

LogDNA FrontEnd Engineer Challenge

- Timeframe: we expect it to take up to 2 hours but feel free to take as much time as you need
- Languages: Javascript
- Frameworks: Vue, React or any libraries you're comfortable with
- Tests: nice to have, but not mandatory
- Docs: nice to have, but not mandatory

Exercise: Building an Alert form

Feel free to use a platform like <https://stackblitz.com/> to avoid setting up everything by yourself.

Your REST API will need:

- an endpoint that will receive recipients list and alert details and console them out (no need to implement email sender)
- an endpoint that will receive alert details and console them out (no need to build storage of any kind)

Your form will need:

- Condition when to send alert. You should be able to specify when you want to get your alerts
- Alert message. You should be able to add an alert message
- Recipients list. You should be able to add multiple email recipients
- Test button. When you click test button your application should hit test alert endpoint that will console out alert details
- Save button. By clicking save button your application should hit save endpoint that will console out alert details

The screenshot shows a web browser window with the address bar displaying 'website.com' and a 'Logdna' tab. The main content area contains a form titled 'Email Alert'. The form has the following fields and controls:

- Alert Message:** A text input field.
- Frequency:** Two radio buttons labeled 'Hourly' and 'Daily'.
- Email Recipients:** A text input field.
- Email Recipients List:** A list of email addresses with checkboxes to the right:
 - bill@logdna.com x
 - john@logdna.com x
- Buttons:** Two buttons at the bottom labeled 'Test Alert' and 'Save'.

And yeah, do not forget about data validation layer

We evaluate your ability to:

- Understand task scope, and prioritize to achieve the production ready result within a limited timeframe
- Choose the right tools (packages, libraries, components, etc.)
- Organize folder/component structure that is easy to navigate
- Decompose the task into a set of subtasks
- Write maintainable and understandable code

We do not evaluate:

- Pixel perfectness
- Features outside of scope
- Knowledge of quirks, hacks, non-standard behavior (no need for crossbrowser support, etc.)
- Memorized knowledge (feel free to use google, docs, stackoverflow, call a friend...)
- Knowledge of a particular package or library

Always feel free to ask questions!