Advice

Project Approach tool

Oleksii Soloviov - 495673@student.saxion.nl

Table of Contents

Introduction

To start, this project for our team was meant to be an implementation of a side bar card dragging feature, auto testing of the system, additional security features and fixes on the VPS side. That said, our team encountered a lot of complications before we even could start implementing the mentioned features. It can be clearly seen in the frontend implementations. There were a lot of application features that from the first sight work, however if you actually start using the application, they do not work.

Object Declaration

As an example: when you enter the main menu after being logged in, the system creates a workspace example for you, so then you are introduced to the application in the onboarding feature. However, if you after delete the example workspace, you will always get an error message in the browser console about ".data() is undefined". This happens not only in this case. First of all, this error means that the constructor of the mentioned object is empty, while being sent to the function. There is the same situation when the user tries to delete the "question" card on their workspace without writing anything in it. The same error appears. There are more examples, however it is not important right now. What is important is that other objects which are created and interreacted with, do not show the same error because either the constructor of them is not empty or this object when is used in a function does not use their constructor. That said, the frontend becomes a mess where even though all the object models have the same looking constructor, but some parts of the system use them or not. As an advice, there should be a consistent same type of way that all the objects in the system are created and used in functions all over the application. The bad example of everything mentioned before is that when the card is created and you want to assign an id to it, it cannot be done unless this card is first assigned to the constructor of an empty card you create below. This looks completely unnecessary and confusing.

Testing

This application has a lot of different interactable features on the workspace you create. While they were implemented nothing went wrong and the features were probably all working as intended, however while our team started working on the application, we had to implement the auto testing feature, so the functionality status of the application may be tested at any time completely covering all Important features. However, we were not able to even begin implementing the frontend tests because the features we were testing did not even work. Returning to the "question" card on the workspace. This card can be accessed through the side bar by clicking on it. That creates a card on the workspace itself. This card has a text input to write a question. However, it turns out that the card cannot be removed from the workspace unless it has a question written in it. Usually, any object on the workspace can be removed entirely. Another example is the saving of the start and end date of the cards. There are 2 places where this can be done. Planning and Creatin page on the Workspace. They were clearly done by different people because the date saving between them conflicts with each other. As an example, if the date is saved on the Creation page, it is still the same on the Planning page. However, if the user then edits only the starting date of the card and returns to the previous page, the ending date becomes empty. This happens because of 2 different date saving functions the application has for these 2 pages.

As an advice for this case the same functionality should have the same looks and features. The calendars could be made the same. That said, the most important advice is that all the features implemented by a party should be tested before handing the project over to another party in the future to avoid an unnecessary waste of time.

Undefined Objects

There are many checks in the application regarding undefined objects. However, many of them are unnecessary like in case of: date saving on the Creation page. The date cannot be saved in the menu of the card, unless the card has a note written in it. There is no reason for checking if the note is undefined because the date can be saved anyway. Moreover, if the card was created and the user did not assign a date to it and then visited the Planning page, the card would have the date being set to the second and third of February. Why do we check if the object is undefined if we can just assign today for the date, so we do not make any checks? This is applicable in the most cases. As an advice, in the future implementation there should be considered the case of assigning a value when the object is created instead of checking if it undefined once without else statements.

Dependencies

Through the history of this application, there were 4 students who were working on it. Some of them were using specific tools to achieve their goals, however some of the tools were not used in the end and the project was left with a ton of unused dependencies which caused a big mess in understanding of what the application actually consists of. As an advice, during the development of the project, there should be used any dependencies developers need, however by the end or during the project there could be a check on what tools are not used / redundant in the system so they can be removed for the future developers. This will dramatically improve the quantity of work others can do by knowing the really used tools in the future.

Multiple People = Multiple solutions

There is a noticeable difference between certain functionality and looks of objects in the application caused by the number of different people working on it in the past. This causes a problem to any future developer because the consistency of the same tools used is not applicable to this case. As an example, while implementing tests, there was a test including start and end date of the card. When a user tries to change them, the calendar window pops up. This window looks a specific way and, in every week, raw only has the selected month numbers. However, when the user leaves the Creation page and enters the Planning one. They can also change the date of cards there. However, the calendar window for the cards there is completely different in both the looks and functionality. How the hours of the cards can be changed and the week raw can actually have days ending of the previous month and starting a new one (30, 31, 1, 2, 3, 4, 5) not as (empty, empty, 1, 2, 3, 4, 5). This causes weird complication when implementing tests for the system because even though the application is meant to have a certain functionality, it is different depending on the Page you are at. As and advice before any feature implementation is started, there should be had a look at how the similar functionality was handled as with the calendars.

Workspace Grid

For some reason the workspace grid is not free for any phase movement around the place. The grid has specific locations where the phase is put in if the user moves it around. That does not allow free movements and makes the user feel restricted by the phase movement functionality. In addition, there is an arrow functionality on the workspace which was turned off because it had graphical issues with phases. The issue is that when the user first creates 2 phases. Then, connects one arrow from the first phase to the second one. At that moment the arrows look alright and there is nothing bad. However, if the user moves any of the phases because the workspace grid is not free, where the user leaves the cursor, the arrow will stay. But then, a moment later a phase will move a little to the side because of the specific grid collation. That said, the arrow will stay either looking through the phase or not touching it anymore. This can be fixed by reloading the page. As a suggestion, there could be completely removed the current workspace grid and made free for any dragging. That way the problems with arrows will disappear and the user experience and smoothness will improv.

Logic & VPS

After testing deployment on the VPS, it was discovered that account login logic was previously created on the frontend. Which means that the project cannot be built for production and hosted statically. Therefore, the application is currently being run using PM2 and accessed via a reverse proxy with nginx, whereas nginx should just be serving the static build files. As for any future development, it is crucial to move this logic to the backend, so that the application can be hosted correctly for production.

Requirements not completed

- User registration with roles
- Undo functionality in the dashboard
- Displaying different views based on the user's role
- Validation of all inputs
- Keyboard shortcuts for dashboard functionality
- Adding connection arrows to show the order of phases on the workspace
- Adding functionality tips to the dashboard

Dependencies (Frontend)

- There are several unused dependencies which can be removed. Installing depcheck and running it will allow you to find them.
- Updating all the dependencies at once leaves many Angular errors which can be fixed by changing the file paths of the broken angular imports in the affected files. If you get any errors regarding the next() function. Either typecast it with void or pass a fake value.

- When migrating all dependencies to their latest version, there are 4 main dependencies which cause the 'npm i' command to fail due to conflicting dependencies. It is advised to use the npm i --legacy-peer-deps command if you want to try running the frontend.
    - angular-resize-event
        - Seems to be part of the cause of Angular dependency conflict issues.
    - tslib
        - Creates an enormous number of conflicts when trying to npm i.
    - ngx-autosize-input
        - Seems to be a part of the cause of Angular dependency conflict issues.
    - devextreme-angular
        - Seems to be a part of the cause of Angular dependency conflict issues.
- Individually updating dependencies might be possible but has yielded little results due to the package.json inherently having issues.
- Successfully updating all dependencies to their latest version and fixing the errors will lead to a compiled frontend however the localhost page will be blank due to a runtime error. I was unable to locate the cause of this issue.