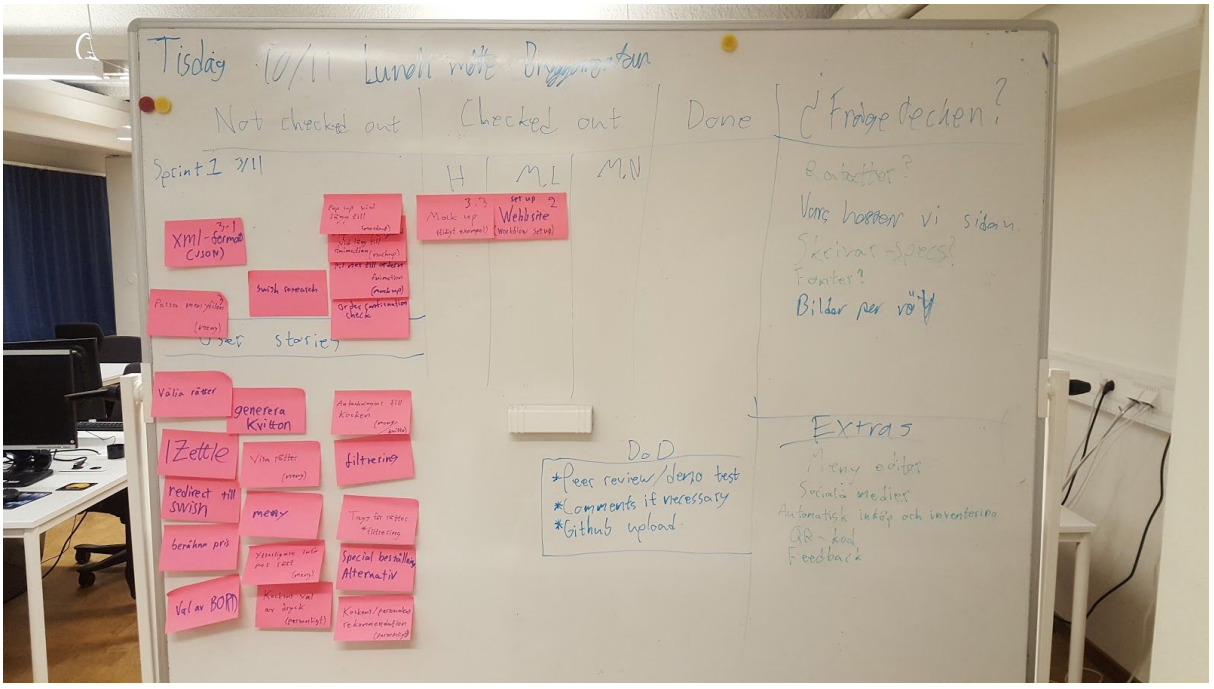
Taking action

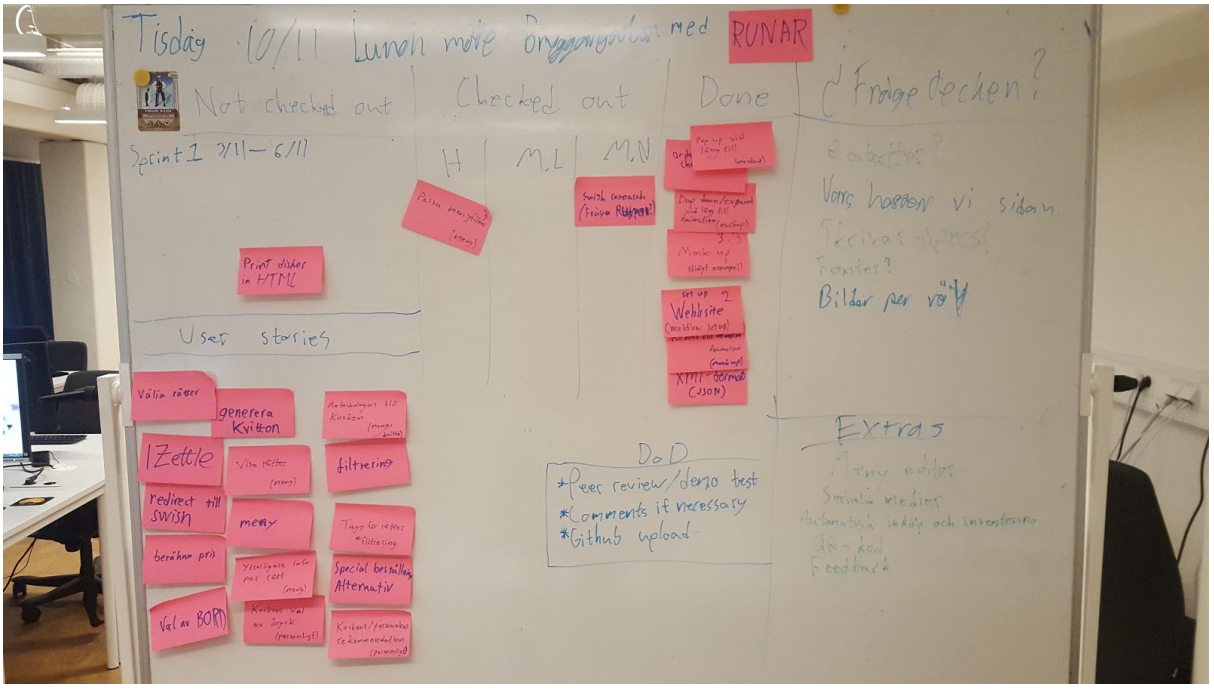
Always check compatibility before starting to write the code.
Define the problem before attempting to fix it.
Continue using Scrum.
Weekly meetings are a must, daily meetings are good but not necessary if there is daily interaction with group members.
Use a common material workplace location.
Whiteboard!
Do not forget to take a break.
When stacking done tasks on the whiteboard always place the post-its vertically while only sticking them onto the whiteboard and not any other notes.

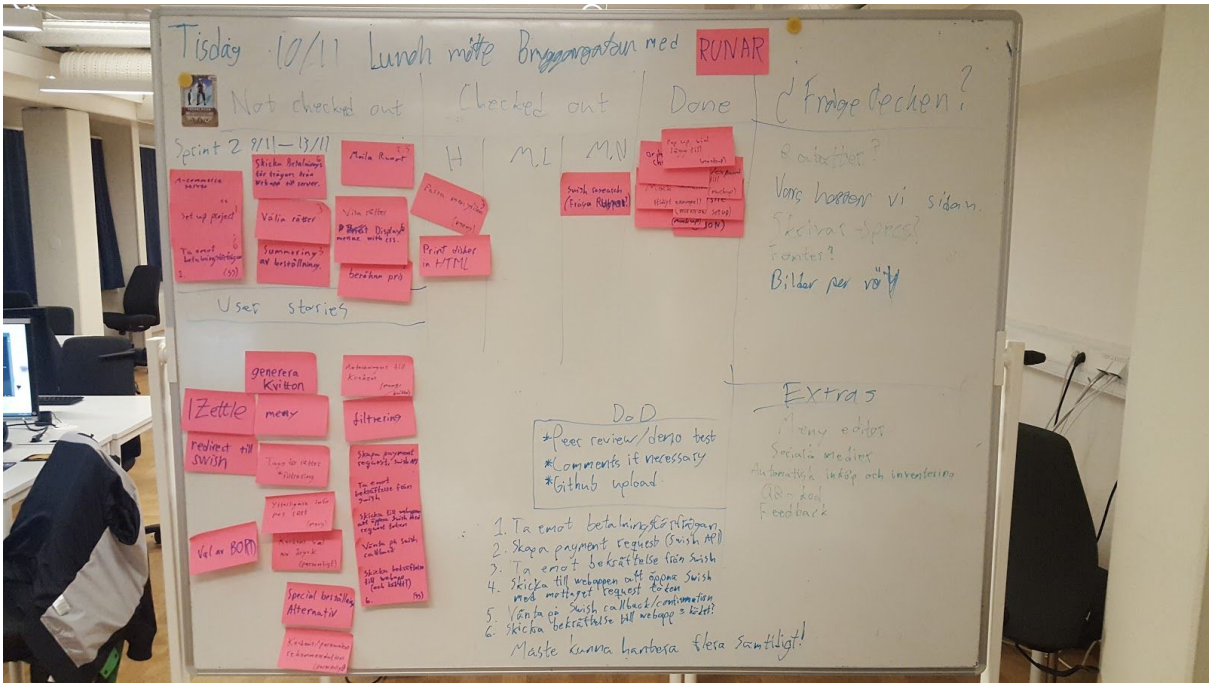
Appendix

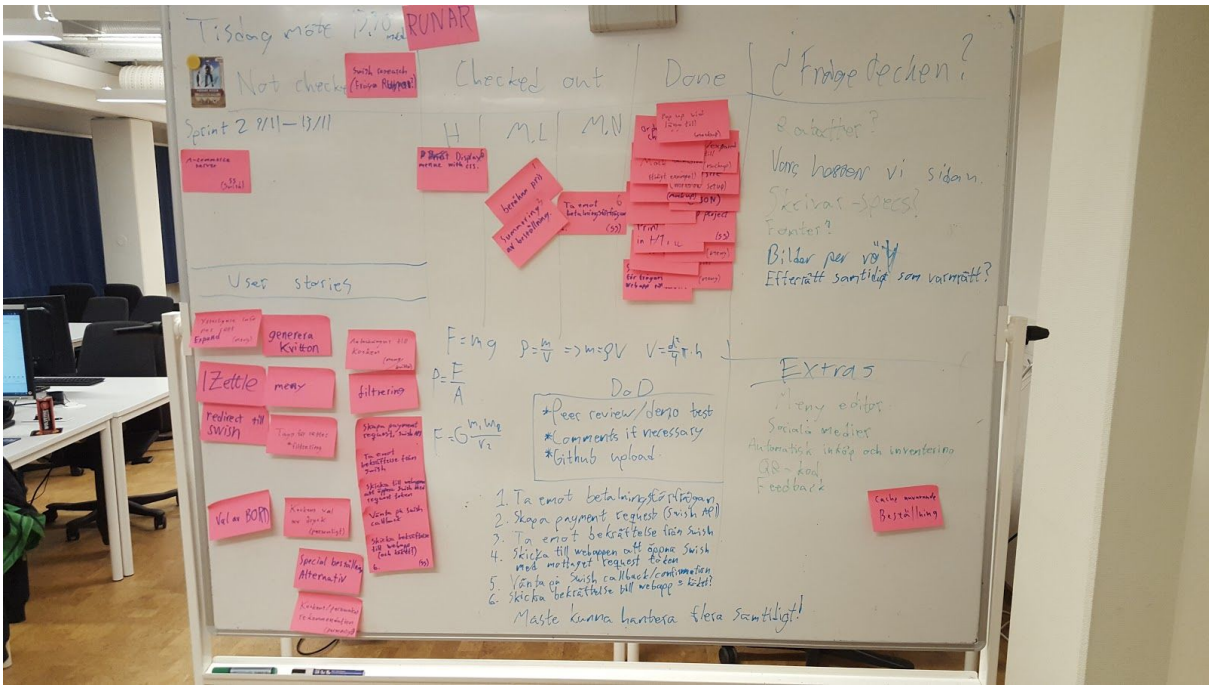
https://archive.gamedev.net/archive/reference/articles/article977.html

Week to week sprint board

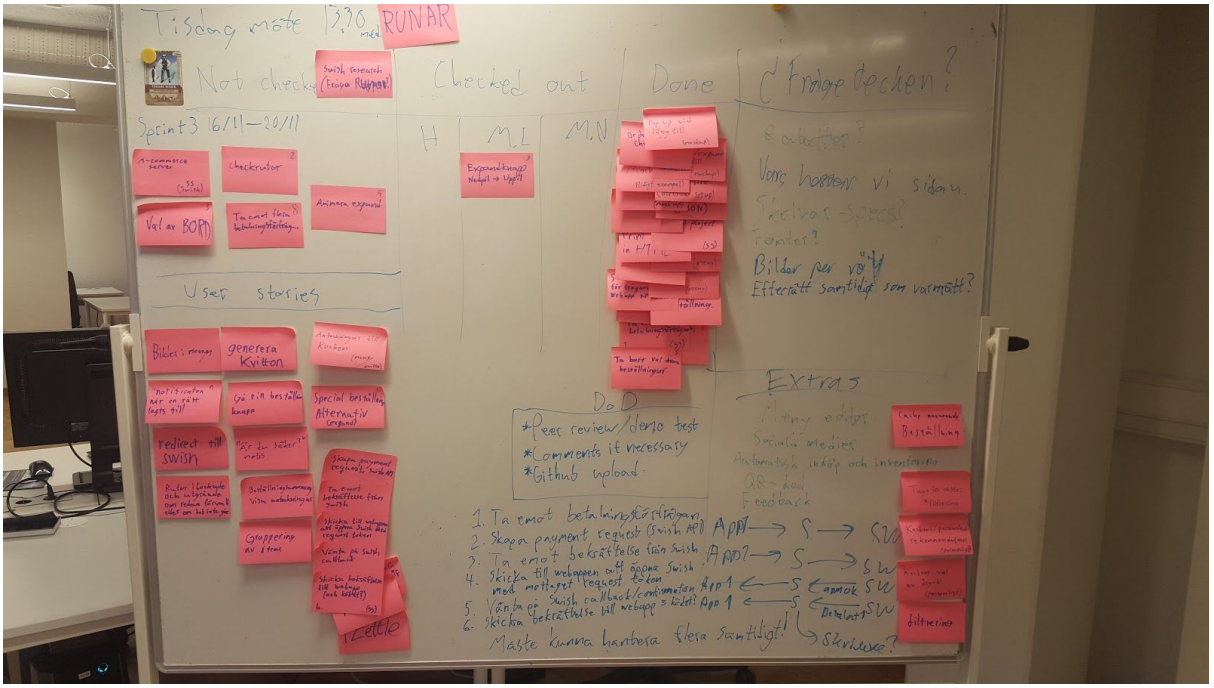


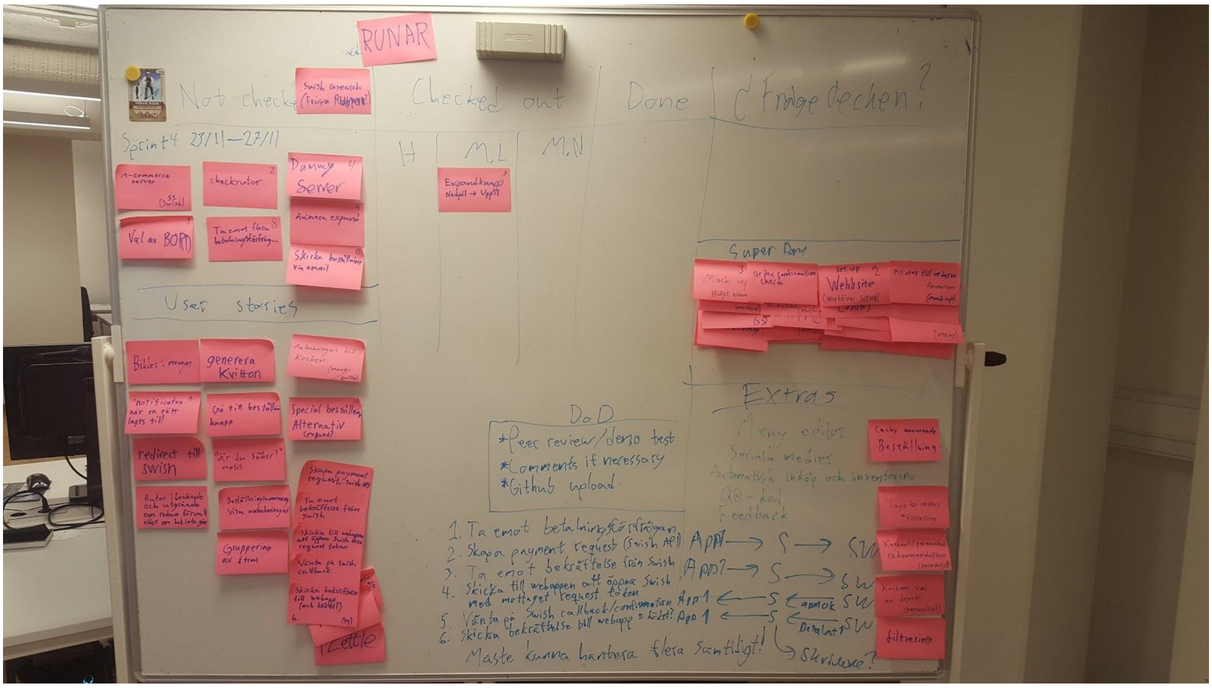


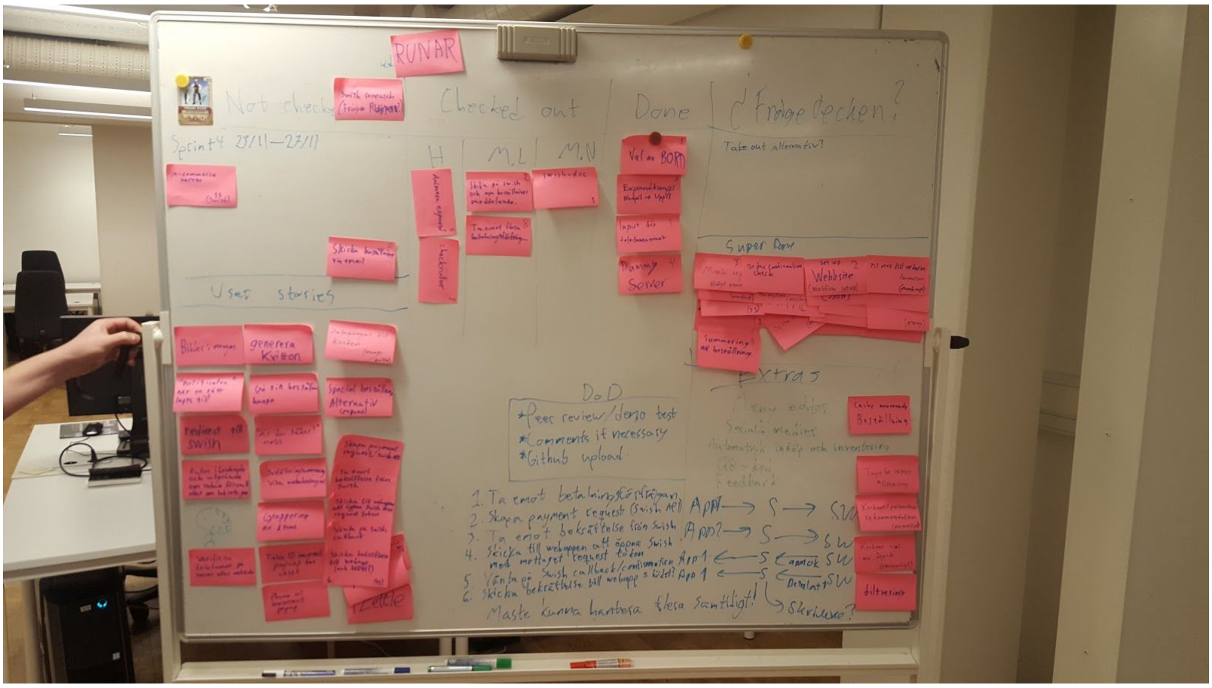

Tisdag 12/11 Lunch möte med RUNAR
Not checked out
Checked out
Done
Sprint 2 9/11–13/11
User stories
Extras
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
Maste kunna hantera flera samtidigt!

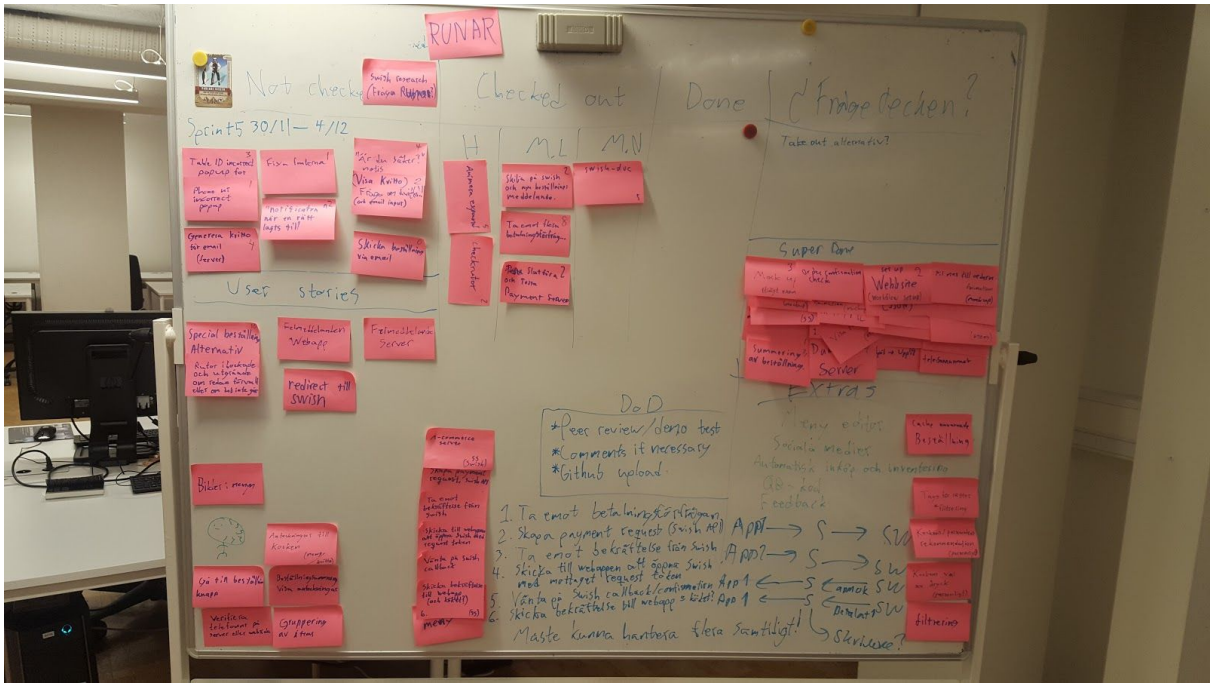

Tisdag möte RUNAR
Not checked out
Checked out
Done
Sprint 2 9/11–13/11
User stories
Extras
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
Maste kunna hantera flera samtidigt!

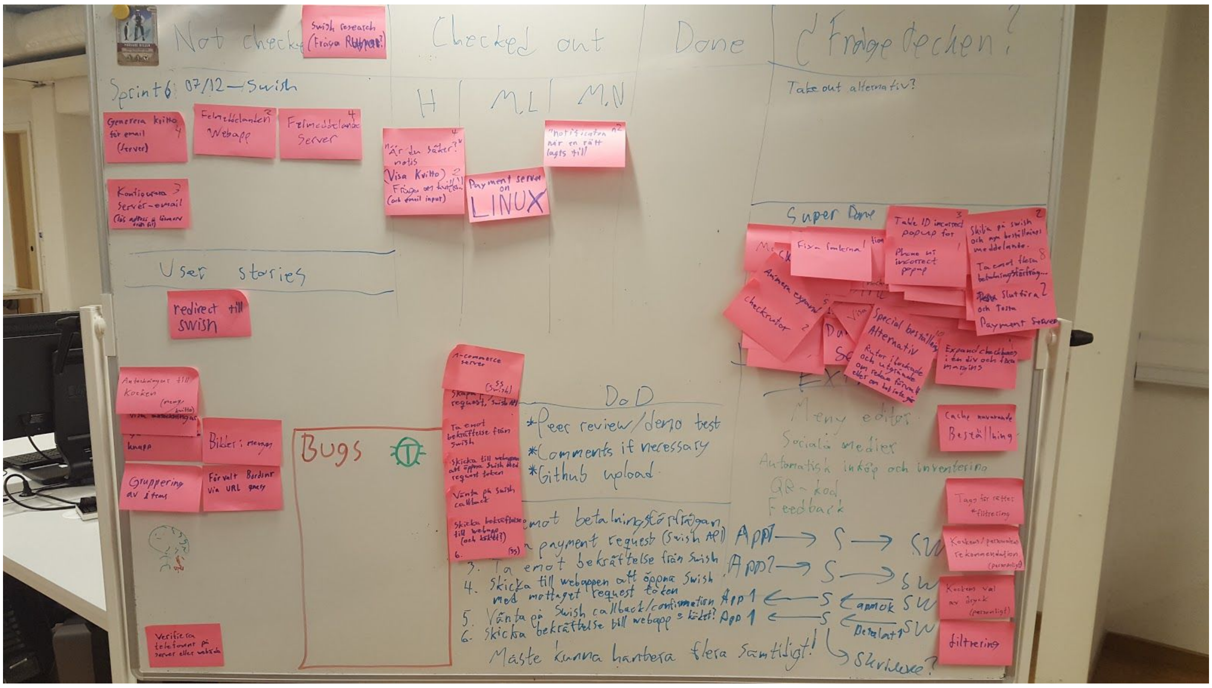

Tisdag möte 13,30 med RUNAR
Not checked
Checked out
Done
Sprint3 16/11—20/11
User stories
DoD
Extras
Måste kunna hantera flera samtidigt!


Tisdag möte 13,30 med RUNAR
Not checked
Checked out
Done
Sprint3 16/11—20/11
User stories
DoD
Extras
Måste kunna hantera flera samtidigt!


RUNAR
Checked out
Done
Sprint4 23/11—27/11
User stories
Super Done
Extras
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
1. Ta emot betalningsförfrågan
2. Skapa payment request (Swish API)
Måste kunna hantera flera samtidigt!

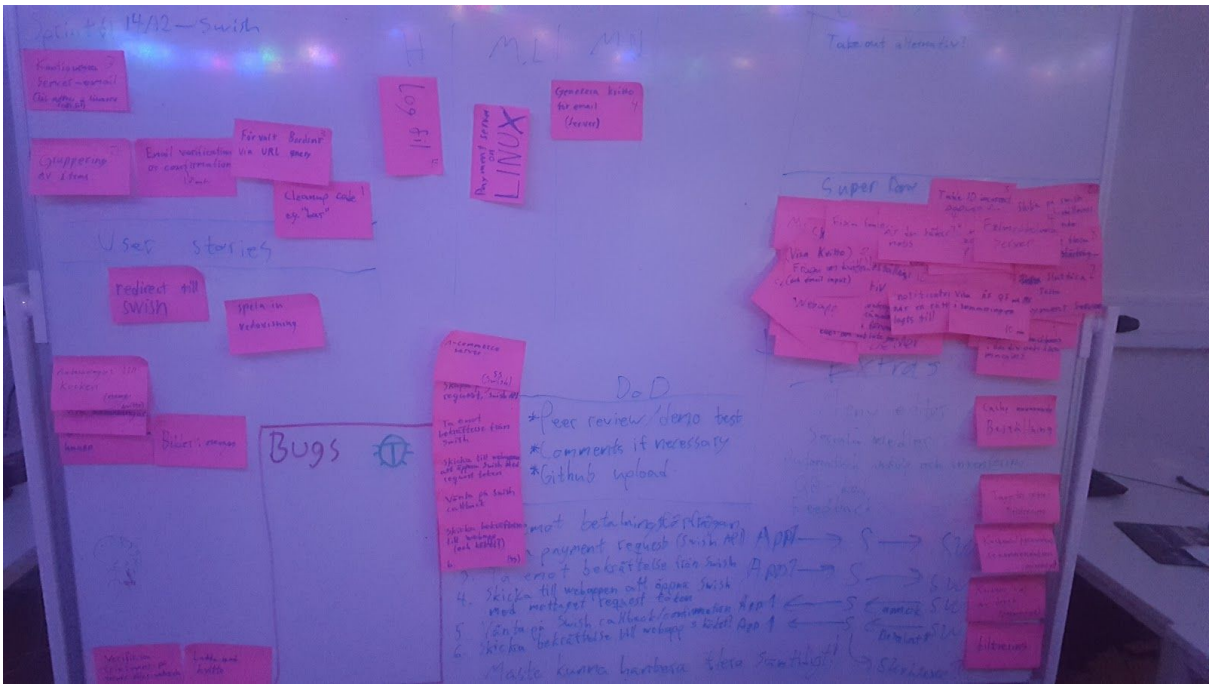

RUNAR
Checked out
Done
Sprint4 23/11—27/11
User stories
Super Done
Extras
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
1. Ta emot betalningsförfrågan
2. Skapa payment request (Swish API)
Måste kunna hantera flera samtidigt!

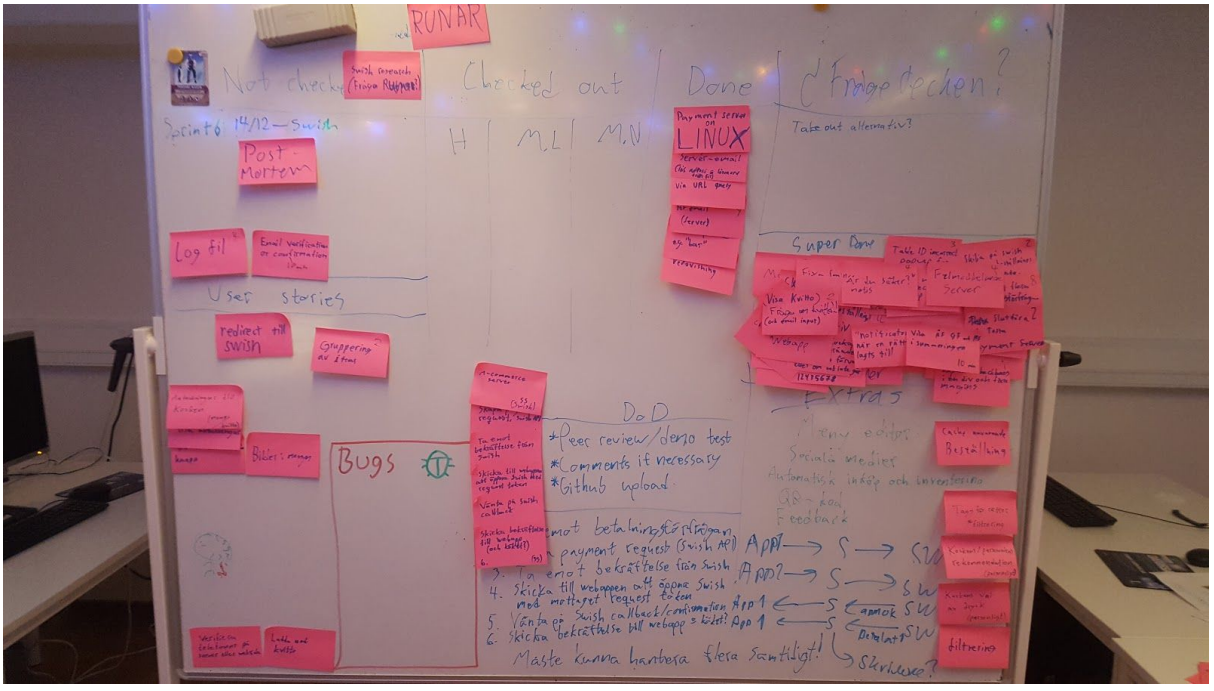

RUNAR
Not checked
Checked out
Done
Sprint 5 30/11 — 4/12
User stories
Super Row
Webbsite
Server
Extras
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
1. Ta emot betalning
2. Skapa payment request (Swish API)
3. Ta emot bekräftelse från Swish
4. Skicka till webappen att öppna Swish med mottaget request token
5. Vänta på Swish callback/confirmation
6. Skicka bekräftelse till webapp = klart!
Maste kunna hantera flera samtidigt!
Skrivelse?
redirect till SWISH


Not checked
Checked out
Done
¿Frågetecken?
Sprint 6: 07/12—Swish
H
M.L
M.N
Take out alternativ?
Swish research (Fråga Runar)
Felmeddelanden Webapp
Felmeddelande Server
"Är du säker?" notis
Payment server on LINUX
User stories
redirect till swish
Super Done
Bugs
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
Extras
Meny editor
Sociala medier
Automatisk inköp och inventering
QR-kod
Feedback
Bilder: menyer
Gruppering av items
Förvalt Bordsnr via URL query
Cache avancerade Beställning
filtrering
Måste kunna hantera flera samtidigt!

RUNAR
Checked out
Done
Super Done
User stories
redirect till swish
Log till
Bugs
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
Måste kunna hantera flera samtidigt!

OBS Sprint 7


Sprint 14/12—Swish
LINUX
User stories
redirect till swish
Bugs
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
Extras


RUNAR
Not checked
Checked out
Done
Sprint 14/12—Swish
Post-Mortem
LINUX
Log fil
User stories
redirect till swish
Gruppering av items
Bugs
DoD
*Peer review/demo test
*Comments if necessary
*Github upload
Extras