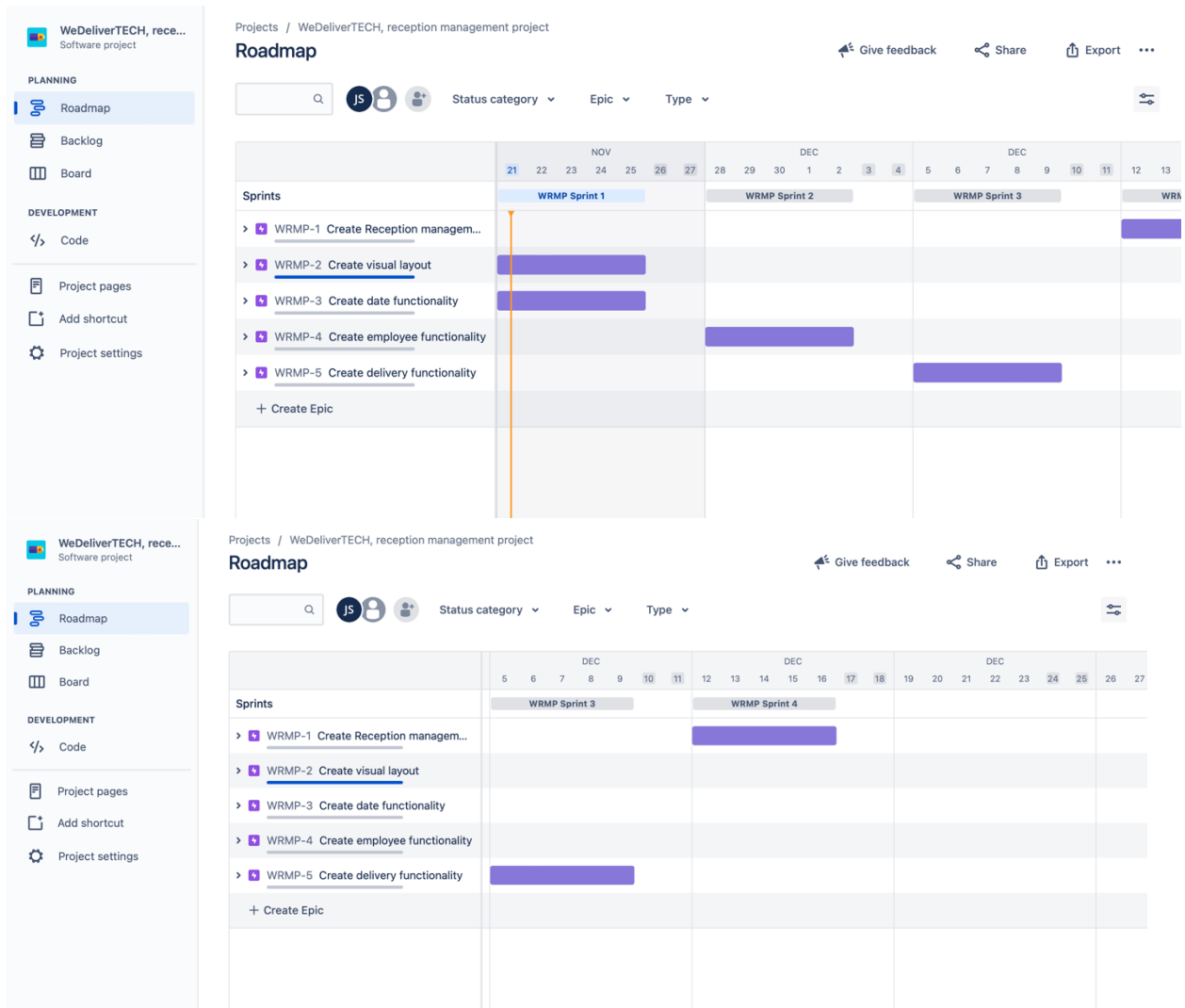


Reflection Report: WeDeliverTECH customer project

Once I received the task, and request for the project from the customer, I began reading through the whole description. I did this to make sure I had a wholistic view of the requested product from the customer. I then read through it again and started to plot in the major epics that I could find from the description:



I broke the task down into five epics:

1. Create reception management tool – as the main overarching epic.
2. Create visual layout – to get started
3. Create date functionality – to begin working with JS once HTML, and CSS was mostly completed.
4. Create employee functionality
5. Create delivery functionality.

I then ordered the epics to the appropriate of four sprints:

The screenshot shows the Jira Backlog for the 'WeDeliverTECH, reception management project'. The left sidebar has a 'PLANNING' section with 'Roadmap' and 'Backlog' (selected), and a 'DEVELOPMENT' section with 'Code'. The main area displays 'WRMP Sprint 1' (21 Nov – 25 Nov, 7 issues) with the goal 'Create HTML and CSS layout, and clock functionality'. The tasks listed are: WRMP-10 (Create webpage for company management, UNDER ARBEID), WRMP-19 (Create HTML Structure, UNDER ARBEID), WRMP-20 (Style with css, UNDER ARBEID), WRMP-22 (create appropriate Js file: wdt_app.js, FERDIG), WRMP-21 (Import Bootstrap and JQuery, GJØREMÅL), WRMP-23 (Create clock functionality, GJØREMÅL), and WRMP-30 (Display time at the bottom: Day, Month, Year, Hour:Minute:Seconds, GJØREMÅL). Each task has a 'JS' icon and a dropdown menu.

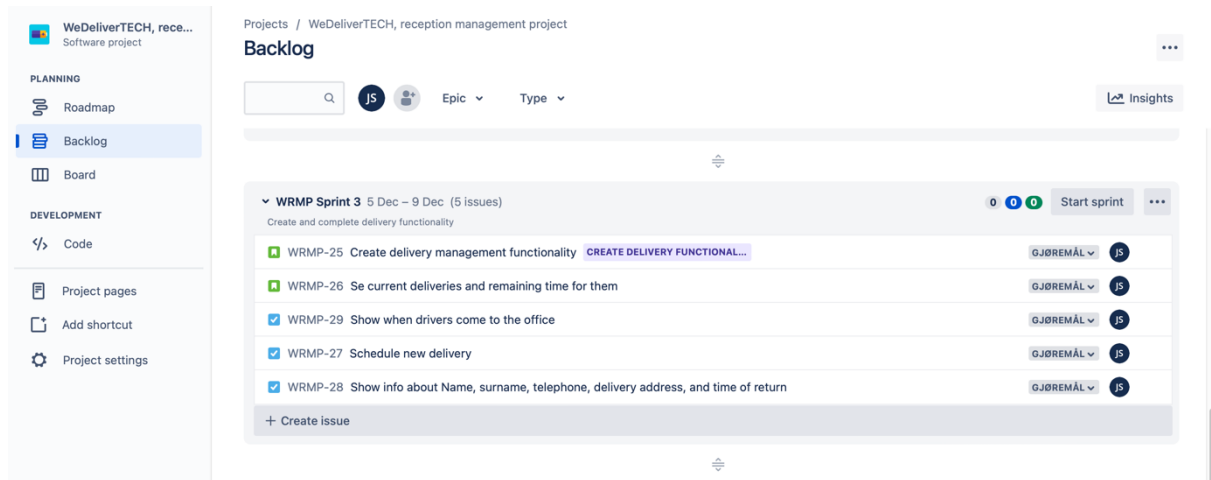
When I created the sprints, I started with creating user stories explaining the customers functionality requests, and what they seek to gain. Then I matched the appropriate epics to the stories and started to fill out task under the appropriate story.

The other sprints look as such:

Sprint 2:

The screenshot shows the Jira Backlog for 'WRMP Sprint 2' (28 Nov – 2 Dec, 11 issues) with the goal 'Create and complete employee functionality'. The tasks listed are: WRMP-11 (Keep track of employees, GJØREMÅL), WRMP-12 (Let employees clock out for meetings, GJØREMÅL), WRMP-35 (Create Employee object, GJØREMÅL), WRMP-13 (Give receptionist overview over employees currently in meeting, GJØREMÅL), WRMP-14 (Let receptionist input time for employee returntime, GJØREMÅL), WRMP-15 (Show leave-time in hours if over 60 minutes, GJØREMÅL), WRMP-16 (Change status from "In" to "out" when clocked out, GJØREMÅL), WRMP-17 (Show toast when employee time exceeds estimate with Image, name, and amount of time from the..., GJØREMÅL), WRMP-18 (Let receptionist call employee, GJØREMÅL), WRMP-24 (Let in-button clear time and set status to "in", GJØREMÅL), and WRMP-34 (Get random employee with Random API, GJØREMÅL). Each task has a 'JS' icon and a dropdown menu. A tooltip is visible over WRMP-18, showing 'Show toast when employee time exceeds estimate with image, name, and amount of time from the office'.

Sprint 3:

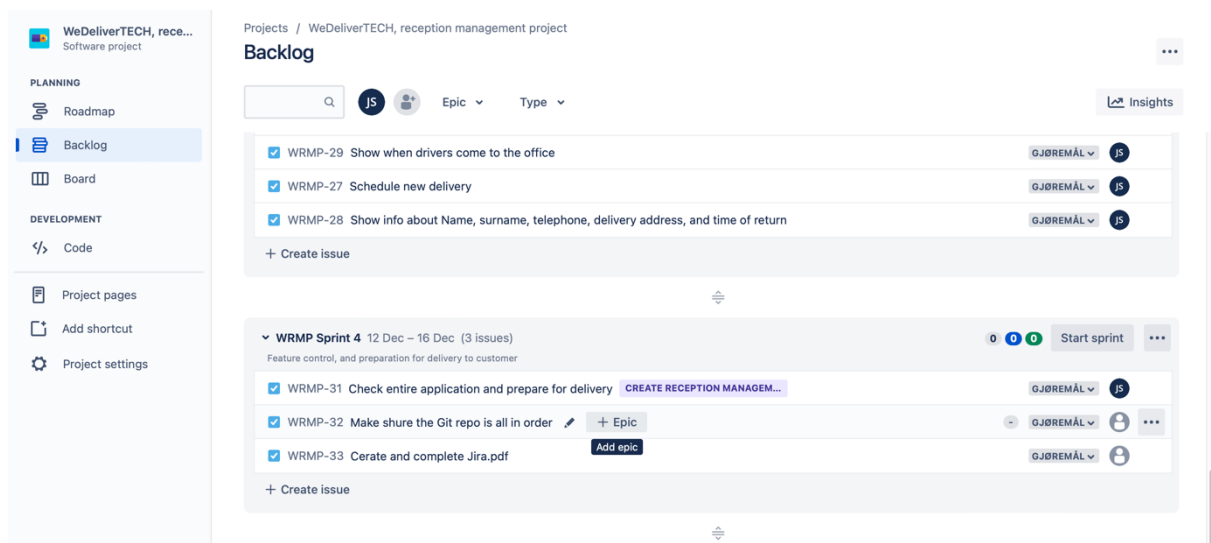


The screenshot shows the Jira interface for the 'WeDeliverTECH, reception management project'. The left sidebar is set to 'Backlog'. The main area displays 'WRMP Sprint 3' (5 Dec - 9 Dec) with 5 issues. The issues are:

- WRMP-25: Create delivery management functionality (GJØREMÅL, JS)
- WRMP-26: Se current deliveries and remaining time for them (GJØREMÅL, JS)
- WRMP-29: Show when drivers come to the office (GJØREMÅL, JS)
- WRMP-27: Schedule new delivery (GJØREMÅL, JS)
- WRMP-28: Show info about Name, surname, telephone, delivery address, and time of return (GJØREMÅL, JS)

Each issue has a 'GJØREMÅL' label and a 'JS' assignee. A '+ Create issue' button is at the bottom.

Sprint 4:



The screenshot shows the Jira interface for the 'WeDeliverTECH, reception management project'. The left sidebar is set to 'Backlog'. The main area displays 'WRMP Sprint 4' (12 Dec - 16 Dec) with 3 issues. The issues are:

- WRMP-29: Show when drivers come to the office (GJØREMÅL, JS)
- WRMP-27: Schedule new delivery (GJØREMÅL, JS)
- WRMP-28: Show info about Name, surname, telephone, delivery address, and time of return (GJØREMÅL, JS)

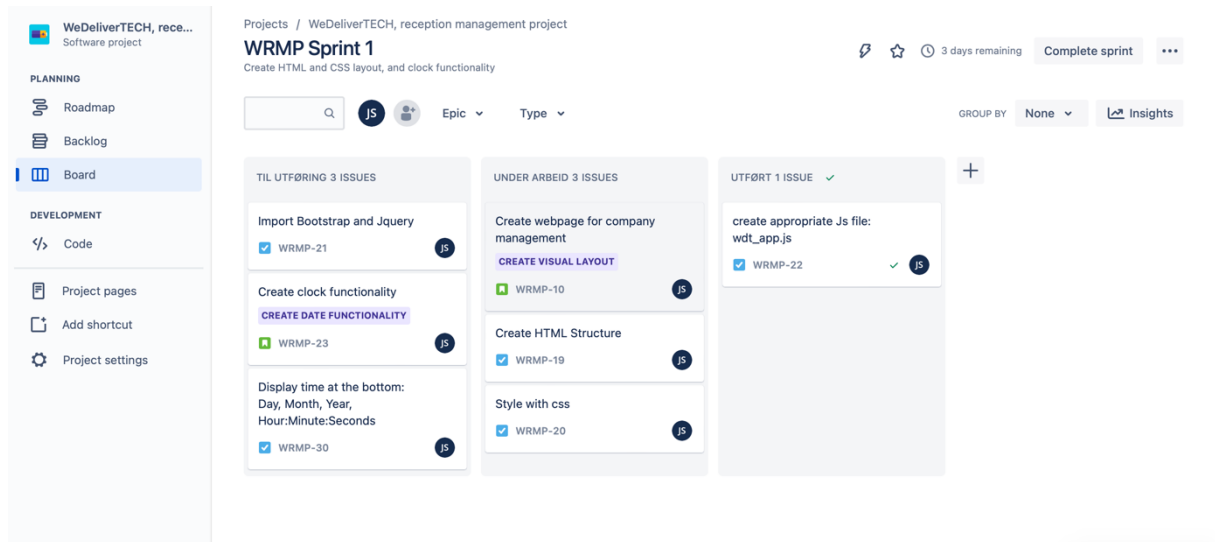
Below these issues is a '+ Create issue' button. The section for 'WRMP Sprint 4' (12 Dec - 16 Dec) has 3 issues:

- WRMP-31: Check entire application and prepare for delivery (GJØREMÅL, JS)
- WRMP-32: Make shure the Git repo is all in order (GJØREMÅL, JS)
- WRMP-33: Cerate and complete Jira.pdf (GJØREMÅL, JS)

Each issue has a 'GJØREMÅL' label and a 'JS' assignee. A '+ Create issue' button is at the bottom.

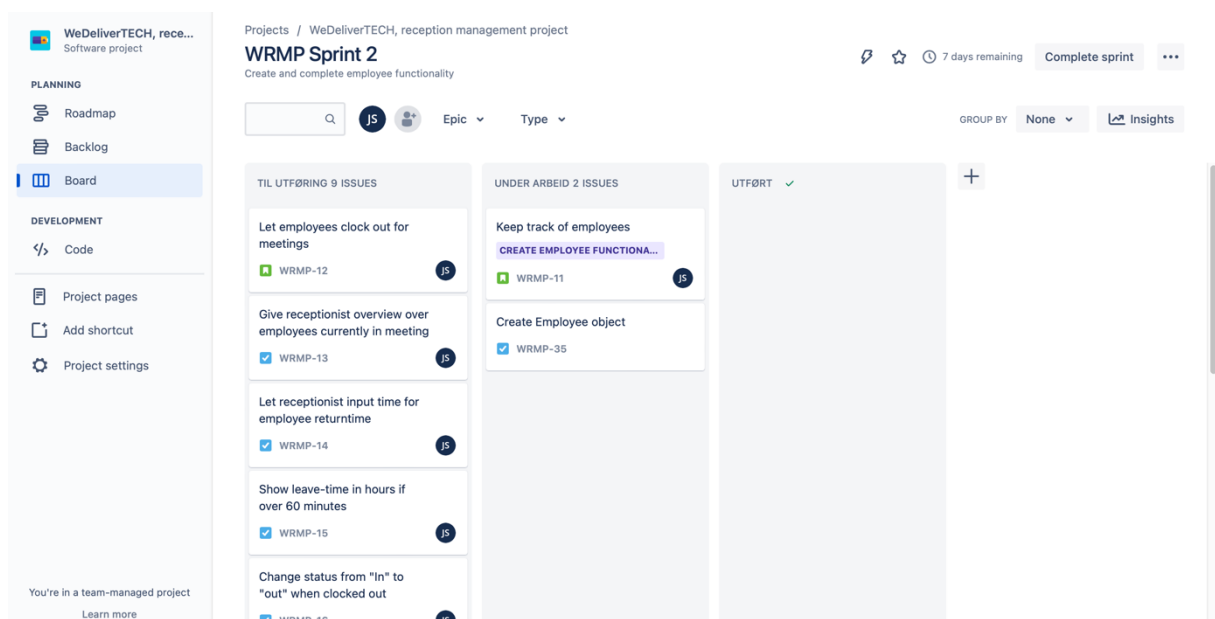
The final sprint will be a quality-assurance sprint. Here I will go over the entire application, making sure everything works, and prepare the product for delivery to the customer. The first of all the epics will be the last to finish, since it has the whole application as its scope.

When I started the first sprint, the tasks would automatically show up on my board:



I would start by moving over the user story at the top of the board. Then I would move tasks correlating to that story over and complete them one after another. Like the example you see here, “create HTML structure, and “Style with css” go hand in hand and are on the board together. I would also move over “import bootstrap and jQuery,” as bootstrap is needed for the css styling.

The first sprint would be completed on time, and I would move on the second sprint as such:



The most difficult part of the application was getting the ability to modify the correct staffObject when calling staffOut() and staffIn(). I ended up doing so by pushing the objects into an array when clicking the correlating image, and modifying the objects inside of that array, then calling a function which updated the correct data on the object. I quickly realized that the key to the entire project was operating not on the table, but on the objects itself, and then simply displaying the updated objects in the table. Most of the operations on the objects where simple, apart from having multiple timers run at once, and not doubling the current timers running, when setting a new one. Once again, the solution was making sure the changes and the timers were set to the objects themselves, so they would not interfere with each other.

The same issue also came up when I came to the deliveries. The best solution I found in this case was to number the table-row. Using slice, I could get the last character of the table-row id, which would correlate to the appropriate object in the array displayDeliveries, which contained all the current deliveries. To validate the time, I compared the selected time of return with the clock functionality on the top of the JavaScript file. To make sure the validation was going, I set an interval, and bound it to the object, and stored the value of the interval in the object itself. so that they would constantly check the time. When the time was out, I would clear the interval inside the object method, with use of the TimerID variable in the object itself to clear out the appropriate interval.

Overall, this was a great challenge. The two main issues that I learned a lot about was operation on object, and time functionality. I have learned a lot from this and will be looking forward to doing task like this in the future.