

# Reflection Report

## Planning

I created a single epic for this project with the goal of creating the dashboard as described in the semester project assignment.

Projects / Kristian\_W\_Finstad\_SP

Issues

Share Export issues Go to advanced search LIST VIEW DETAIL VIEW ID ...

Search issues Q Project Type: Epic Status Assignee More + Reset Save filter BASIC JQL

Type	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created ↑	Updated	Due	
📌	KWFS-1	Dashboard	👤 Unassigned	👤 KP Kristian Finstad	=	TO DO	Unresolved	Nov 21, 2022	Nov 21, 2022	Dec 17, 2022	...

🗣 Give feedback 1-1 of 1 🔍 < 1 >

I created three stories that representing a fully functional feature, adding them as child issues to the epic.

Projects / Kristian\_W\_Finstad\_SP

Issues

Share Export issues Go to advanced search LIST VIEW DETAIL VIEW ID ...

Search issues Q Project Type: Story Status Assignee More + Reset Save filter BASIC JQL

Type	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created ↑	Updated	Due	
📌	KWFS-2	General layout	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		...
📌	KWFS-3	Staff Tracking	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 23, 2022		
📌	KWFS-4	Delivery Tracking	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 24, 2022		

🗣 Give feedback 1-3 of 3 🔍 < 1 >

After breaking up the features in three categories, i broke these down into smaller tasks representing the stages of each story. I did not wanted to micromanage the project by making each required function a Task. These tasks could be added as child issues in the Tasks created.

Projects / Kristian\_W\_Finstad\_SP

Issues

Share Export issues Go to advanced search LIST VIEW DETAIL VIEW ID ...

Search issues Q Project Type: Task Status Assignee More + Reset Save filter BASIC JQL

Type	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created ↑	Updated	Due	
☑	KWFS-5	Design Staff table	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		...
☑	KWFS-6	Design Delivery Table	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		
☑	KWFS-7	Design Schedule Delivery Form	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		
☑	KWFS-8	Import staff data from API	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 23, 2022		
☑	KWFS-9	Populate Staff table with Data from API	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 23, 2022		
☑	KWFS-10	Add functionality for staff tracking	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 23, 2022		
☑	KWFS-11	Add functionality to delivery form	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 24, 2022		
☑	KWFS-12	Populate delivery table	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 24, 2022		
☑	KWFS-13	Add Navbar	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		
☑	KWFS-14	Add Clock	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		
☑	KWFS-16	Add functionality to delivery board	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 24, 2022		
☑	KWFS-17	Create Dashboard layout	👤 Unassigned	👤 KP Kristian Finstad	=	DONE	Done	Nov 21, 2022	Nov 22, 2022		

🗣 Give feedback 1-12 of 12 🔍 < 1 >

Jira Software

Your work ▾
Projects ▾
Filters ▾
Dashboards ▾
People ▾
Apps ▾
Create

Q Search

Kristian\_W\_Finstad\_SP  
Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

Projects / Kristian\_W\_Finstad\_SP

Backlog

Q

KF

Epic ▾

Type ▾

Insights

▼ Design Sprint

22 Nov – 28 Nov (7 issues)

Complete sprint

...

Create Design of web application

KWFS-2 General layout DASHBOARD

TO DO ▾

KWFS-17 Create Dashboard layout

TO DO ▾

KWFS-13 Add Navbar

TO DO ▾

KWFS-5 Design Staff table

TO DO ▾

KWFS-6 Design Delivery Table

TO DO ▾

KWFS-7 Design Schedule Delivery Form

TO DO ▾

KWFS-14 Add Clock

TO DO ▾

+ Create issue

▼ Staff Tracker Sprint

28 Nov – 5 Dec (4 issues)

Start sprint

...

KWFS-3 Staff Tracking DASHBOARD

TO DO ▾

KWFS-8 Import staff data from API

TO DO ▾

KWFS-9 Populate Staff table with Data from API

TO DO ▾

KWFS-10 Add functionality for staff tracking

TO DO ▾

+ Create issue

▼ Delivery Tracking Spring

5 Dec – 12 Dec (4 issues)

Start sprint

...

KWFS-4 Delivery Tracking DASHBOARD

TO DO ▾

KWFS-11 Add functionality to delivery form

TO DO ▾

KWFS-12 Populate delivery table

TO DO ▾

KWFS-16 Add functionality to delivery board

TO DO ▾

+ Create issue

You're in a team-managed project

Learn more

	NOV							DEC							DEC							DEC							
	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Sprints	Design Sprint							Staff Tracker Sprint							Delivery Tracking Spring														
<a href="#">KWFS-1 Dashboard</a>																													

My Jira board only have the three default columns of To Do, In Progress and Done. In this project I want to mainly focus on the work and less on managing the project. Other columns could have been added, but would probably just be skipped for this specific project.

The screenshot shows a Jira Software interface for a project named "Kristian\_W\_Finstad\_SP". The main view is a "Design Sprint" board, which is a Kanban-style board with three columns: "TO DO 5 ISSUES", "IN PROGRESS 2 ISSUES", and "DONE".

**Left Sidebar:**

- PLANNING:** Roadmap, Backlog, Board (selected).
- DEVELOPMENT:** Code.
- Project settings:** Project pages, Add shortcut, Project settings.

**Top Bar:** Jira Software, Your work, Projects, Filters, Dashboards, People, Apps, Create, Search, 3 days remaining, Complete sprint, Insights.

**Design Sprint Board:**

- TO DO 5 ISSUES:**
  - Add Navbar (KWFS-13)
  - Design Staff table (KWFS-5) - **Selected**
  - Design Delivery Table (KWFS-6)
  - Design Schedule Delivery Form (KWFS-7)
  - Add Clock (KWFS-14)
- IN PROGRESS 2 ISSUES:**
  - General layout (DASHBOARD KWFS-2)
  - Create Dashboard layout (KWFS-17)
- DONE:** (Empty column)

**Bottom:** You're in a team-managed project. Learn more.

## Design

I started creating the general layout of the page following the provided mock-up and following the branding guidelines and adding the features not described by other stories. I did not add any extra to the design, though a background colour to the navigation bar would have been nice.

A zebra striped table would have been nice, but since no secondary colours was provided it would have broken the branding profile. On the same premiss I used white background on hover with a 0.1s transition for the hover effect and to show selected table row. Rounded border lines for tables using Bootstrap can be quite painful, so I left it out since it is of a minor aesthetic concern.

I used the table as container for the delivery driver form to ensure the design was consistent for all elements on the dashboard as shown in the mock-up. I also added a custom error style for the inputs unsure if the Bootstrap error handling would fit the branding profile.

## Programming

Since the application is supposed to be object oriented, I wanted to make a main Application class to hold values and methods regarding functionality. Using encapsulation would ensure only the run method would be publicly available. After consulting a teacher, I wrote all functions as standalone since it was implied that was expected of the functions mentioned in the grading criteria. But I did write the functions in regard to easy transition to methods inside the application class.

I tried to keep all code regarding the grading criteria within the scope of each function, extracting only what was required for consistency in separate functions.

I made the API call with an old fashion **XmlHttpRequest()** using a query string to limit the data only to what was required. I would prefer using **fetch()** in a later version for easier handling of more complex API calls.

I do not quite understand what is meant by *"Inheritance is used in object creation, using the data from the API call"*, but the StaffMember class does inherit the Employee class. After being encouraged in a previous assignment, only used a single object as input parameter for the class constructors. I would have preferred using a single parameter for each value defining the value type in the doc string due to the data transfer object not being defined locally. In addition using an object literal on instantiating the object feels a bit like boilerplate code. The data was stored in a private field inside the application class for later handling. This data is currently available outside the Application instance due to outside functions requiring access.

Since there is no id's included in the data models, I used the email address for staff member objects, and telephone number for delivery objects, as unique identifiers when searching in data arrays.

I created a simple unique id generator for creating ids for generated table rows and toasts with the purpose of easier document queries.

In the form, I used a try/catch/finally block to avoid page reload on submit or in the occurrence of an error. I also extracted the value from each field individually for easier handling of validation and validation error.

The validation of the time field format was a bit redundant due to the default format provided by the input field. But it's added as an extra safety, plus it is always fun to play around with RegExp.

I do not entirely understand what is meant by "*Inheritance is used in the Delivery Driver object creation*", but the DeliveryDriver class do inherit from the Employee class. This class also have a single object as input parameter on the same premiss as the StaffMember class. Object literal is also used when instantiating due to the nature of the input values.

I stored the SVG icons as text in wdt\_app.js for easier access and configuration.

I've added a bunch of comments and doc strings in the source code for the purpose of planning code and keeping track while working with the application. Especially having doc strings defining value types in parameters have been really helpful while developing in a dynamical typed language.