Reflection report

Documenting progress and decisions made throughout the semester project.

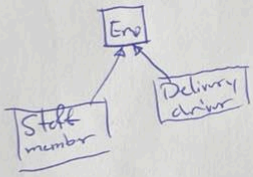
Ever since I saw the “Semester project “ listed in the “Study plan”, I was both curious and excited for what we would be assigned. Waking up early that day and checking our tasks was a little scary. I know we were given four weeks for the task, so I knew it had to be somewhat comprehensive.

After reading the task and its requirements, I quickly noted down my first thoughts for how to develop this project. My initial notes looked like this:

Semester project

- Alerts
  - SweetAlert2
- Toasts
  - jQuery

- Staff table
- Schedule delivery table
- Delivery table



- Navbar

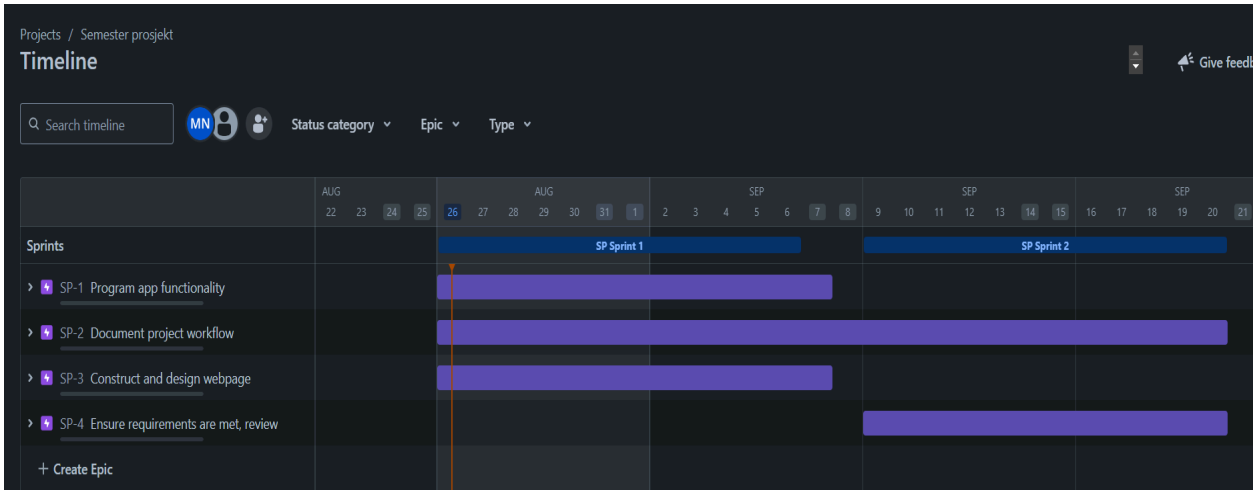
I jumped right into Jira and created a project for planning. Understanding that scrum is a valuable tool, I started using it immediately for my own project management. It was of course a requirement as well, but using it was also in my own best interest.

It set the project duration to four weeks, matching the SP1 duration. I created four epics as seen in the screenshot below. I created two sprints with two weeks duration each. Details are shown in the screenshot below.

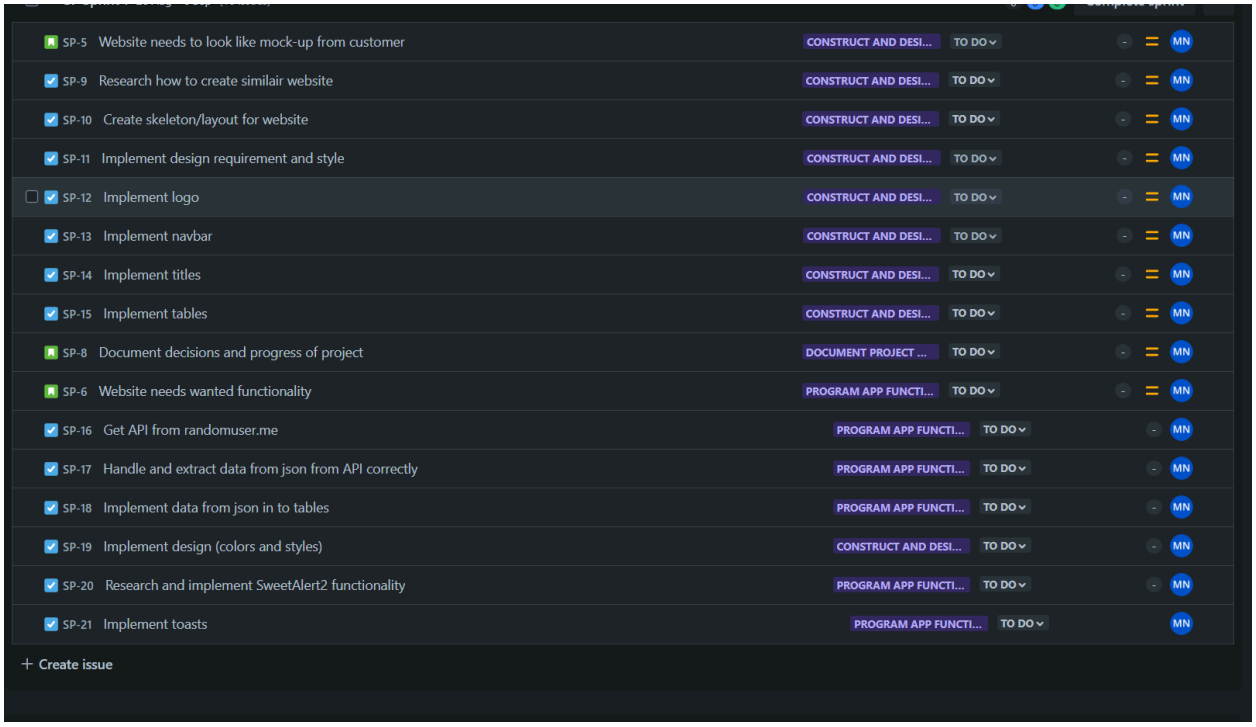
I created these issues because I thought four different epics fit the project nicely.

- Functionality
- Design
- Documentation
- Quality assurance

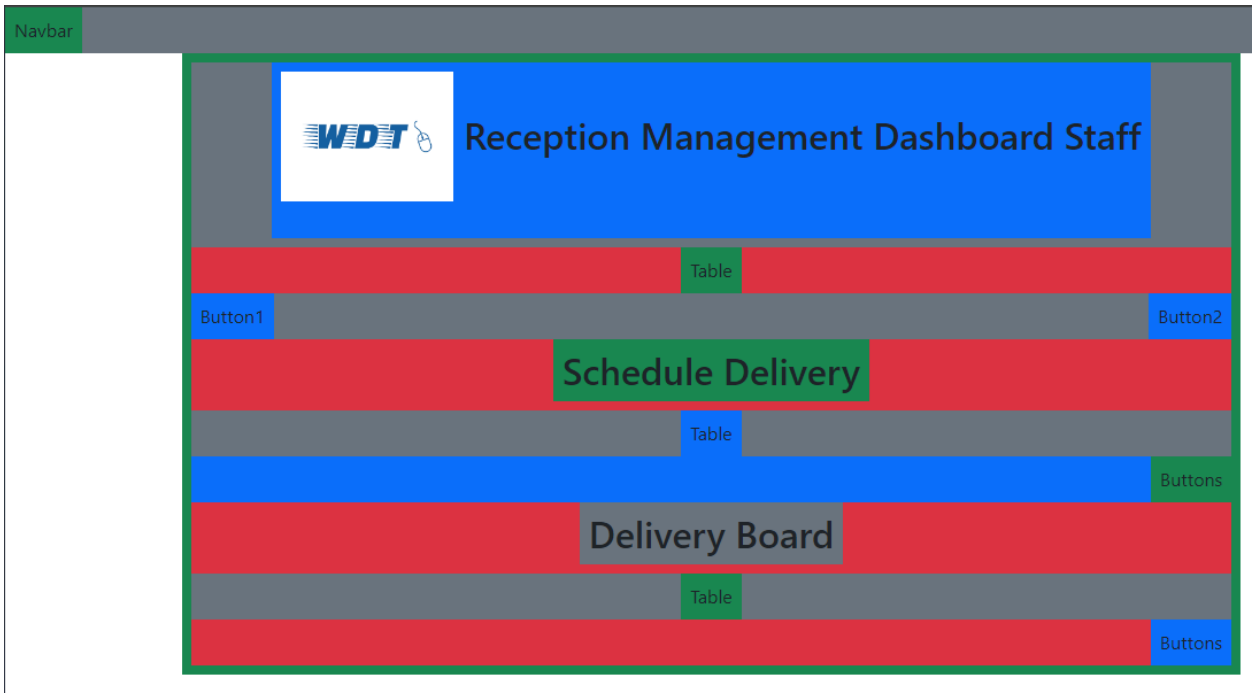
Screenshots and elaborations on the progress of the project will follow below, all the way till page 18.

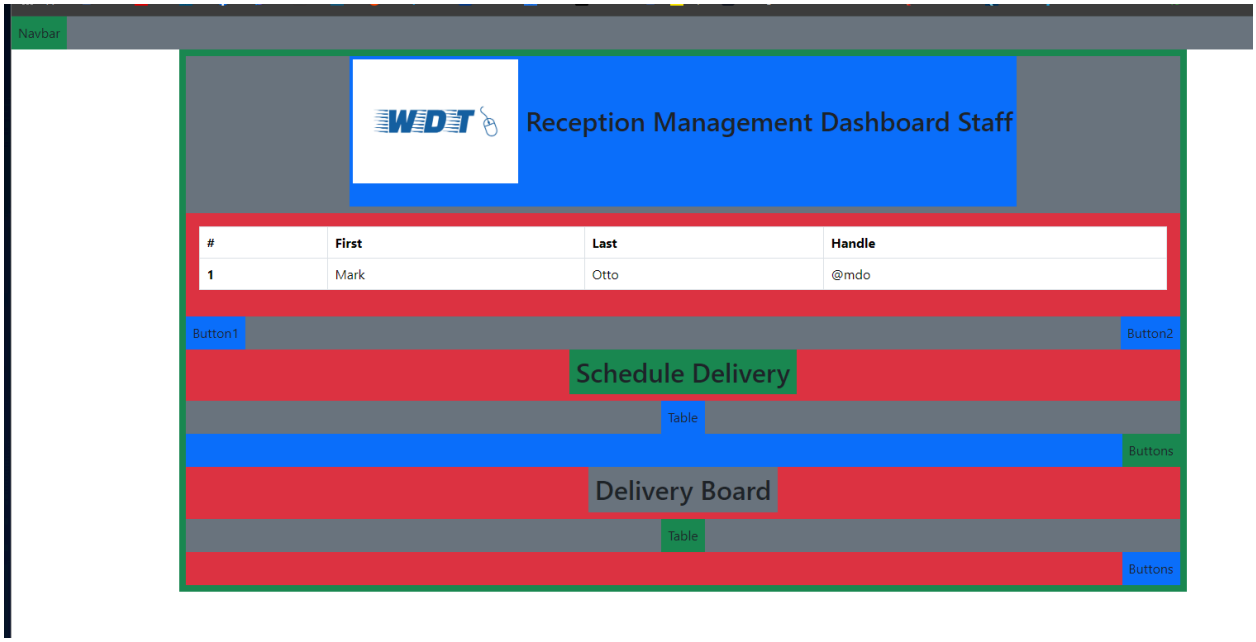


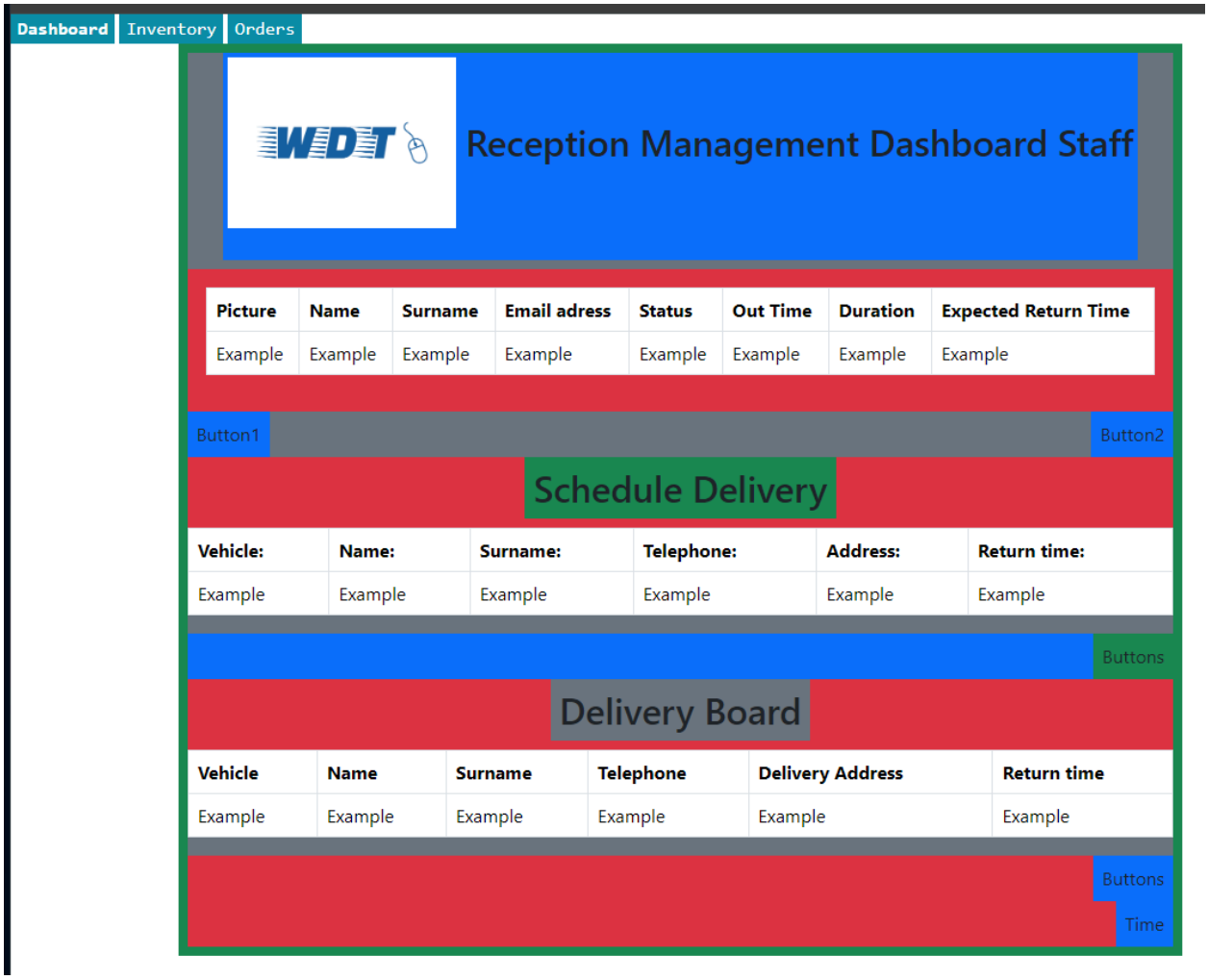
I created a lot of tasks and stories that came into mind. Some of the tasks I created were vague and some were specific. The important thing was to actually note them down and get an overview of the tasks I had in front of me.



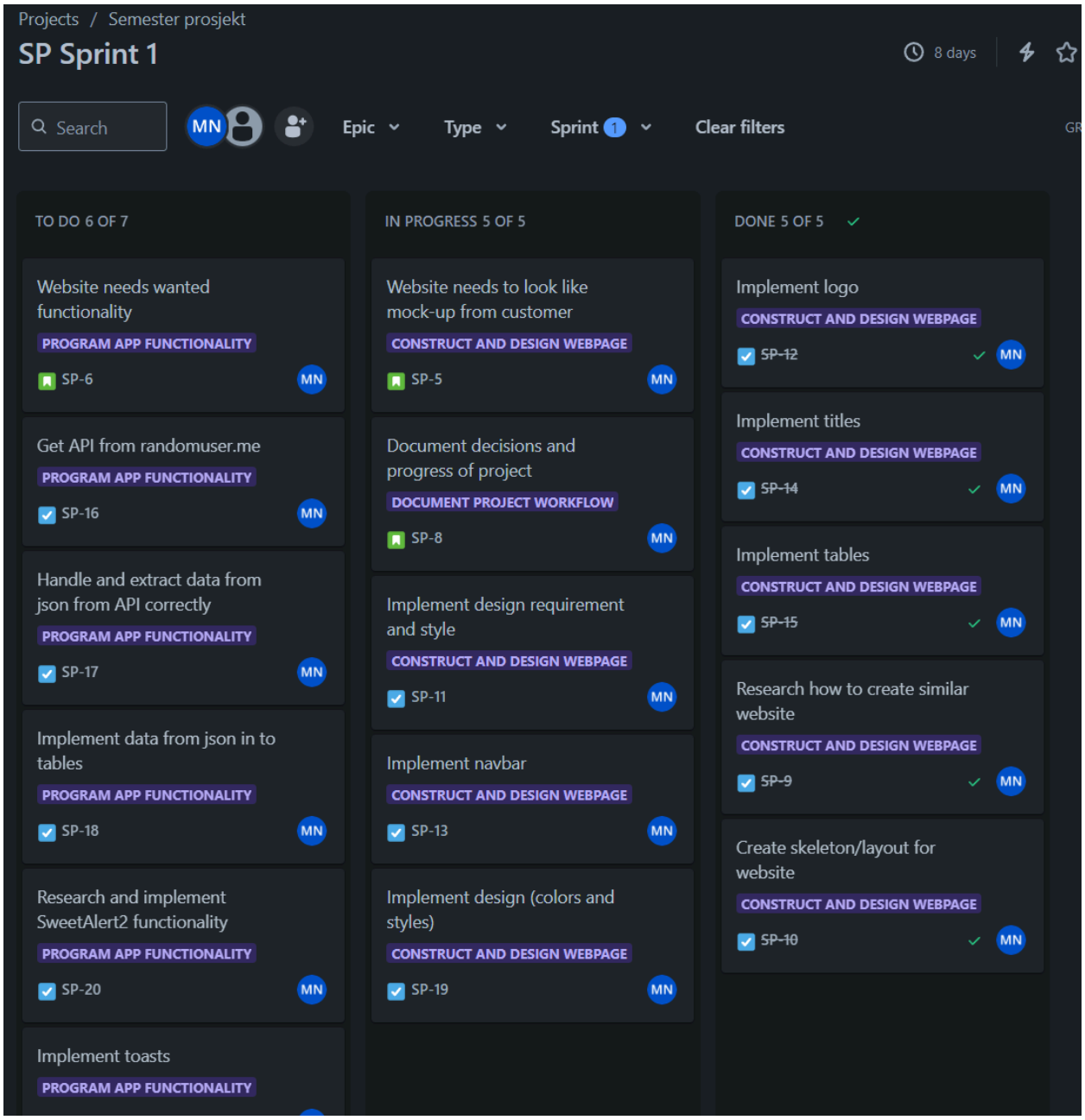
The task I started with was the design, because it felt easier and it would be nice to have to progress in something I can actually see and “touch”. This was done on day 1 and day 2.




Navbar
Reception Management Dashboard Staff
First Last Handle
1 Mark Otto @mdo
Button1
Button2
Schedule Delivery
Table
Buttons
Delivery Board
Table
Buttons


Dashboard
Inventory
Orders
Reception Management Dashboard Staff
Picture Name Surname Email adress Status Out Time Duration Expected Return Time
Example Example Example Example Example Example Example Example
Button1
Button2
Schedule Delivery
Vehicle: Name: Surname: Telephone: Address: Return time:
Example Example Example Example Example Example
Buttons
Delivery Board
Vehicle Name Surname Telephone Delivery Address Return time
Example Example Example Example Example Example
Buttons
Time

Here is a screenshot of my jira board on day 2 at 13:41, with some screenshots of the progress. I colourized all the divs to make it easier for me to see which div was on top of which.



Picture	Name	Surname	Email add
	Dennis	Nicholson	dennisN@examp
	Ivan	Garza	ivan.garza@exam

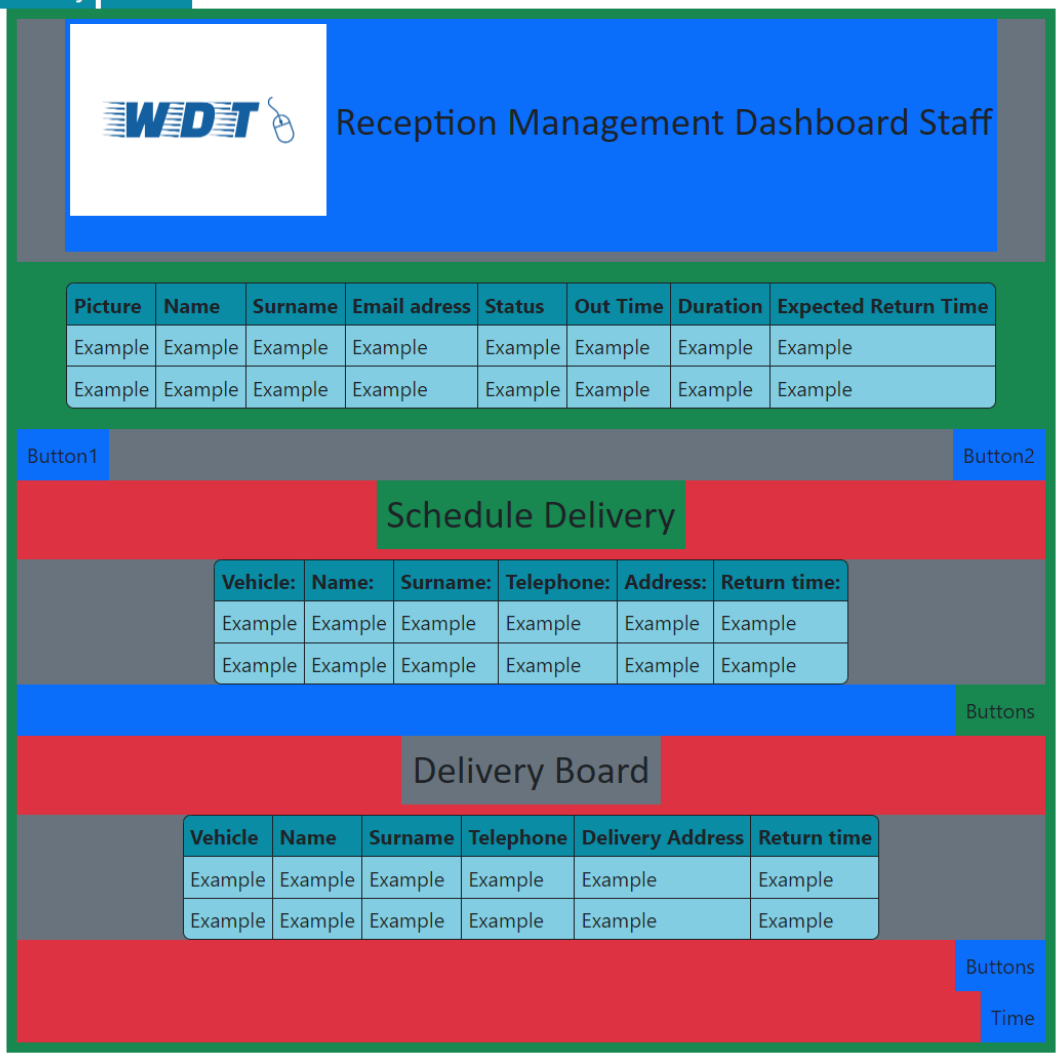
Out

Schedu

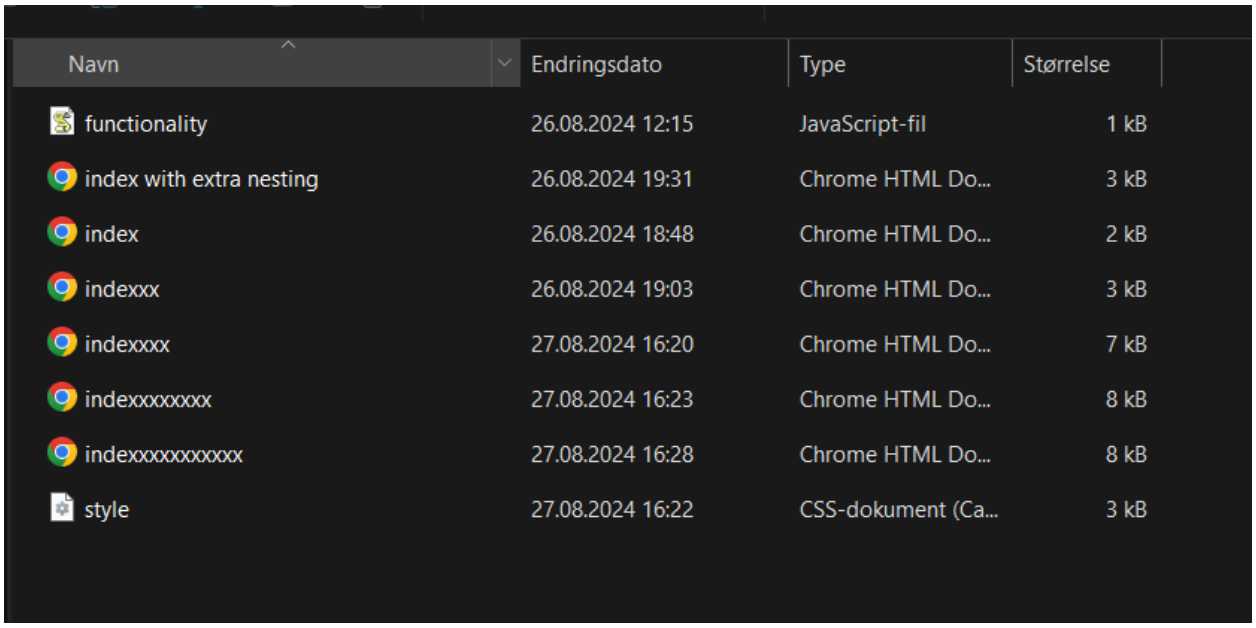
Vehicle:	Name:	Surname:	Te
_____	_____	_____	____

Deliv

Dashboard | Inventory | Orders

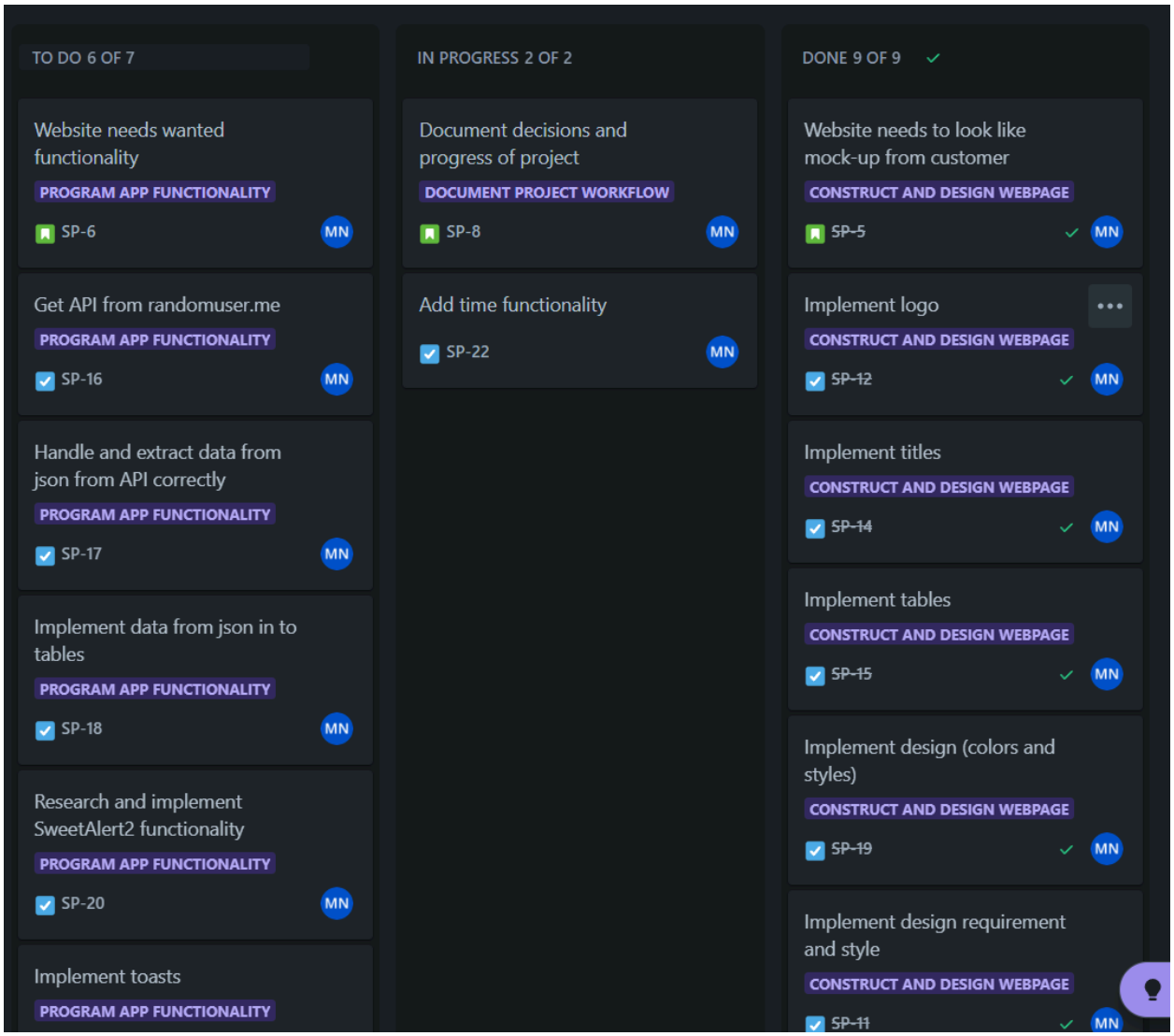


I created multiple index.html files to allow for risk free experimentation

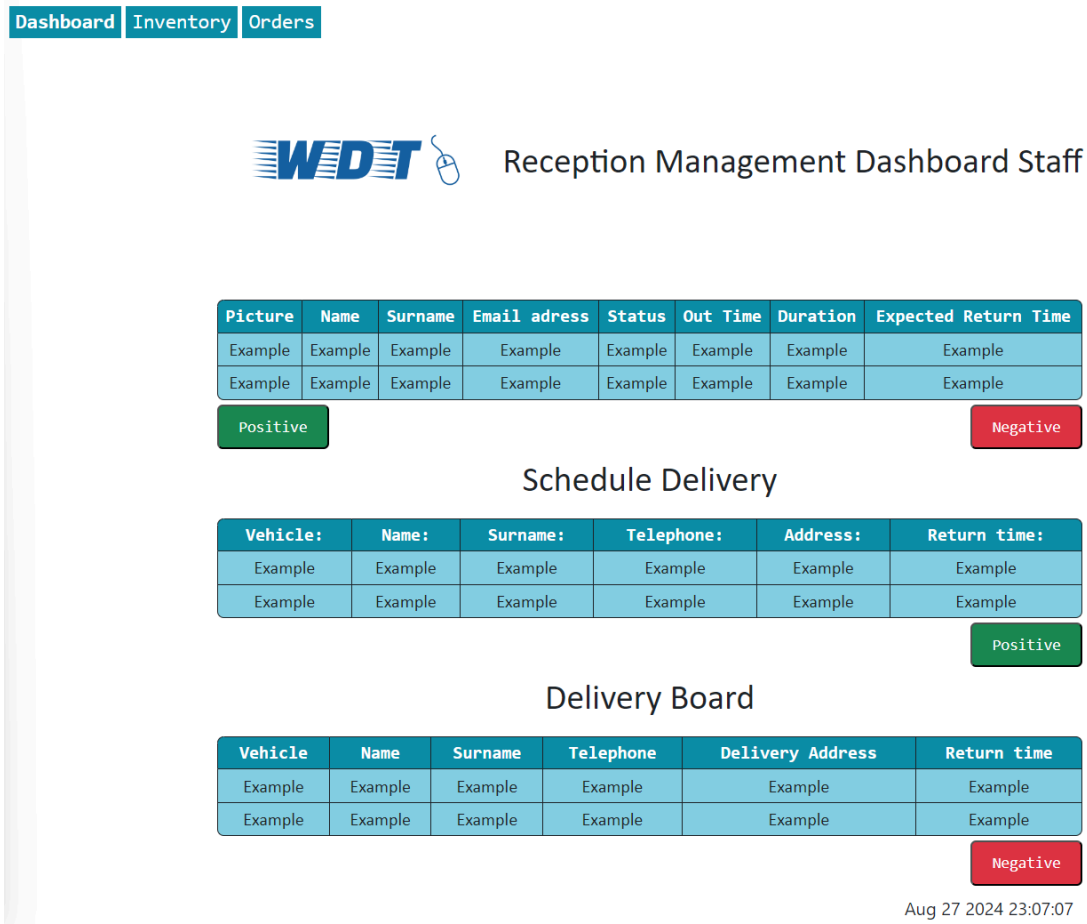


Navn	Endringsdato	Type	Størrelse
functionality	26.08.2024 12:15	JavaScript-fil	1 kB
index with extra nesting	26.08.2024 19:31	Chrome HTML Do...	3 kB
index	26.08.2024 18:48	Chrome HTML Do...	2 kB
indexxx	26.08.2024 19:03	Chrome HTML Do...	3 kB
indexxxx	27.08.2024 16:20	Chrome HTML Do...	7 kB
indexxxxxxxx	27.08.2024 16:23	Chrome HTML Do...	8 kB
indexxxxxxxxxx	27.08.2024 16:28	Chrome HTML Do...	8 kB
style	27.08.2024 16:22	CSS-dokument (Ca...	3 kB

Day 2 at 18:11. 9 tasks done.



Day 2 at 23:07. Functioning time display.



Day 3 at 15:24. Progress on functionality.

Dashboard | Inventory | Orders

Reception Management Dashboard Staff

Picture	Name	Surname	Email adress	Status	Out Time	Duration	Expected Return Time
Example	Example	Example	Example	Example	Example	Example	Example
Example	Example	Example	Example	Example	Example	Example	Example
	Jayanth	Chiplunkar	jayanth.chiplunkar@example.com				
	Joona	Kivi	joona.kivi@example.com				
	Theo	Hughes	theo.hughes@example.com				
	Jaime	Calvo	jaime.calvo@example.com				
	Rose	Roberts	rose.roberts@example.com				

Positive Negative

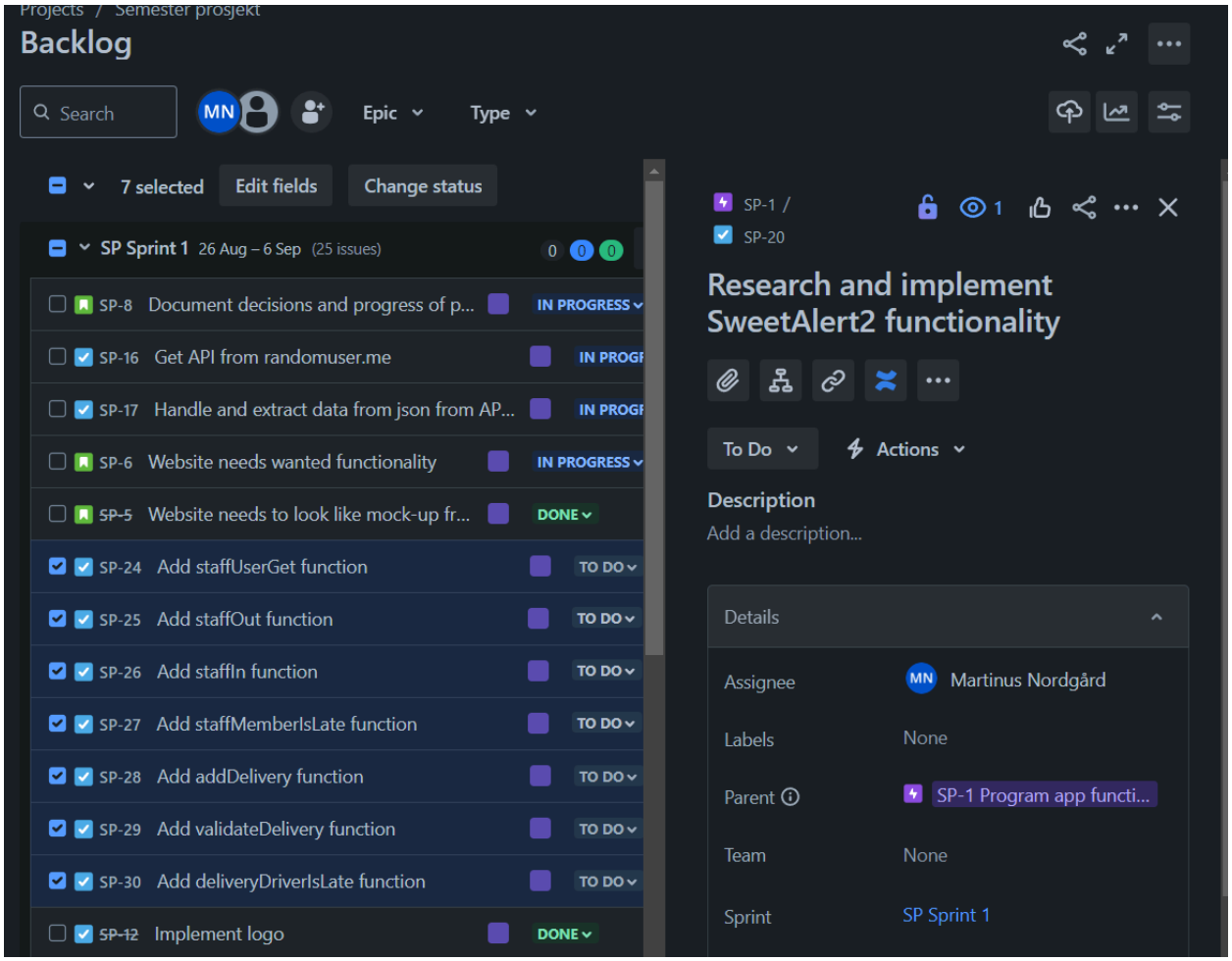
Schedule Delivery

Vehicle:	Name:	Surname:	Telephone:	Address:	Return time:
Example	Example	Example	Example	Example	Example
Example	Example	Example	Example	Example	Example

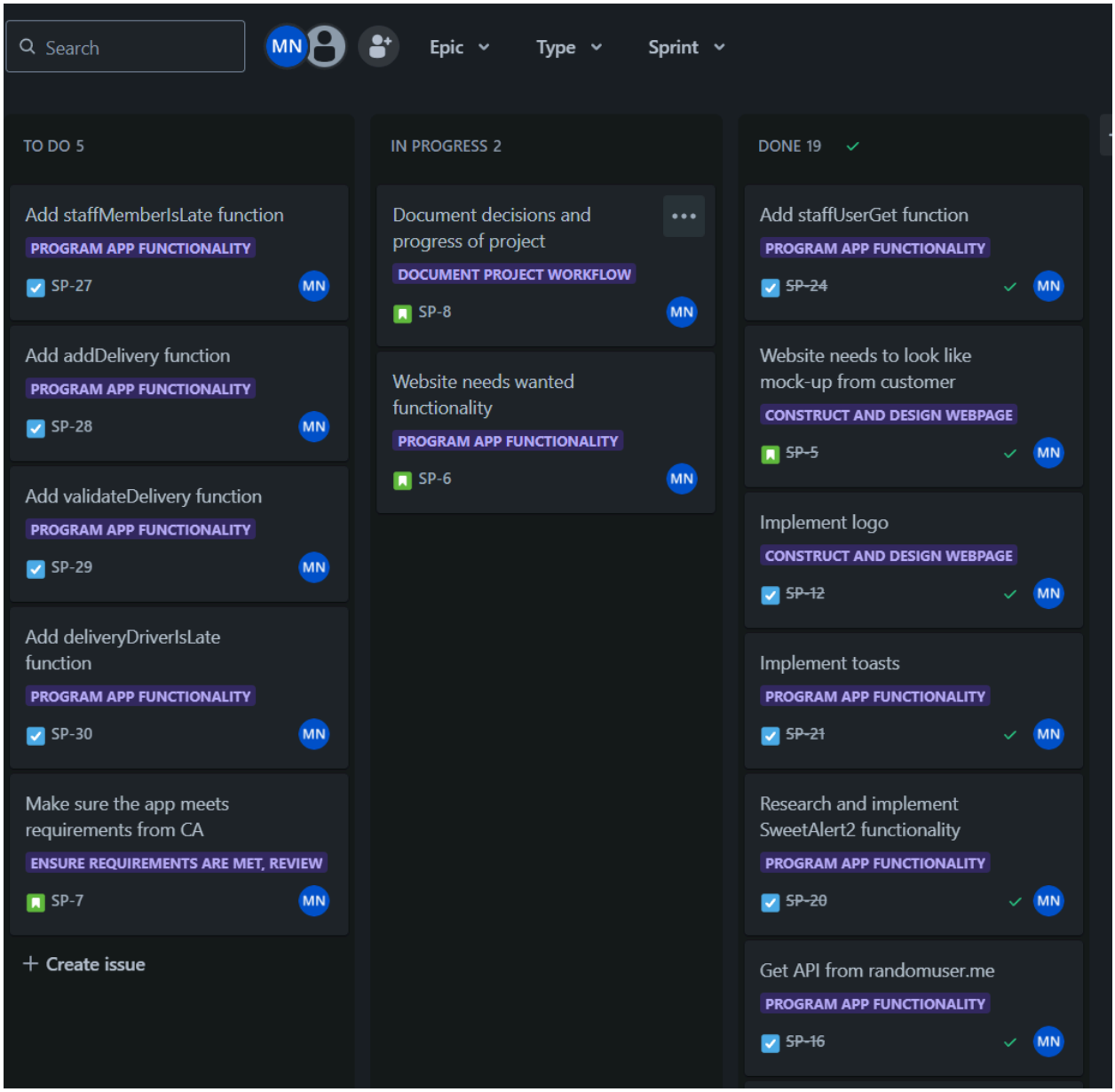
Positive

Delivery Board

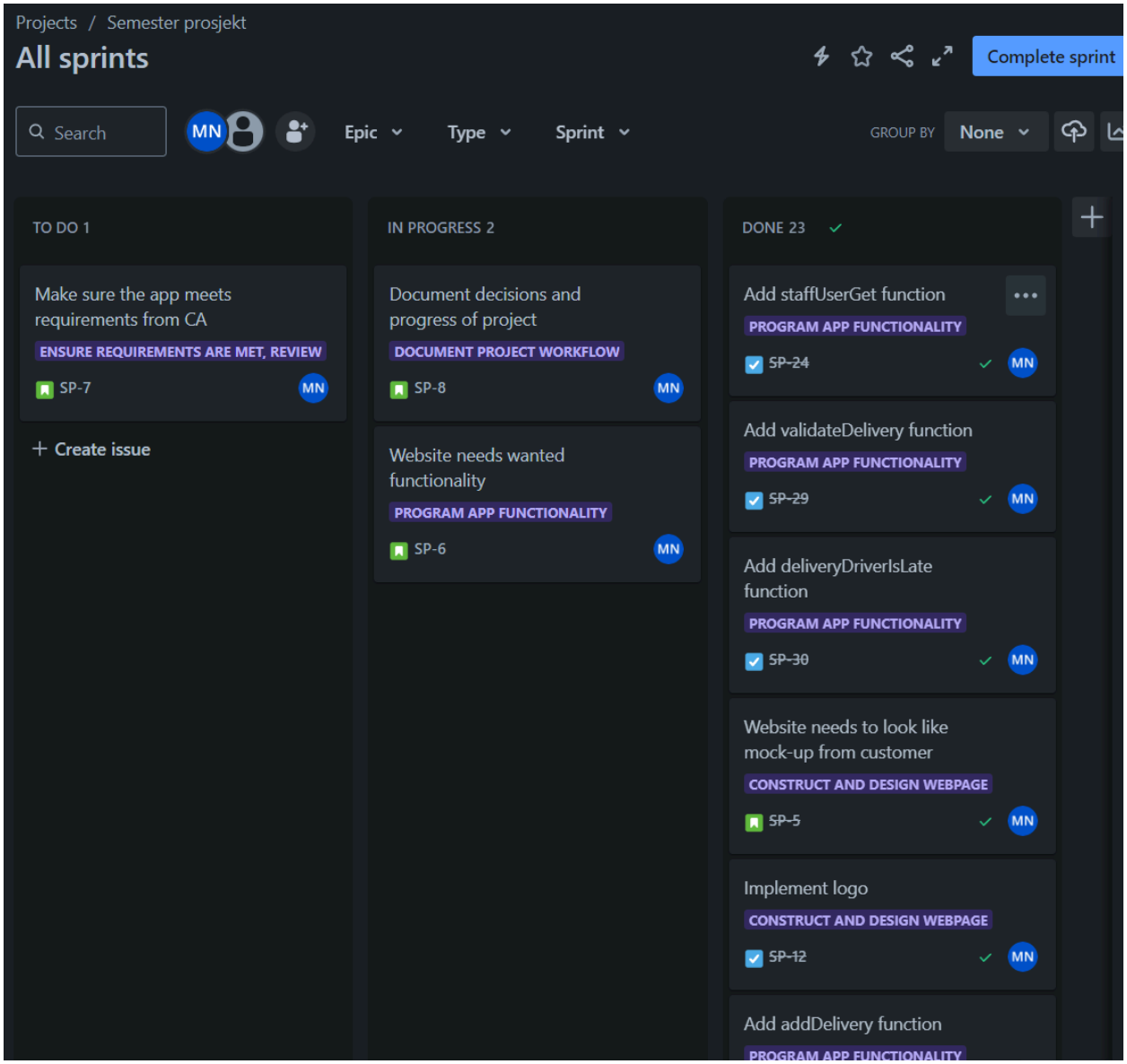
Day 4 at 15:24.
Added more tasks to my sprint, because the existing ones were too vague. Splitting issues down to smaller ones.



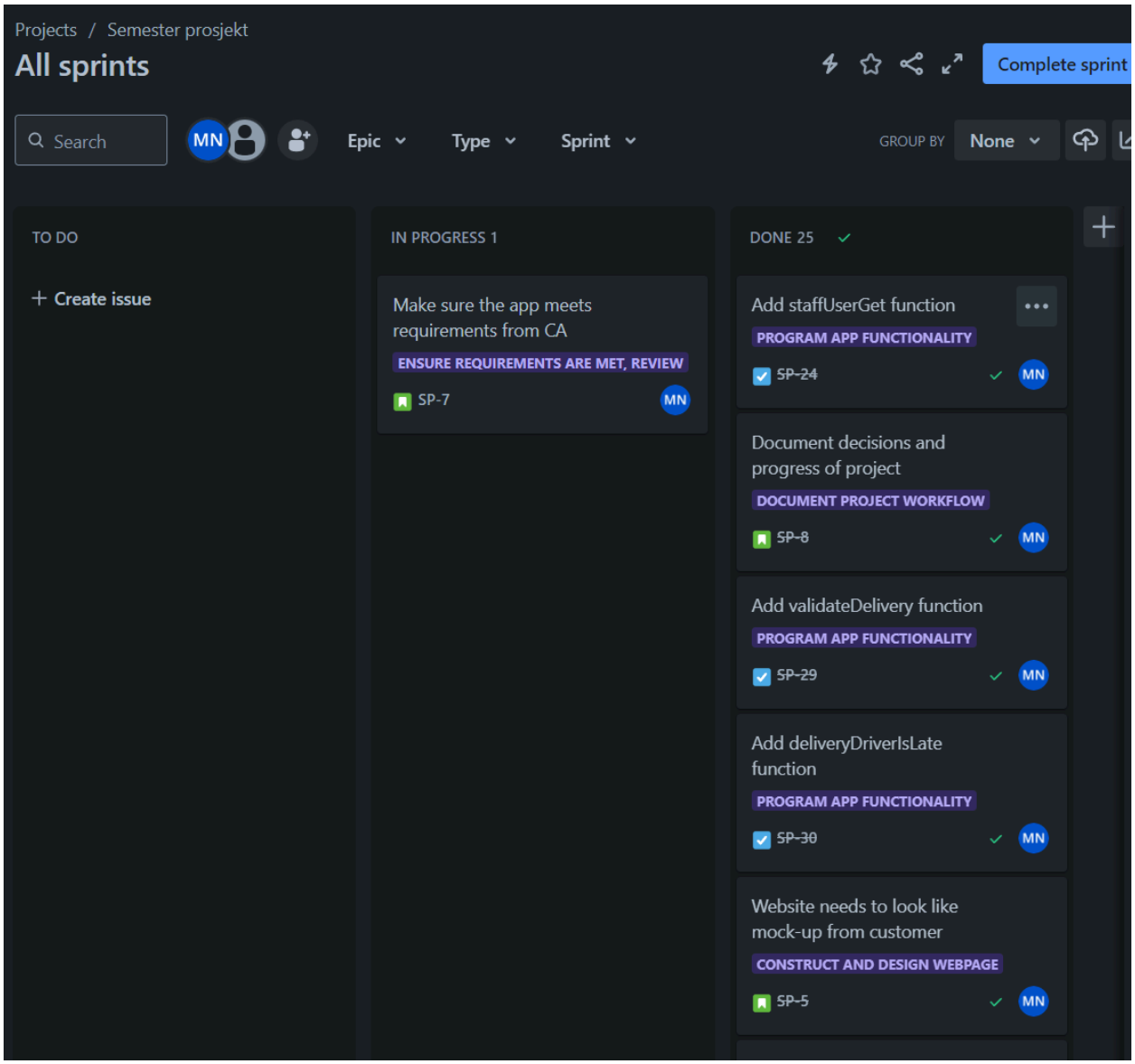
Week 2, day 3 at 14:53. 19 tasks done.



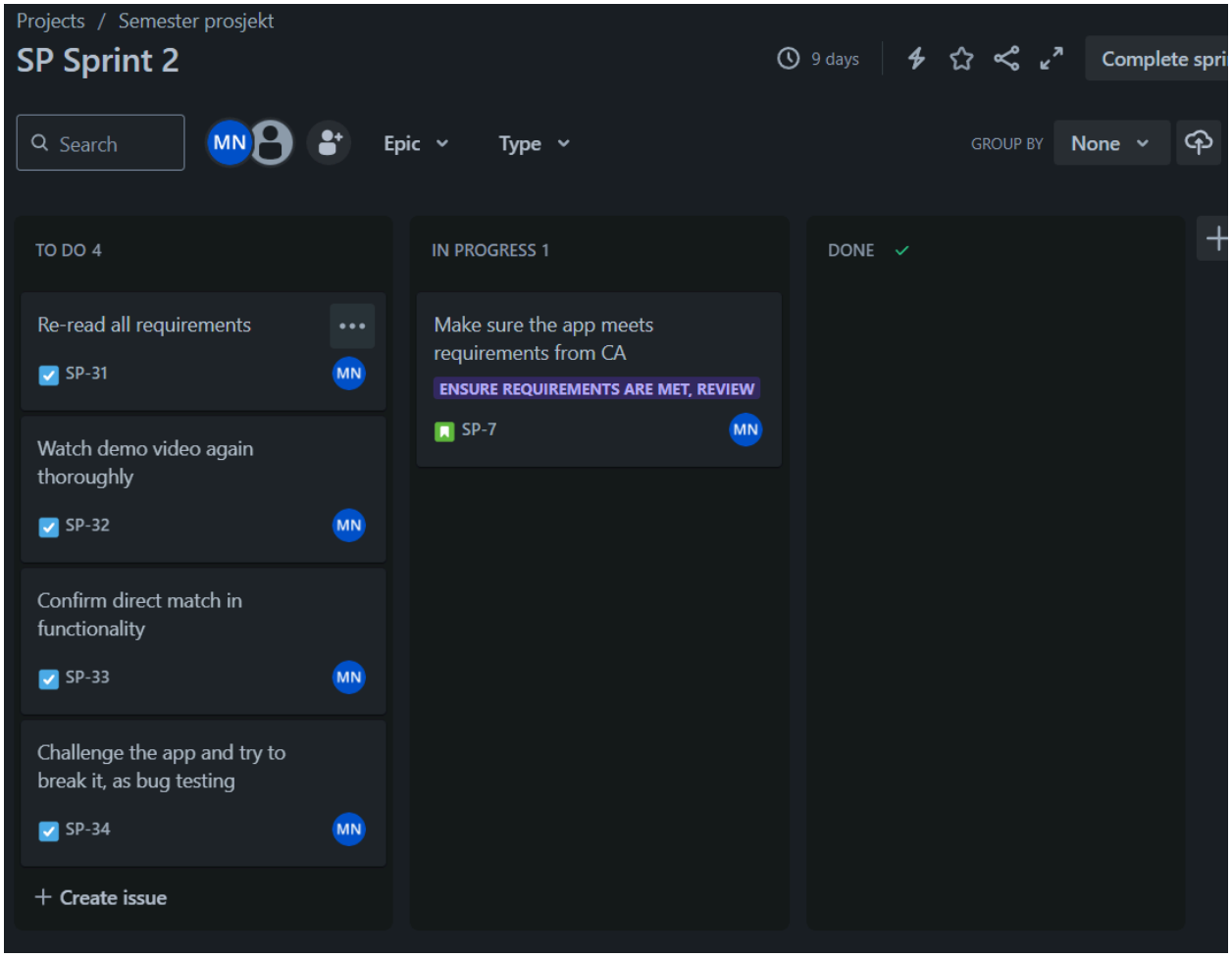
Week 2, day 6 (saturday), 12:55


Projects / Semester prosjekt
All sprints
Complete sprint
Search
MN
Epic
Type
Sprint
GROUP BY
None
TO DO 1
Make sure the app meets requirements from CA
ENSURE REQUIREMENTS ARE MET, REVIEW
SP-7
MN
Create issue
IN PROGRESS 2
Document decisions and progress of project
DOCUMENT PROJECT WORKFLOW
SP-8
MN
Website needs wanted functionality
PROGRAM APP FUNCTIONALITY
SP-6
MN
DONE 23
Add staffUserGet function
PROGRAM APP FUNCTIONALITY
SP-24
MN
Add validateDelivery function
PROGRAM APP FUNCTIONALITY
SP-29
MN
Add deliveryDriverIsLate function
PROGRAM APP FUNCTIONALITY
SP-30
MN
Website needs to look like mock-up from customer
CONSTRUCT AND DESIGN WEBPAGE
SP-5
MN
Implement logo
CONSTRUCT AND DESIGN WEBPAGE
SP-12
MN
Add addDelivery function
PROGRAM APP FUNCTIONALITY

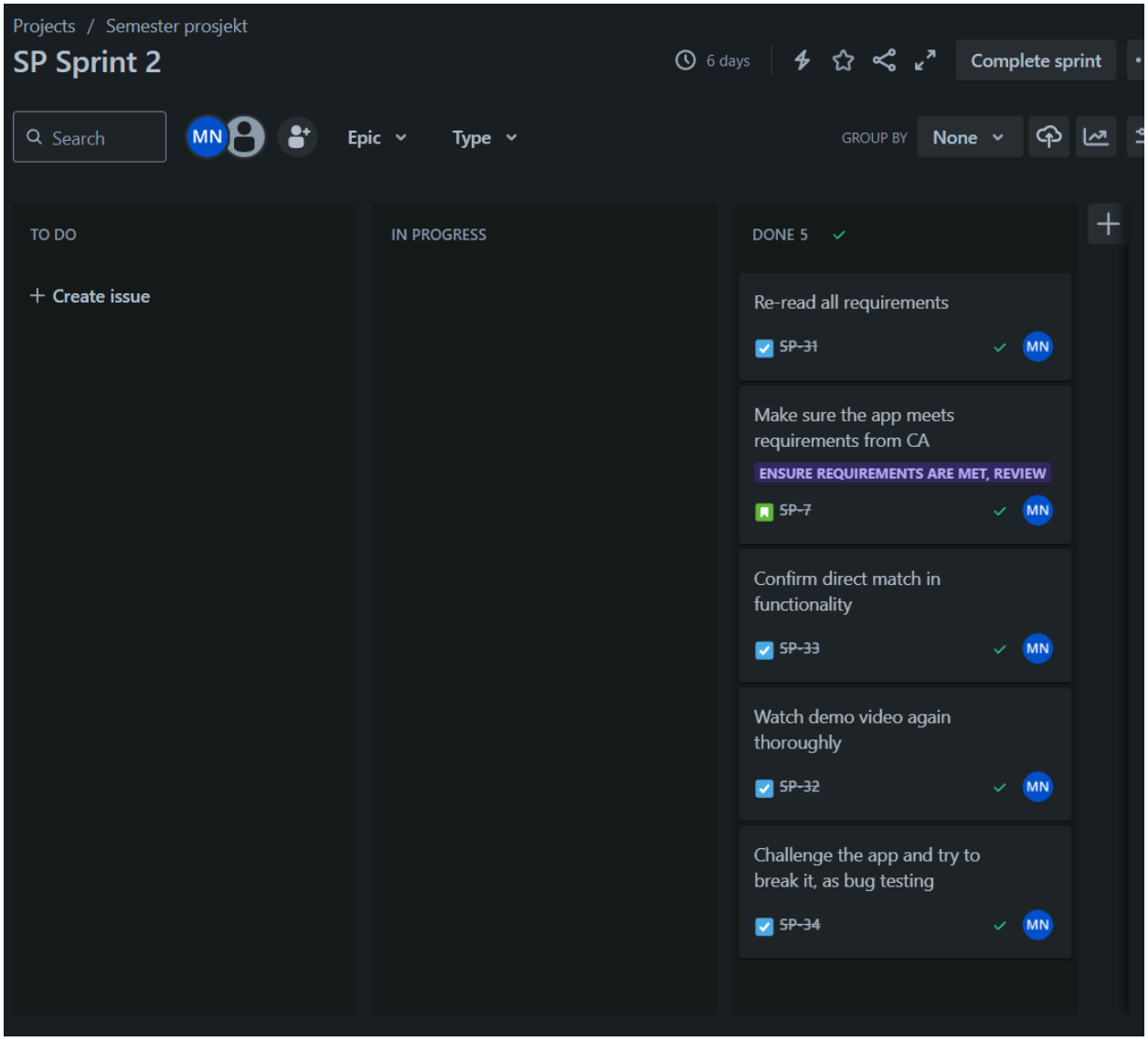
I had one task left in sprint 1, which was transferred over to sprint 2.



Here is sprint number two. I added tasks to this sprint, for me to solve in week 3 and week 4. I started working on this, and found some bugs and logic errors. Spent some time clearing these out.



My last jira screenshot. Week 3, day 3, 16:01.



Below are some notes I wrote down during the project. This will serve as my summary. I have also added some notes after completion.

- I started with creating the basic layout for the web page, with fake content

- I started using flexboxes. I can remember the www-council recommending flexbox to be the recommended standard.

- Nesting divs was tough when structuring the page, but I added a wide range of background colours to the divs to have control over the z-axis.

- I spent too much time on making the navbar. Had to make a decision between getting navbar from the bootstrap documentation or making my own buttons with hovering functionality. Was unsure how similar the navbar had to be to the mock-up. The hovering dropdown menu is inspired from W3-schools examples.

- Added some slight animation on the navbar to make the experience a little bit more pleasing, with inspiration from a Shorts video on youtube. The Dashboard button looks "active", but in reality only has a "bold" property. It's more a conceptual display. I could have made a real "navbar" on demand.

- Initially made tables with bootstrap, but returned to CSS-styling. Company requirements made CSS a better option.

- I originally hardcoded toasts into the HTML file, and used triggers to reveal them. Employee information was of course injected in the toast so correct information was displayed. This worked okay with the employees, because there were always max 5 employees.

- When I started working on the delivery table and corresponding toasts, I realised there could be anywhere between 0 to many (0..*) entries in the table.
I then needed to solve the toasts by making each "delivery driver" responsible for its own toast. Now the method .deliveryDriverIsLate() is responsible for injecting

the HTML file with a toast element. The toast element itself is responsible for deleting itself, both for hiding it and deleting the element as a whole in the HTML file. I could probably have skipped having both functionality for hiding and deleting the toast. Deleting would be sufficient.

- The staff member leave duration can be set to 0. This is handy to test toast functionality, since a toast can be triggered immediately (5 seconds max)

- There is set an interval for checking in staffMembers are late. This interval is set to 5 seconds. An equal interval is set for checking delivery drivers as well. Intervals can easily be set to for example once a minute or once a second.

- When a toast has been triggered, unique ID is stored in arrays (warningGiven and warningDeliveryGiven). The toast trigger checks if the unique ID already has received a toast, before firing. When an employee is IN or delivery is CLEARED, the entry is removed from the array, allowing for a new toast if the employee is out for another errand or the same driver has a new mission. The warningGiven array for employees saves their email, assuming this will be unique for each employee. The warningDeliveryGiven saved telephone+name+surname to ensure uniqueness, though telephone would probably have been sufficient.

- Schedule delivery. For selecting Car or MC, I first used radio buttons. It was easy to add icons to the radio buttons, but a bit more challenging when it came to a “select” menu, which doesn't seem to support it. It was harder to fetch input values from the radio buttons, so I transitioned from radio buttons to select. Reading the requirements, I realised the scheduling didn't require icons. Only the Delivery board required Icons, so that's why the schedule doesn't have icons for Car and MC.

- During development the API host randomuser.me has significant delay in their responses. Up to 15 seconds for a single response.
Their website was just slow, so I concluded that it was the API that had performance issues. I was worried it was my code which had gathered performance issues.

- Potential improvements: The API call is done one after one. Generation of the staff table would be smoother and faster
  if all necessary API calls were done immediately, and the table would be generated as the json object becomes available.
  Currently, generation happens one row at a time.

- When creating a delivery driver object, the system assumes uniqueness to the driver's phone number. Adding multiple drivers with the same phone number will/can cause issues to the app.

  This is because when a delivery driver object is created, a table row is created with an attribute with the drivers phone number. This attribute is used for targeting the row to remove, when the delivery driver is cleared.

  This could probably have been solved in a better way. Either by:
  - Assigning a different and more likely unique ID to the table row.
  - Using a different logic to target the row and remove it.
  - Hindering the user from inputting a phone number which already is “in use” by a row.

- I’m not proud of injecting raw html with javascript straight into the brower’s document, but it gets the job done and I’m satisfied with the result. I would use insertRow() and insertCell() if I wish to clean up my code.

- In README.md, I added “How to use web application” in case this was required, so I wouldn’t miss out on score points.

- I have left in the “commented-out” code. It serves as a portrayal of prior iterations and to display evolution and progress. The grader can inspect this if he/she likes.

Closing words:

- **This project has really taught me a lot. I didn't realise I would have the ability to create such a website. Although it's not really that fancy functionality, there is a lot of work behind it. I have felt success and mastery, but also frustration and helplessness. I have worked thoroughly and been committed from day one, and made solid progress each day. Being aware of the "Parkinson's law: Work expands so as to fill the time available for its completion" has kept me moving forward from day one.**
- **The code I have written feels like a mess, and it probably is a mess. But I'm also sure that it works and gets the job done, which aligns with the values of agile; delivering working software. This has been my focus; making it work, and then move on to the next task. If I dwell for hours on minor issues, progress halts.**
- **"Fail fast" is a value of scrum. My workstyle has reflected this. Implement it, test it, crash it, revise it, admire it.**
- **A lot of the implemented functionality could probably have been solved in a much better way. I probably solved some problems in an elegant and simple way, and other problems in a complicated and messy way. Nothing is perfect, and it's a beautiful property of coding that it almost always can be improved. "An idiot admires complexity. A genius admires simplicity".**

Thank you for your time and effort in grading this semester project.

Best regards,
Martinus Nordgård