THESIS MONTHLY PROGRESS REPORT

May 7, 2020

Aristofanis Chionis Koufakos
CERN

Contents

0.1 Progress During April 2
0.2 Milestones for current month 5

0.1 Progress During April

The source code of the Slides App can be found in these two mirrors:

- In Github
- In Gitlab.

The application can be accessed from this link: https://slides.web.cern.ch/
Most important points:

- A reliable Text editor was created. I used this project as a base and extended it by adding custom functionality, for saving the text.
- A reliable library to handle resize and move functionality of React components was found and integrated in the project. Reactablejs. I had to calculate the coordinates with a mathematical type. So when the dragging of an Item stops, I have to save the new (x, y) position to the Redux Store. To calculate this from the emitted event, I do:
  - const x = e.client.x - e.clientX0 + coordinate.x;
  - const y = e.client.y - e.clientY0 + coordinate.y;

  1. const x and const y, are the variables to be stored.
  2. e.client.x and e.client.y are the final coordinates of the item as recorded by the library.
  3. e.clientX0 and e.clientY0 are the coordinates from where the item started its dragging.
  4. coordinate.x and coordinate.y are the most recent values of x,y as stored and used in the Item's state.

  So the combination of the above coordinates, give us the final position of the element which will be stored in the redux store. As a result, Images and Text boxes can be dragged and resized.
- Multiple Images and Text boxes can be put in any given slide.
- Capability of renaming the presentation by the user, while saving, after this the user can continue editing his presentation with the new title without the need of re-importing it.

- Integration with Phoenix paused for now, because there is still pending work from the Phoenix side until they have it production ready.
- Focus on Slides from Browser approach.
- Storage of pictures in the production is now fully understood, I know what was wrong and I am working in the right direction to fix it. The problem was that I was deploying only the frontend build in Openshift and that way I didn't have access to the server to store the images. The server's code was not getting deployed.
- Separate backend and frontend of the application, now both work independently from one another, this helps in maintenance, good coding practices, building and deploying in production. Now there are two projects in Openshift (the production cluster used for deploying dockerized apps at CERN), slides and slides-backend.
- Added Authentication in the Frontend, using the new CERN SSO and added the keycloak instance in my redux store, from there I am able to retrieve username and the authentication token which will be used when sending requests to the back-end server, to upload an image or any other action which needs backend-frontend interaction.

Notes on Project Documentation:

The best way to write documentation for JS Projects seems to be the following:

- JSDoc, is a standardized way of writing comments in your code to describe functions, classes, methods, and variables in your codebase.

```
/**
* Adds two numbers together
*
* @param {number} first The First Number
* @param {number} second The Second Number
* @returns {number}
*/
function add(first, second) {
  return first + second;
}
```

Listing 1: Example of Documenting a simple function

- use Documentation.js to generate documentation based on all the comments you wrote in your project. What this does is to translate your carefully written JSDoc code comments into readable html or markdown documentation.
- *It is probably a good idea to output documentation directly in .md, in order to use it with Mkdocs?!*
- Use Flow, this seems to be a nice to have, but optional step, to make documentation easier to write. Flow helps you avoid writing the types of a function, because it extracts this information automatically. It seems to have a nice ReactJS integration as well, provided that our frontend is fully written in React, this can be really useful. Definetely worth looking more into it.

source

Knowledge on storage solutions:

1. EOS is a powerful storage system and is used by applications when storage needs to be shared with other applications or accessible by users.
2. In our case, we need to store the images temporarily on the server, and the server should render them statically. In this way, the frontend, will be able to access them.
3. When the user hits the 'Save' button, the server packs all of the images that live in the user's presentation folder, also writes a presentation.JSON file and creates a compressed file with the name (title.slides), then sends that to the client side, so that the users can save his presentation locally. When the integration with Phoenix continues, the server instead of sending this title.slides file to the client-side, he will send it to the Phoenix backend storage.
4. So in our case we need to mount a CephFS storage volume to our slides-backend server.

0.2 Milestones for current month

- Add Authentication in slides-backend, using an authentication adaptor for nodejs, from keycloak.
- Be able to store images in the production environment.
- Fix some more minor issues and open the Slides App to the first test users!
- Change the presentation theme on-the-fly when someone is creating a presentation.
- Show to the users how each theme looks like before they choose it.
- Start the Slideshow functionality.
- Start the export as PDF functionality.