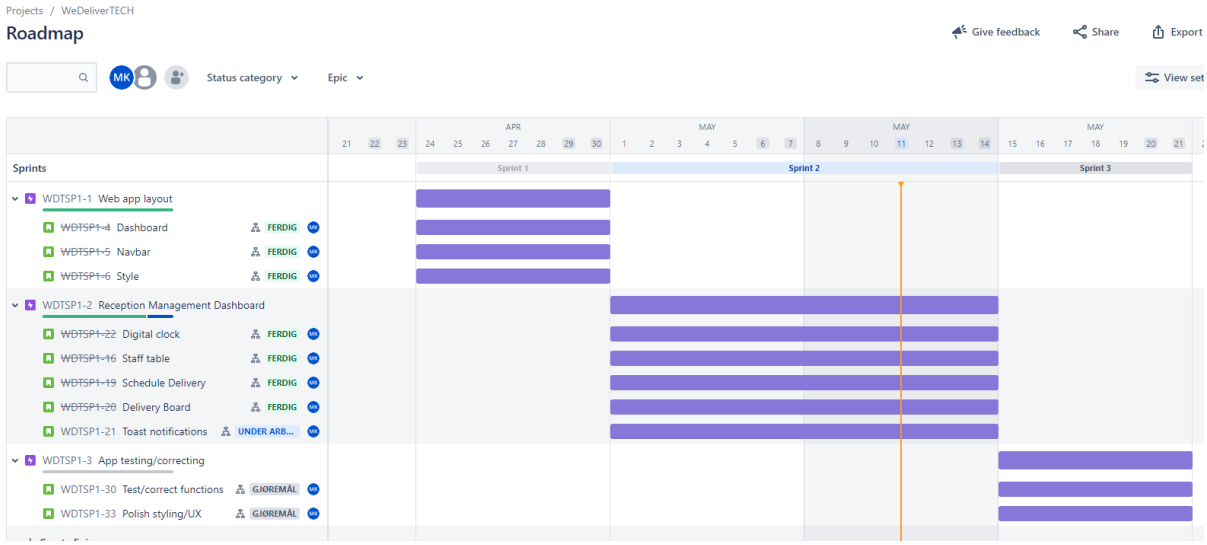
Mari_Kristiansen_sp1

I planned out the project from three epics, which were then used for three sprints. “Web page layout” for the first week – the designing of the webpage. The two next weeks for “Reception Management dashboard” – all the functions to make the webpage useful for the receptionist, and the last week for “App testing/correcting” – to test all functions, fix any “leftover” errors, and polish design.

Full roadmap with epics and stories:



For each epic I wrote down stories which then had subtasks. The stories were essentially all the functions that needed to be completed, for the “Reception Management Dashboard” epic. For the “Web app layout”-epic the stories were how I imagined the web layout would be from the customers mockup - the navbar was one story, the dashboard with the tables and the digital clock was another, and then styling and design was a story.

The subtasks were the stories divided into smaller parts, to help keep track of every required functionality, and making sure that the Company Branding Profile was taken care of.

All subtasks (Jira can’t (to my knowledge) display all subtasks at once):

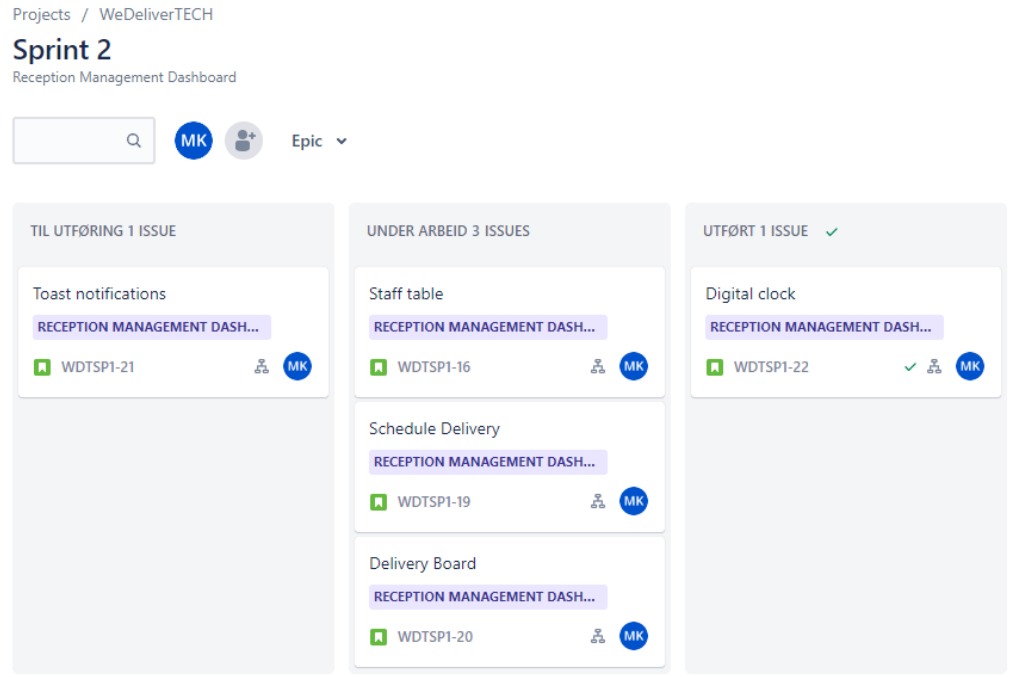
Epic	Story	Subtask
Web app layout	Dashboard	Add tables
		Add buttons
		Add necessary text
	Navbar	Create navbar
		Create dropdown items
	Style	Add style to tables
		Add style to buttons
		Add style to navbar

Epic	Story	Subtask
Reception Management Dashboard	Digital clock	
	Staff table	Populate with API
		Clock in/out with buttons
	Schedule Delivery	Validate input of driver info
	Delivery Board	Display input from "Schedule Delivery"
		Clear selected driver from board
		Confirmation popup when "clear"-button is clicked
	Toast notifications	Toast for late staff member
		Toast for late delivery driver

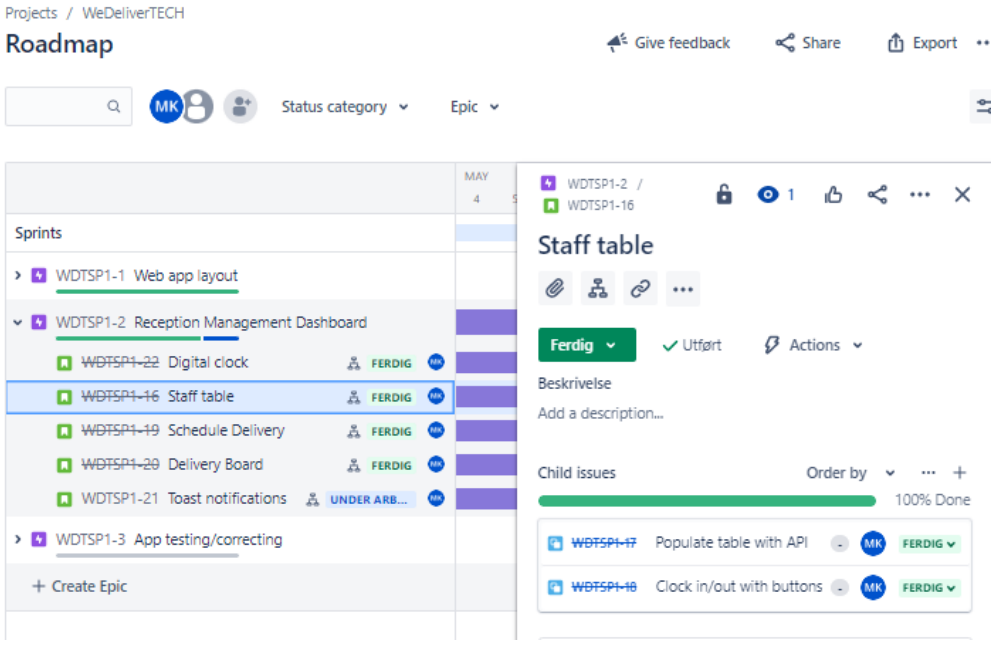
Epic	Story	Subtask
App testing/correcting	Test/correct functions	
	Polish styling/UX	Add/edit styling

Other Jira screenshots:

During sprint 2:



Example of epics – stories – subtasks:



I generally find the design part to be fun and easy to do, so I finished the layout after pretty much two or three days. Had a few challenges, especially with the navbar, but managed to figure out the styling I had in mind. The last week I had time to over the styling, remove unnecessary code, and do final touch-ups.

I understood quickly that the functions would be the most challenging for this project, so I started the sprint early since I finished the first sprint early. It was very overwhelming at first, so I spent a while repeating the PRO-modules and other modules from PRF. After a couple of days I tried to not overthink and tried a “one-thing-at-a-time” approach, basically just as I planned in Jira..

The most challenging functions:

- staffUserGet(): In itself, the API-request and using it to populate the table, was fine to do. I added “/?results=5” to the api-link to get 5 random users. I looped over each staffMember to get the wanted data and used it to create a new Staff object. I created an empty array before the function, to where the objects are pushed. I did the groundwork first, but went back to edit the function here and there, to add eventListener among other things (did this throughout the project as well).

- staffIn() and staffOut() function were next, but had to make a function for selection of a row as well. Therefor I did this almost simultaneously. rowSelected() adds a class to the clicked row which have styling to it. Like the function is now, if the user selects another row, the class is removed, meaning the user can only select one row at a time.
  The staffOut functions works like if a row is selected (only from the staffTable) the user is prompted to enter minutes away. If the input is a number, at least 1 or higher, the function sets the outTime to the current time, and calculates the expectedReturnTime by adding the input minutes to the current time. Depending on the amount of minutes, the expectedReturnTime is displayed in “hr” and/or “min”. The function then finds the staff objects from the staffList array and sets the new values to the object.
  staffIn() was pretty straightforward. The function finds the selected staff object from the array and sets the object’s status, outTime, duration and expectedReturnTime to “In” and empty strings.

- I created the addDelivery() function next, just to make sure that the inputs are displayed in the Delivery Board first, before making the validateDriver() function. I had challenges getting the selected vehicle icon. After trying different things back and forth I ended up setting a value to the icons in the label element, and from that did an if else statement which set the vehicleIcon to whichever vehicle value was selected in the radio and used that in the creation of the td. The function iterates over the driverList array and creates a new row for each driver. I also wanted the input fields in the table to be cleared after the driver’s been added to the Delivery Board. For this I set the checked radio to false, and the other input fields to empty strings.

- Then for the validateDelivery() function I did if statements with regular expressions for each input (see sources used in readme.md). I had some trouble with duplication of the drivers if one of them was cleared from the table, and the user then tried to add drivers. For this I set the deliveryBoardTable.innerHTML to an empty string in the addDelivery function. If all validations are fulfilled, a new DeliveryDriver object is created and added to the driverList array. And finally I set the addDelivery() function to call the validateDelivery() function, and return without proceeding if it’s not fulfilled.

- I thought the toasts would be somewhat ok to do, but with the condition of it to show for a specific Employee if they are late, was a bigger challenge than I anticipated. Had to do a lot of research for this, and I spent a lot of time and frustration with this. With the help of a few sources, I found a code that worked for me. I had to split the expectedReturnTimeValue to hours and minutes and then setHours/Minutes, with hours and minutes as parameter, to the returnTime and then compare it to the current time. I set the functions to run every five seconds, so that if the toast is closed but the employee is not checked in or cleared, the toast will show up again, but hopefully not annoyingly too often. I made two toasts in html, one for staff and one for driver, so I could style them to be on each side of the screen.