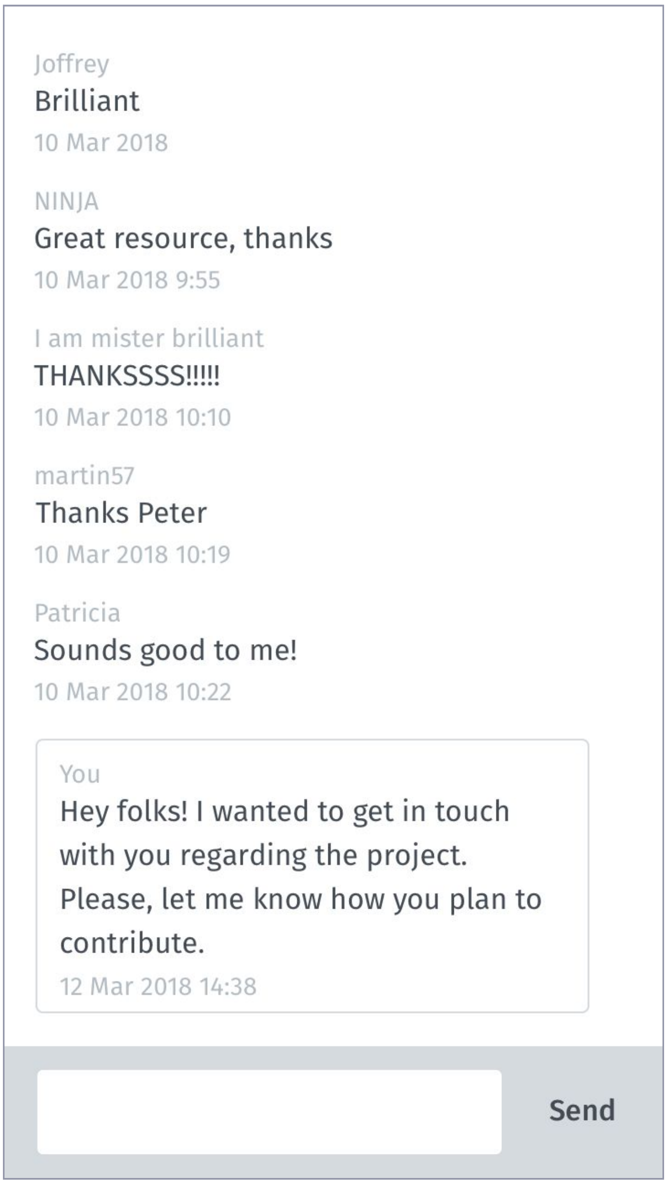
Doodle

Full-stack coding exercise

Congratulations! Your application for the full-stack position piqued our interest. You can code – we trust you on that one. However, we'd like to see your style!

We want you to build a simple chat application which is able to send messages and display messages from all senders:



Joffrey
Brilliant
10 Mar 2018

NINJA
Great resource, thanks
10 Mar 2018 9:55

I am mister brilliant
THANKSSSS!!!!!
10 Mar 2018 10:10

martin57
Thanks Peter
10 Mar 2018 10:19

Patricia
Sounds good to me!
10 Mar 2018 10:22

You
Hey folks! I wanted to get in touch with you regarding the project. Please, let me know how you plan to contribute.
12 Mar 2018 14:38

Rules and such

We understand your time is precious and would not want you to spend more than 3 hours on this over the span of one week max. The outcome should be runnable locally on a UNIX-flavoured OS in a standard browser.

We'd like you to create a simple backend in Java or Python with an API for messages that reads and writes data from a database, and a frontend in Javascript to interact with that API.

Please host your code repository online (GitHub, Bitbucket, or any other) and share it with us once you have finished.

Building the API

Please, use Java or Python for the implementation. for The API should have two capabilities:

- Receive new messages from the client
- List all messages in chronological order for the client

Building the frontend

Go easy on this one. No perfectly polished CSS is needed. Just present the messages as depicted in the wireframe.

Please, use JavaScript (Vanilla JS, React, ...). We want you to provide a responsive implementation. Keep in mind that Doodle is used worldwide and has to work on commonly used browsers.

What are we after?

It is ok if you don’t complete the assignment to every bit of detail – we don’t expect you to. Try to prioritize your solution by what you think is more important. Tell us what motivated your technology choices, how you tackled the task, what you would do differently were you given more time, what you would differently a second time around, etc.

Here are some pointers for you:

- Commit often, write useful commit messages
- Code readability
- Concise API design
- Performance: it should load reasonably fast on a mobile device
- Design: we are not looking for pixel-pushing, but we love attention to detail

Next steps

We will carefully review your exercise in a timely manner, and might then invite to an interview, where we can discuss your solution, and explore what you would do differently.