Code Assignment for Symfony with Vue.JS Position

Estimated Time: 1-2d

For this assignment you are required to build and demonstrate a custom list view component based on Vue.JS and integrate it with a demo Symfony2 web application.

Overview

Deposit Station (`DS`) is a frontend component that acts as a right hand menu and provides list of patients / groups and ability to organize data based on various criteria. Data is retrieved via a JSON (see file `data.js`) from backend Symfony controller. It consists of 2 main logical elements:

1. **Data Pane** - contains a number of [filtered] data items representing patients or groups, depending on the navigation option that is currently selected. It also has one navigational element “More” which allows for more data to be loaded in the list. In addition, each patient data item has a details arrow, which when clicked will display information from the `details` field of the JSON.

2. **Navigation Bar** - modifies the data displayed in 1. by ordering, grouping or filtering it. It contains 3 main navigational components and 1 filtering functionality that works in conjunction with the 3 navigational components:
    a. **AZ** - will modify the Data Pane to contain an alphabetical list of active patients. Active patients are patiens who have `{status: 1}`.
    b. **Group** - Data Pane will display all group names that contain active patients. Clicking on a specified group will list patients in this group with current group name. There should also be a "go back" arrow link to the list of all group names.
    c. **Archive** - Data Pane will display all archived patients alphabetically ordered. Inactive patients are patients who have `{status: 0}`.
    d. **Filter** - when the currently active navigational component is clicked (e.g. AZ is active and user single clicks on it) a search bar should within `DS` appear where search terms can be entered. Items in the JSON who have a `filterBox` field should be filtered using its `value` field. You can assume `type` will always be `'text'`. Search bar should have `x` button for closing it.

Annex A displays a visual representation of the functionality described above. Choice of actual design is up to you, the important part is that you end up with a web page which is integrated in a Symfony2 web application that has a Vue.JS powered Deposit Station component. This component should be able to handle large data gracefully and should behave smoothly. There's a bonus section listing additional features.

Implementation

Included in the archive which you have received are:

- This document
- `data.js` file describing JSON structure. This needs to be returned by a Symfony2 controller (see **Implementation**)
- An archive containing a Vagrant setup with a demo Symfony installed. This is so that you have a quick dev env at hand at `192.168.33.99`. VM runs Symfony2, PostgreSQL. If there's anything else that we need to install while testing, please make sure you include it in the document describing deployment instructions.

The following items need to be implemented:

1. Create a Symfony2 Controller & Action that will deliver json (see `data.js` file for structure). It's safe to use the hard coded JSON which we have provided for testing purposes, however submission will be assessed based on performance to handle big data (20-25K items) **- 10p**

2. Attach vue.js 2.0 and all required libraries to application - **5p**

3. Create Deposit Station Vue.JS component as described in **Overview** section **- 70p**

4. Last picked list (and/or specific group) should be remembered on page reload (local storage preferred, but cookie is OK) **- 10p**

5. Implement pagination for items as "more" button (append next page instead of replacing it) - **5p**

Criteria for Assessment

Your submission will be assessed on the following criteria:

- sound UI which provides a smooth experience for the user
- code analysis and design decisions

- additional points will be gained if any of the bonus features are implemented, provided that core functionality is there.

Deliverables

A `.7z` archive containing:

- web app source code
- instructions for deployment (if anything else is needed other than extracting files to specified folder)
- a brief document that describes implementation (doesn't have to be release level doc)

Bonus Features

These will serve as differentiating factor between two comparable implementations. However they will not give a poor implementation of "core" features an edge over a sound implementation that has no bonus features. In other words they are optional.

- Make DS work as a right hand menu to a Symfony patient page. When a patient is selected in DS, the page for that patient must be loaded and use FOSJsRoutingBundle for route generation **- 30p**
- Use smooth transition effects for switching lists & filters - **10p**
- Suggest implementation & performance optimization for lists with > 20k items **- 10p**
- Implement events that allow external components to switch current navigation list **- 10p**
- There will be many DS in application - keep this in mind when you save last picked navigation in local storage or cookie **- 10p**

Some General Observations

Please note that there are several candidates involved in the interviewing process. Your application will be evaluated against theirs and one or two selections will be made based on actual submissions. Make sure that this assignment is something that you are willing to invest time given stated conditions. Also, it is okay to send a solution that is incomplete, we will consider it, but make sure you describe what is missing. Please feel free to ask any questions regarding specs and best of luck! :)

ANNEX A - Deposit Station UI Example