Recruitment case: Frontend Developer - Using DI Adresshelper API with example

Background

The aim of this task is to assess the way you approach problems, including your coding style, expertise and willingness to experiment, as well as provide us with a common ground for a technical interview.

We'd love to see what kind of solution you come up with and how you approach problem solving.
There is no hard time limit set for this task, but generally a few hours should be the goal. Due to time constraints, we don't expect all the bells and whistles and you're encouraged to focus on your core strengths and things that you think are important for production service — you may leave notes and TODOs if there are parts of the implementation you didn't have time to complete. This task is something that one could use a lot of time on in general, so we understand that it might not be perfect!

You will be using DI's API in a limited time, namely our route Adresshelper API. Your feedback is important to us, so let us know what you think about the task — before you started or after you're done ☺️

Preparations

You can have a closer look at the Addresshelper API documentation and what it does here:

DI Address Helper V2

The API itself is a huge collection of addresses that can validate official and unofficial tasks in the nordic countries. Your task is to familiarise yourself with the API and then create a UI towards the API. The goal of the task is to implement an addresscheck UI where users can validate their address(es).

The UI should be implemented using React and the users should be able to fill in one or more addresses and validate these towards the API. The response to the client should be of your choice, but it could be all from a ✅ to the longitude/latitude of the address(es) i.e. libraries, components and design to be used is entirely up to you but the assignment should follow these guidelines:

- React - **Required**
- Typescript - **Required**
- Any build tool, e.g. NPM, Webpack, Turbo, etc. - **Selectable**
- Libraries to use - **Selectable**
- Testing approach - **Selectable**
- A README that documents how to set up and run the application, how you approached it and what you would improve.
- Whether you are submitting your solution via GitHub or compressing the solution into one zip file we would appreciate if you include the Git commit history. This provides us with a view on how you break down problems and how you work.

Important information:

We have created a API key for you! Please use the TEST environment and this key:

```
905679e0-2da7-4be1-94a2-23646d8d3488
```

Example query:

```
curl 'https://staging-ws.di.no/ws/json/addressHelper/v-2/NO/streetSearch/akersgata?apiKey=905679e0-2da7-4be1-94a2-23646d8d3488'
-H 'referer: http://localhost:3000'
```

Expected time to implement

There is no expectation to use a lot of time. You should not feel like you need to spend more than a few hours on this. We do not expect all the bells and whistles to be present, but during the second interview feel free to elaborate on any ideas you might have had during the task that you were not able to implement.

How we evaluate

When reviewing your solution we try to make sure we're consistent in our evaluation by following five core themes:

- **Correctness** - We don't expect you to handle all the possible edge cases but we do expect the solution to adhere to the core requirements laid out in this document.
- **Documentation** - We would like to see your approach documenting your solutions! This can take many different forms, and we are curious to see how you go about this!
- **Testing** - We will have a look at your approach to automated testing.
- **Readability** - Writing code in a team requires having empathy for how other team members will interpret that code. Here we look for things like duplication, method names, variable names and consistency.
- **Application architecture** - We will look at the structure and architecture of your solution.

Questions we may ask

- What framework(s) and libraries did you use to implement the app? Why did you choose them?
- What would you add or do differently if you had more time?
- How would you implement CI/CD?
- How would you approach translation and localization?
- How would you implement tracking and analytics?
- What testing did you do and why?

Good luck!