

Backend Coding Challenge

Hey Aragats!

First of all, thanks for your interest in working at Outfittery!

We'd like to propose a coding challenge to you. The purpose of this challenge is to:

1. enable you to have a basic understanding of a part of our product;
2. let you interact with people you'd be interacting with on a daily basis, and have close-to-real discussions with your (hopefully) future colleagues;
3. let you demonstrate your understanding of API design principles and coding skills;
4. give us a glance of those softer skills of yours.

We want to not only make sure you're a good fit to our needs, but that we can also provide a place where you'd like to work!

Today you'll be invited to a Slack channel with all of us. If you have any questions along that period, please don't hesitate to ask!

Besides your coding skills, we'll be assessing:

- your interaction with your soon-to-be colleagues;
- your understanding of the task;
- your challenges to the task (if any);
- how you communicate & document your intents and decisions;


Please notice that this is not a "closed" challenge. As a matter of fact, it's still quite "raw" and experimental, so, if you find that any of the artifacts we provide to you should be changed and improved, go ahead and do it. We just ask you to please let us know about it, so that we can continuously improve ourselves :)


Now let's get to the task.


A little bit of context: Booking a call with a stylist


If you have tried our product already and tried to order a box, you might have been asked if you'd like to book a call with a stylist, as to have a better understanding of

what you're looking for and how to pack the best possible box for you. This is how that screen looks like:


Style profile



About you


Order



When should your stylist call you?

The more you share, the easier it is for your stylist to pick out clothes you'll love.

 I don't want a call ☐

< February 2018 >

Su	Mo	Tu	We	Th	Fr	Sa
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	1	2	3
4	5	6	7	8	9	10

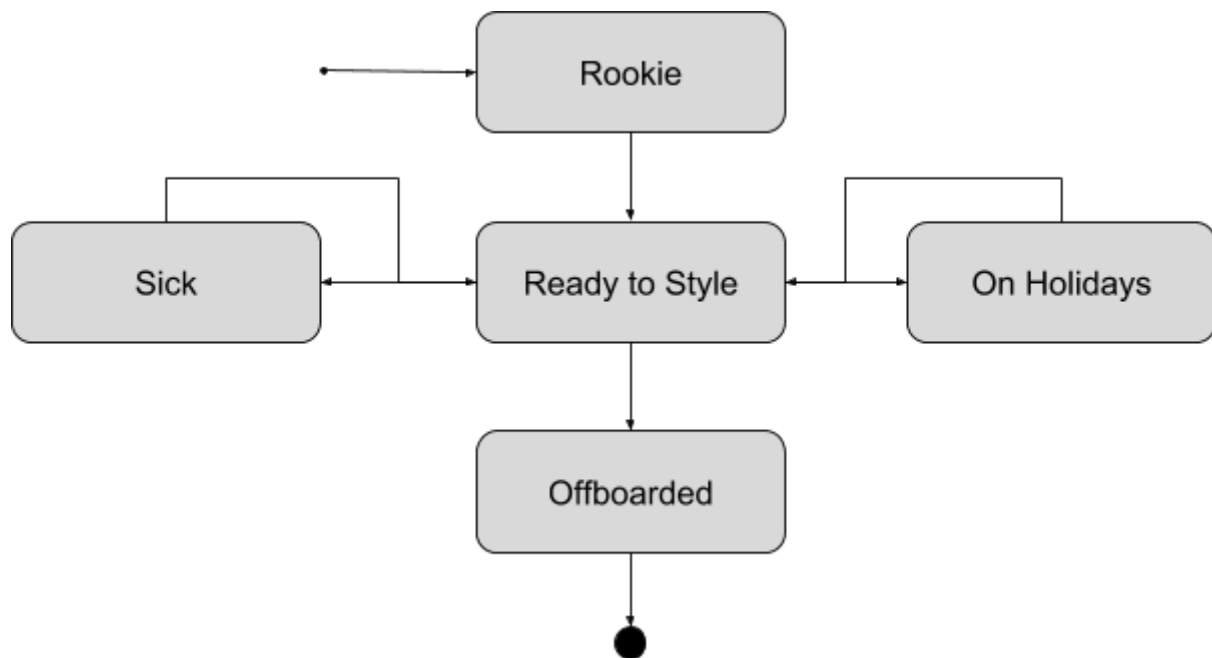
Choose time

08:30

For this page to be displayed, we need an endpoint behind the scenes that will check the availability of our 150+ stylists and offer our customers their available time slots.

As you can imagine, nothing in life is as simple as it seems. After all, stylists are human beings like you and me: they need to be trained to perform their duties, they get sick, they go on holidays and eventually some of them need new challenges and wish to leave the company.

This very simplified stylist lifecycle that we just described could be depicted as follows:



So, when a stylist joins the company, she begins as a **rookie**; meaning, will spend some time being trained on their duties and can't take any calls.

Once their training is done, the stylists are **ready to style**; customers can book calls with any of them.

Nevertheless, if a stylist gets **sick**, goes out **on holidays** or resigns (meaning, she's **offboarded** from our staff), they obviously won't be able to take any calls. Hence, their appointments need to be reassigned to other stylists so that customers still get their calls with a stylist and get a perfect box!

But that's not the only way to book a call with a stylist.

I'm pretty sure you've heard about Excel. Apparently it's very popular in some companies, especially when people need to upload a set of records to create orders ;)

Outfittery is no different: we have a system which our customer service fellows can use to upload Excel sheets and create orders for multiple customers at once. Our customer service doesn't have information about stylists availability at hand (bear in mind that this is very volatile information) and so they just upload the orders and let the system automatically book a call with the first available stylist.

The task

We would very much like you to implement a RESTful API that enables:

- internal users to manage stylists readiness to work;
- customers to see the list of available time slots (as per previous screenshot);
- customers to book a call on a specific time slot;
- automatic booking of calls for orders uploaded via spreadsheets as described above;

That entails not only proposing endpoints to enable those actions, but also:

- Implementing the small state machine we described before;
- Identify and implement elements of domain model that are necessary and sufficient to cover the functionalities described so far (such as **orders**, **customers**, **stylists**, **time-slots** etc.)

Things to keep in mind:

- We're talking about an Internet business, hence between a customer visualizing a time slot and confirming the selected time slot, another customer might have already taken that particular slot.
- You're not required to deliver everything, but in case you run out of time, we'd like to know what was your overall plan, what did you leave out, why did you leave that out and what did you plan to do next; prioritizing what gets done and what's left out is also part of the task.

We will not give you many hints on the properties of our domain models.

Identifying the need for them in the system is your task, you are free to use your imagination.

It's fine if you come up with just a bare minimum of properties per class to make general sense.

The model does not have to be fully realistic.

Simplifications that you can use:

- **Customers and stylists are characterized just by the ID.**
Things like email, name, country, etc. - you are encouraged to put into the model, but the only purpose to that should be to make it feel like a real thing. *E.g.* seeing names of people instead of just numbers helps to make input/output more human readable and easier to follow by the audience during the presentation.
- **Each customer can be handled by any stylist.**
There shall be no limiting factors like: language used by customer, his age, style preferences, VIP/celebrity status.

- **The application does not have to deal with Excel files.**
You can assume that some other software reads Excel file and then calls your application's "add many" endpoint.
 - **Time-slots are of equal length and start at fixed intervals.**
Typically our stylists mark time of day when they can have appointments. They block time for lunch or meetings. **You don't have to care about it.** Assume that timeslots are 30 minutes long, there is 16 of them in the day (indexed 0-15) between 9:00 and 17:00 and stylists work 7 days a week (ouch!).
Keep in mind that a time-slot is no longer offered to the customer only if all stylists are already booked at this time.
- It might be easier to limit stylist schedule to X days in the future only - you are allowed to do so (in fact we also plan only next 2 weeks for each stylist).
- **Ignore the fact that changes in Stylist state affect already booked appointments** - in real world, if the stylist goes to sick leave or vacations, all appointments should be reassigned to somebody else. A feature like that would be too much for the time constraints of this task, skip it.

The solution

First of all, whatever tech stack you pick, the solution should run. Every feature that you decide to deliver, should run end to end.
It would be a nice touch if your solution is not only easy to build, but also easy to run.

When it comes to programming languages, we kindly ask you to use either Java, Groovy or Kotlin.

When it comes to build tools, REST implementations, DI containers, it's your call. If you want to go with Spring Boot, go for it. If you're more of a Spark guy, just do it. If you want to go wild and use Struts 1.2, we'll take it (but you should go see a doctor, like right now. Seriously, I mean it).

You might be asking yourself: am I expected to do all of this and still write unit tests? The answer is: it's your call. As we said before, deciding on what gets done or not is part of the prioritization and we'd like you to document your decisions, either they related to prioritisation, design choices or implementation choices. We trust your judgements but you should also expect us to question them :)

By the end of your test period, you should make your solution available in a GitHub repo.

Good luck!

As previously stated, feel free to ask questions, suggest improvements... our intent is to assist you in completing your task in the best possible way.

We wish you the best of luck and hope you'll start working with us soon!

Your feedback

(completely optional, feel free to ignore it)

Hi! We hope you have enjoyed this challenge!

This task is still quite raw and experimental. As much as we'd like you to have the best experience with it, we reckon that it might have some flaws.

So, in the spirit of continuous improvement, we'd like to ask you what were your general impressions while working on it.

So, how was your experience with this challenge? What did you like about it? What didn't you like about it? What could be better? Did you have too little time to work on it? Or did you think the timeframe we gave you was enough? We'd love to hear back from you!

Once again, thanks a lot for considering working with us! Best of luck!