# **Temper Frontend Assessment**

### Attachments:

**Example screenshots** 

## Goals

- Robust panel based navigation in a modern JS framework
- Easy to use and extensible boilerplate (possibly open sourced in the feature) template code.

Temper has just finished the redesign of the new Client planning and onboarding tool. Using this tool Clients can manage their company's and shifts.

The design relies heavily on a hierarchical navigation pattern using stacked (side) panels. The panels should be able to provide easy navigation to end users.

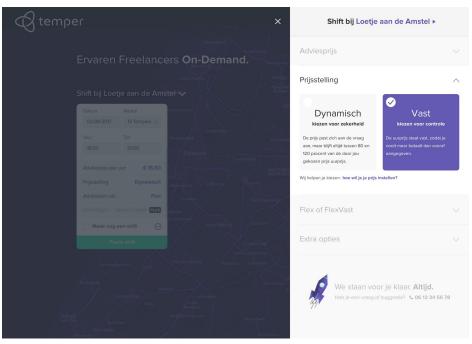
### Use case

As a client I want to be able to make detailed settings when adding a shift. Some of the settings require a multi level side panel. For example: I want to link the user that will be the floor manager during the shift. First I press "link user" and the side panel shows me a list of users. The user I'm looking for is not there yet, so I press "add user". The side panel now opens up a second layer with a form.

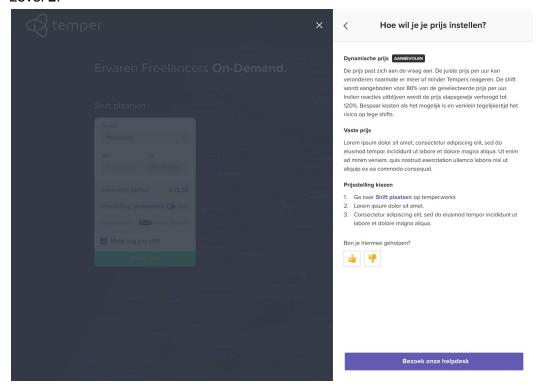
You could compare this with the settings menu of iOS. The first level is the root level, when you click 'Privacy', you're navigating to the second level. When you press 'contacts' you get to the third level. At the top you'll see a button taking you back to the previous level.

# Example of a side panel

### Level 1:



## Level 2:



# Requirements

### Must

- ES6 syntax
- Clean CSS3
- Clean and easy to read code and project structure
- Navigation inside the root panel, implementation should look something like
  <u>UIViewController</u>. A component that is used in native iOS development a lot. Panels
  without internal forms, buttons, collapsibles etc. Some example text + buttons are
  sufficient. The main goal is setting up the core framework of the panels.
- Navigation using a back / close button. Panels should be closeable using a back / close button. Whichever is appropriate in the current context.
- No eye candy, only functionality

### **Should**

 Rethink your initial code design and check if all components are of an acceptable size (no. of lines). If it can be extracted, extract it into a separate component.

### Could

- Popover style panels. A panel should be able to be presented on top of another panel, with the same possible internal navigation.
- Implement responsive design where panels are 100% wide on mobile.
- Vue-router usage
- Implement design

### Won't;

- Tree style navigation (split view) panel
- Pixel perfect design rebuild

# Hints

- Vue-router (router-view, hash based, client side routing)
- https://vuejs.org/v2/guide/
- Single file components

You can find more examples of the new design in the zip file.

https://goo.gl/ftoQdg