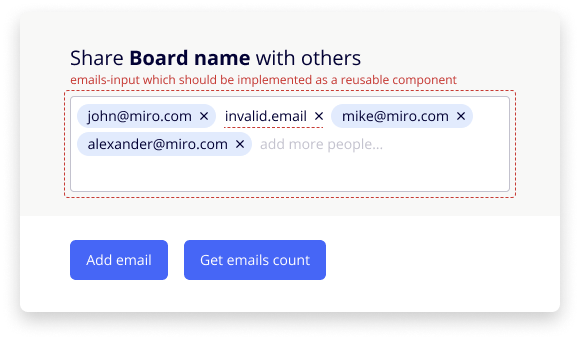
# Frontend test assessment

  
Please create a form based on the screenshot above.

Input field with emails (let’s call it **emails-input**) should be implemented as a separate js script/component, so it can be used in any other form independently.

Two blue buttons of this form should be implemented outside **emails-input** and interact with it via its public API.

An example of **emails-input.js** usage, excluding the form layout and buttons:

<!-- form header -->

<div id="emails-input"></div>

<!-- form footer, buttons -->

<script src="emails-input.js"></script>

<script>

var inputContainerNode = document.querySelector('#emails-input');

var emailsInput = EmailsInput(inputContainerNode, {...options});

// Handling **Add email** and **Get emails count** buttons, etc.

</script>

## Assessment requirements:

* A link to a demo page with implemented form should be provided. You may host it on [GitHub Pages](https://pages.github.com/).
* A link to the public git repository with source code and built version should be provided as well.
* Git repository should contain a README that describes how to use that component. Examples + documentation.
* Demo page should work in IE 11 and the latest versions of Chrome, Safari, Firefox, Edge.

## Functional requirements:

* Email block should be created by pressing Enter, entering comma, or by losing focus on the input field. A Block can be deleted.
* Input width must depend on the parent container’s width and height. If parent width changes, emails should be redistributed by rows.
* Other than that input should neither depend on the form or page styles, nor conflict with them.
* If input has too many emails, user should be able to scroll it.
* Pasted emails should be converted into blocks immediately. If multiple comma-separated emails are pasted (e.g., “[ivan@mail.ru](mailto:ivan@mail.ru), [max@mail.ru](mailto:max@mail.ru)”), they should be converted into multiple blocks.
* "**Add email**" button adds a random email to the list.
* "**Get emails count**" button shows an alert with valid emails count.
* Do NOT implement editing of added emails.
* It should be possible to create several emails editors on the same page.
* **emails-input** should have no external dependencies like React, Lodash or any polyfills. Usage of TypeScript, Less, Webpack, etc. is for your consideration.
* Tests are not mandatory, but having them is a plus.
* Performance isn’t a big concern, but there should be no major flaws, such as memory leaks or re-rendering all email blocks every time you add or remove a single email.

## API:

An **emails-input** instance should implement the following API:

* A method to get all entered emails. Both valid and invalid.
* A method to replace all entered emails with new ones.
* Ability to subscribe for emails list changes.

## Design:

<https://www.figma.com/file/CWdAs3rN4d2gZpnoN7ZPvj/Share-test?node-id=0%3A1>

* Log into Figma to see details: font, sizes, colors, offsets, etc.
* Open Sans font can be found [at Google Fonts](https://fonts.google.com/specimen/Open+Sans).