Front-End Code Challenge Specs

Viventium Software

Objective: Implement a simple issue posting system (Comments / Issues / Tasks / etc.) that supports inline HTML and Tags

1. Display a list of elements on the screen based on the following API definition:

```
{
        id: “1”,
        title: “This is an item”,
        text: “This is a description of the item, it might describe a bug/task/comment, it
can also display <a href=”www.google.com”>Links</a>”,
        tags: [“bug”, “issue”, “etc”]
}
```

2. Each element on the screen should support the following features:

   2.1. An element should be read only by default (with no input elements)

   2.2. In read only mode, **internal element text should support HTML tags**

   2.3. Each comment element should have the following buttons:

      2.3.1. Edit - Clicking on an Edit button should transform the element into an editable widget that allows editing all the properties of the element.

         2.3.1.1. When in edit mode, there should be a way to cancel the changes or save them.

         2.3.1.2. **Allow adding existing tags, based on tags in other elements or new ones if the user types a tag that doesn’t exist.**

         2.3.1.3. Comment text should support simple html tags.

      2.3.2. Delete – Deletes a comment element

3. **On the top of the page, there should be a way to filter elements based on tags**. The user should only be able to select tags that are available in any of the elements.
4. At the end of the comment list, there should always be an editable comment element for adding new comments.

General Guidelines

1. Use Angular version 8 or higher
2. Define at least **1 module, 2 components** and a **Data service**
3. Use LESS/SASS (Bootstrap styling is fine, try to add some custom modifications)
4. Since there’s no API defined, use mock data in the form of a JSON file, but utilize **Angular Http Client** for that.

4.1. Only one place should be aware of the Mock usage. The rest of the code should work with it as if it's coming from an async API.

4.2. Commit the result to Github/BitBucket

5. No need for RTF support, simple html tags are enough. The screenshots I provided are just for general idea. There is no requirement to use that specific UI design.

6. **Bonus (optional) – Auto Calculate basic math expressions:**

6.1. In View mode, detect basic math expressions and display the calculated result instead of the equation.

6.2. The system should only support basic expressions with + and – operators. Parenthesis are not supported.

6.3. The calculation logic should be manually written, no library should be used for the calculation logic (e.g. Eval)

6.4. Basically, you should implement a "Calculate" method that accepts a string (valid one) and returns a number (the result). The method should parse the string and perform the calculation.

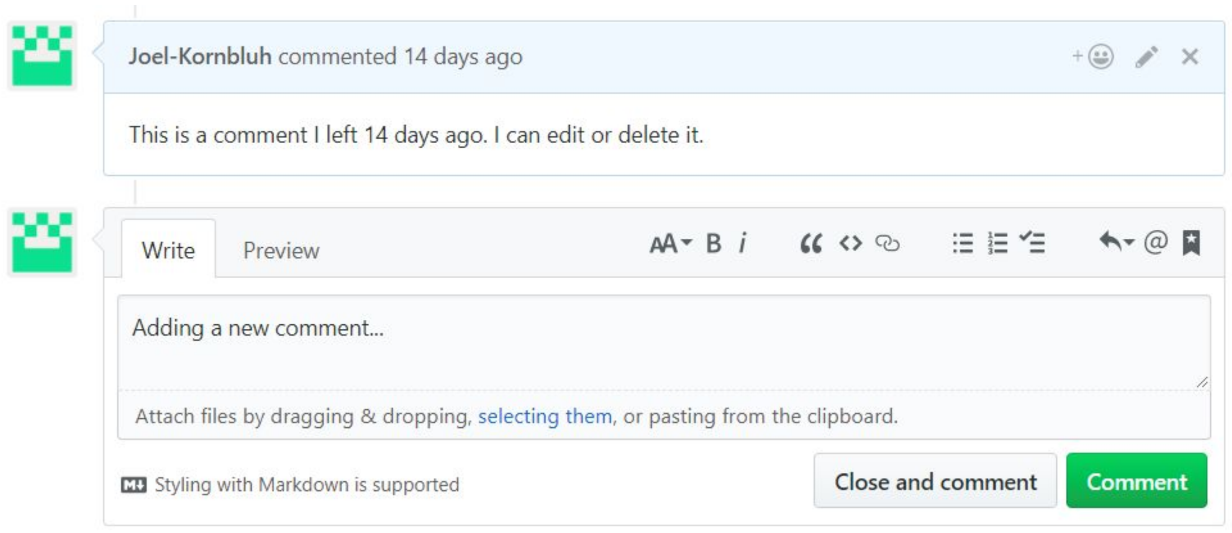
6.5. Example:

6.5.1. If a user edits a comment and writes the following in the comment: "4+5-6", when saving the comment, in View mode, the text should be 3.

6.5.2. You can be creative if you want in regards to how you display that the value is calculated by an expression.

We don't want the full issue tracking system in Github, I gave it as an example for the commenting part. The task is to display a list of entries based on the API and allow editing them.

You can refer to Github Issues (commenting system) as a general example:



Joel-Kornbluh commented 14 days ago

This is a comment I left 14 days ago. I can edit or delete it.

Write
Preview
Adding a new comment...
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Close and comment
Comment