The following task should take 3-4 hours to complete. If you spend 6 hours or more on this task, please send us whatever results you have, even if the application doesn't work.

Some problems to be solved in this task are close to the problems that we're dealing with in our commercial products. The system requirements are the same as for our products.

When assessing the results of your work, we will be paying most attention to the architecture and design of the application, performance, code quality, accuracy of static typing, and usability.

(A note about comments: you don't need to comment everything! Leave only the comments that provide considerable help in reading the code. Comments must be in English.)

Along with the resulting source code, please submit a short description of the architecture (2-3 paragraphs), in English.

Sticky Notes

Your task is to implement a single-page web application for sticky notes. You are required to implement at least 3 of the following 4 features:

1. Create a new note of the specified size at the specified position.
2. Change note size by dragging.
3. Move a note by dragging.
4. Remove a note by dragging it over a predefined "trash" zone.

You are encouraged to think about the best UI for these features that is possible to implement in the specified timeframe.

System requirements:

1. The web application is intended to be used on desktop. Minimum screen resolution: 1024x768.
2. The following browsers should be supported: latest versions of Google Chrome (Windows and Mac), Mozilla Firefox (all platforms), Microsoft Edge.

Technologies:

1. Language: Typescript.
2. You should use React without stock components.
   The general idea is to avoid using readymade solutions, so that we can fully assess how you design and engineer solutions in the scope of a small task like this.
3. If your project requires building, provide the necessary instructions.

As a bonus, you can implement a few optional features:

I. Entering/editing note text.
II. Moving notes to front (in case of overlapping notes).

III. Saving notes to local storage (restoring them on page load).
IV. Different note colors.
V. Saving notes to REST API. Note: you're not required to implement the API, you can mock it, but the mocks should be asynchronous.