Behave Health

Landing page: http://behavehealth.com

We expect you to work full-time with us, without having any other client while you work with us. We are happy if you want to work more than 40h, but we have a limit of around 50h - 55h per week. If you want to work more than that, you must demonstrate that you are still efficient.

Intro

Behave Health is a well-funded **startup** developing a SaaS Electronic Health Records and Practice Management platform for use at substance use disorder treatment centers within the US. Our platform streamlines management and operations for our customers' facilities, empowering providers to spend more time offering care and less time dealing with administration.

Our team is spread on three continents (North America, Europe and Asia). We are a team of 9 people, 5 non-technical, and 4 technical. As within most startups, our team members take upon multiple roles - we don't consider people limited by their job description, and we encourage this, for the right kind of skillsets.

We are just about to launch an alpha test, with our first client.

Our stack

Our stack consists of two web applications, based on Angular 1.5.x, a restful RoR 5 API, and 8 iOS applications.

The web applications are using John Papa's styleguide, with some custom flavours. It's written in a way which would allow us to upgrade to Angular 2/4+ in an easier manner (we added Typescript, we don't use directives, structured as components).

The API is what's fueling the client applications. We outsourced our login/authentication to Auth0, but besides that it's pretty straightforward. Restful CRUD operations, same API for both iOS and web clients. PostgreSQL 9.6+ database, Redis + Sidekiq. Development is hosted on Heroku, and Production is on healthcareblocks.com (DevOps as a service).

The iOS applications are written in Swift 3.x.

Note: Our test coverage is extremely low. Since we didn't exactly know the form of our end product, we took an iterative approach based on feedback received from our future clients. Now we got to a point where features stabilised and we introduced "Tests Friday" - every Friday we write tests for our past work.

Our process

Weekly review call
Each week, on Monday we have a rewiew call with all the product team at 10am PDT. During the meeting, within 5-10min we each present all the work we have done during the past week, and we mention what we are planning on finishing during the current week.

Besides the weekly check, we have calls between the technical members on a need basis. Usually this happens almost every day. We always keep in touch between us, we mention if we're going to be AFK and we set presence expectations beforehand.

New features
Each feature starts with an idea or a requirement from our clients. We setup a meeting with the parties involved in the feature (Clients, Business, Therapists, Billing, Designers, Developers) and we start gathering the requirements for the feature - we go through imagining an initial version of that feature, from a technical point of view, and we set any technical limitations we may have from the beginning. Based on this initial meeting, we start designing the feature with the constraints we have defined and iterate with the parties involved. The design consists in writing the front-end code (we don't really use Sketch much, since we already have our UI framework written pretty well, so we design things directly in code.

After the design is signed off, we spec it out. This is being done either with a screencast video, explaining how the web client should work, with a Skype call + notes, or with a written Paper document. This spec covers the feature in a comprehensive way, which would allow developers to fully implement it. This is usually being done by design or product manager.

Your role

We're looking for a like-minded talented, product focused full-stack developer/engineer, who can deliver complete, well thought out solutions for features, and it's not afraid to eventually lead the development of new ones. We expect you to take ownership over these features, slowly increase your stake within other parts of the systems and take decisions together with the other team colleagues.

We would prefer having a longer relationship, and we're open to discuss fair arrangements if we all agree continuing working together.

Starting with

To help you get more comfortable with our codebase, the first feature that you will be in charge of will be to implement a more complex system for user roles and user permissions. In order to properly fulfill this task, you will have to understand the majority of our system.

To give you an idea of what this consists of, I prepared a short overall description of the task:

Context
The current system consists of 10 different user roles, each with its own permissions. The number of the user roles will increase in the future. Permissions are being saved within the database, and they are based on both actions that the user can take and the user interface. Granularity is an important

aspect of this problem.

*For example, lets take a House Manager, which is allowed to **see some** of the demographics data of a patient, but it is not allowed **to edit it**. In this particular case, we have permissions on the data that the user role can see(some demographics), and we also have permissions on the actions that they can do (edit).*

Most permissions are based on CRUD + other additional functions. When we think in the context of the whole application, this quickly adds up to hundreds of checks. You can see an excerpt of how the permissions are currently being saved in the db here - https://cl.ly/kRTI.

Requirements
The permission list should either be defined by us (as a JSON), or generated automatically based on certain data points that we pre-define.

*Important **Permissions** requirements:*

- *the permissions should be checked for and applied on both the user inteface and the backend functions*
- *be able to update the permissions of a User Role, from a UI interface, found within the web application - the data being kept/saved inside database*
- *be able to override the general role permissions for employees on an individual level*
- *be able to setup permissions/limit data between relations (eg. Therapist has access only to certain Patients - found through a relation)*

*Nice to have requirements for **Permissions**:*

- *have permissions control access on the fields that are being sent through the serializers to be displayed on the front-ends*

*Important **Roles** requirements:*

- *refactor the employees to abstract out similar functionality to the same entity*

*Nice to have requirements for **Roles**:*

- *get to a point where we can automate the generation of user roles (right now, each user role is considered a different entity, as we didn't wanted to get to a point where we had a 100 column polymorphic table). We considered this a better problem to solve, after we stabilise the user roles, but maybe there is a better approach for this.*

Interview questions

1. Are you ok with being exclusive to us, like mentioned at the beginning of this doc?
2. What are your thoughts about this document? Did anything you've read trigger any beware/strange signals?
3. What is your professional story? How did you get to do what you do?
4. How much experience do you have with our current stack - RoR APIs/AngularJS 1.5x/Angular 2+? What is your current development environment - OS/IDE?
5. Do you have experience architecting APIs and web applications? Can you shortly give me some examples of what you did and the size of these?
6. Are you an opinionated developer? If yes, how do you approach your colleagues regarding your thoughts and opinions?
7. Have you implemented complex user roles and permissions, with the permissions being set/saved within a database, and updatable?