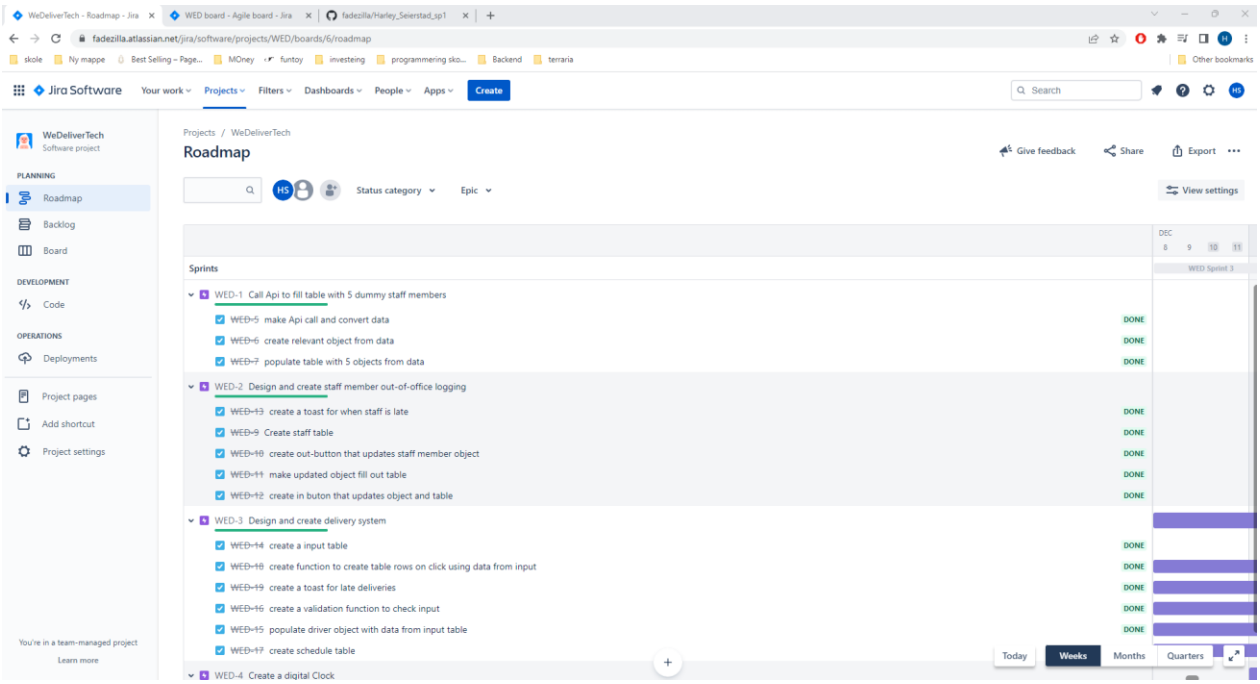
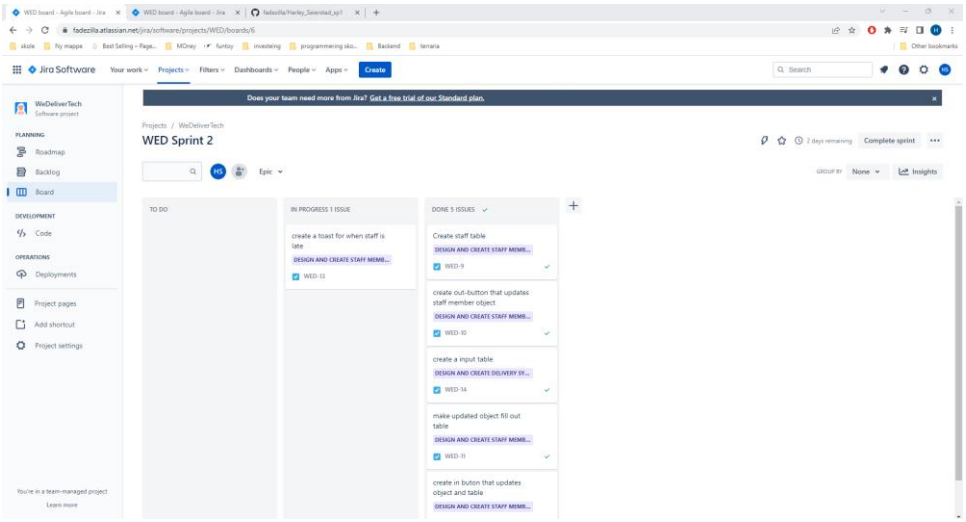
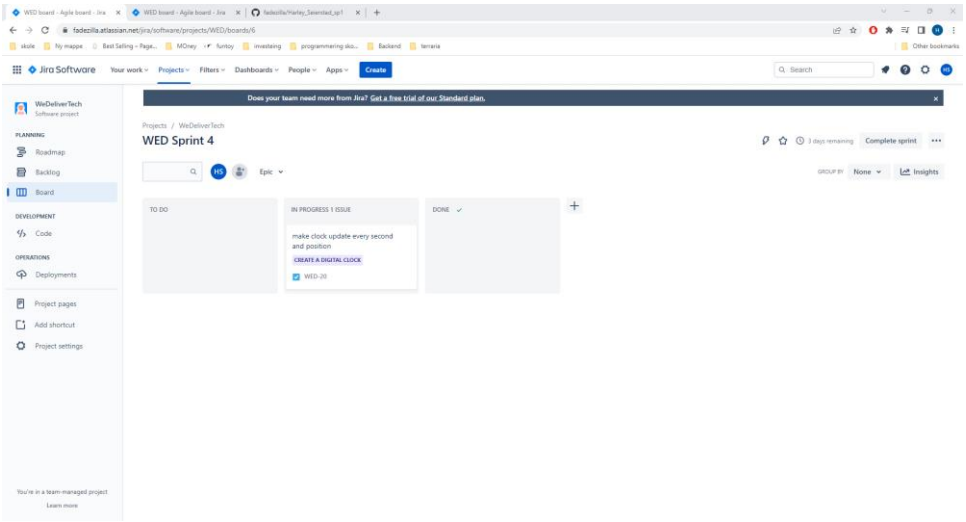
# Reflection paper

For this project I had 4 epics and several tasks within each issue.

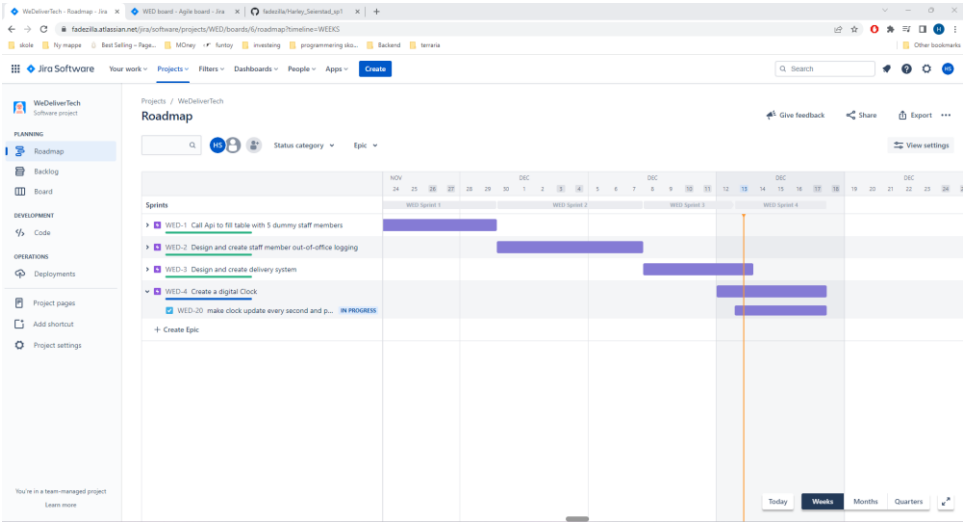



Here`s a picture of my roadmap with the epics and a bunch of tasks. The tasks were mainly big pointers for what needed to be done within the epic. Some examples of these are, have a function that deals with the Api call, use that data to create relevant objects, create schedule tables and so on. Just mainly easily understood tasks, so when I start on one, it won't be any confusion.

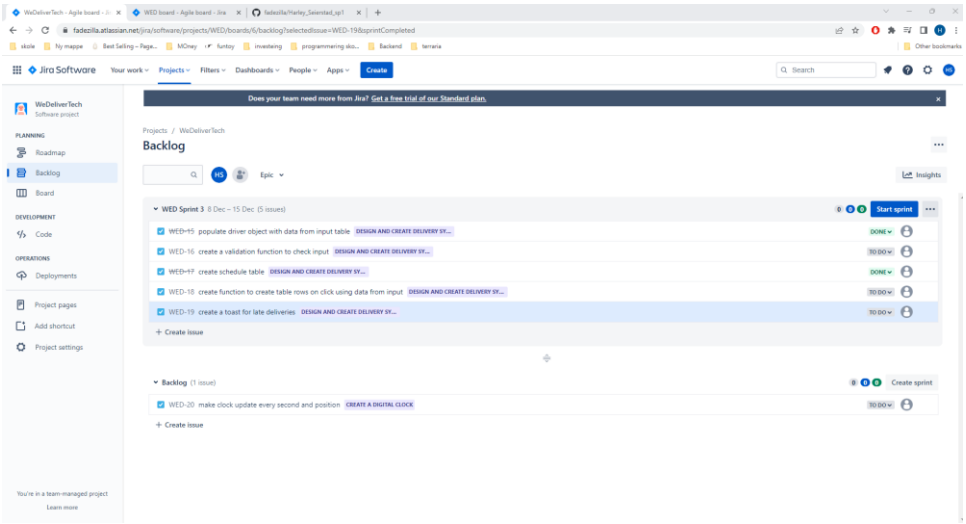





Here are two of my sprints, showing the progress I've done and what I'm currently doing. This is important as I can always tell where I'm at and where I left off and what is needed to be done when I'm back at “work”.




Here`s the roadmap that shows the expected progress I should have, while this is nice to have to see what to expect, it's not quite accurate, as the two first epics were much larger than the last two. But regardless it's nice to have, but as I said, 80% of my time were spent on epic 1 and 2.




Here`s the backlog showing that I'm about to start on sprint 3.

First when I started the project, I had major issues with making everything work, mostly because I misinterpreted the task. I started out with fetching the Api inside the staff member class at first, as well as having the staffMemberIsLate function, while after a week of struggling with this, it actually worked somehow, with the exception that the object creation part of it wasn't used at all, I simply used the functions inside the classes and other than that, the class was mostly useless. After I realized that I did not understand the task properly I started over on this, and used the class for object creation properly using the data we got from the Api call. This was actually rather easy when I did the "right" thing. And instead of having a giant function that does everything inside a class, I instead had more smaller functions which was way neater and easier to read, as well as easier to implement.

After this was up and running, I simply made the Api URL have it so that it only gets 5 users, so if the company at a later stage want more fake users, they can simply increase or decrease that number in the URL and everything will work just the same.

As for the toast that will pop up if after the user is late, was quite easy to bind to a specific table row once you use the object creation using the class. Each new object created by the class was made at the same time and within the same function as the table row being added, so I simply took the input for duration and used this input to calculate the rest of the fields as well as starting a timer on that specific class object, that will fire that specific class object`s staffMemberIsLate toast.

If the user is back before the timer is done, I will simply stop the timeout by calling the specific class object that was created with the same table row. I do this by getting table row index of whatever row I currently have an active class on (I can only have one active class at the time, and whatever row I click on, will get the active class on it).

Epic 3 was kind of rinse and repeat, I re used a lot of the code, there were some differences of course but nothing that caused issues.

Epic 4 was just to create a digital clock, this is very simple as you just get a new Date object, and grab whatever parts you want from that object and display it.

I do wish I tried a bit harder on understanding the task before jumping straight in to coding, as this caused a lot of headaches. But after that initial hiccup and after rereading the task several times, I finally understood what was expected and needed, and this made all things much easier. After I've been struggling, I learned so many different methods and coding tricks that when trying to do it the correct way, it was a breeze.

I will admit that I was a bit lazy to keep updating the Jira board at all times, I struggle with this when its only for me, but next time I will definitely be more obedient with updating it regularly and using it more properly.

But regardless having the Jira board does have it positives, it's very easy to see the progress, and what's missing and what to do next once a problem has been solved or a task have been finished.