**Backend engineering challenge**

This challenge is used by DOT infotech for evaluating candidates for backend development positions.

This challenge is a chance for engineers at DOT Infotech to see how you code, organize a project and gather information to implement a specification.

**Deliverables**

The expected deliverable is a fully functional project that includes the following:

* Code of the application
* Tests of the application
* Documentation for launching a development environment and running the application

**Technical stack**

* [Node.js](https://nodejs.org/en/) with a framework of your choice, optionally you can use [Typescript](https://www.typescriptlang.org/)
* A relational database of your choice

Except for these requirements, feel free to use whichever libraries, frameworks, or tools you deem necessary.

**Expectations**

Your code will be reviewed by multiple engineers at DOT Infotech and will serve as the base for a discussion in interviews.  
We want to see how you approach working on a complete project and strongly recommend that you work on this challenge alone. We expect the code to be professionally structured, commented, documented, and tested.

If anything is unclear, feel free to ask any questions that might help you understand the specifications or requirements better.

**Delivery**

Your application can be sent to us as a GitHub repository or as a compressed archive containing all the deliverables.

**The challenge**

You're building a Preference Center for your users where they can manage their choice regarding the channel they want to get notified on. They can choose between getting notified by email, SMS, neither, or both. To do this you need to provide an API to manage your users and their consents.

A user will be identified by an email and can have multiple consent change events that modify the user consent status. Each choice made by a user generates a consent change event. A consent change event belongs to one user.

For audit purposes, it is expected to always have a full history of the events that led to the user's current consent status, as the user can make changes to their consent at any point in time.

The goal of this challenge is to build the simplest possible consent and user storage API. If you believe there are edge cases that are worth discussing and too complex to implement for the challenge feel free to create a list of them and we will discuss them during the interview.

**Product specification**

The API you are building supports CRUD operations on user and event entities with the following rules:

* A user can have multiple events that, when applied in the order of their creation, will generate the current user consent status.
* A user accepts only one required field (email) that must be a valid email address and unique. If any of the requirements are not satisfied, the API must return a 422 response.
* Consent IDs can be one of the following: email\_notifications or sms\_notifications.
* Consent change events can only be read and created, not update or deleted.
* A consent change event belongs to a single user.

To keep the challenge as short as possible the mandatory routes and methods are the following:

* /users [GET, POST, DELETE]
* /events [POST]

**Example**

After creating a user, the request will return:

{

"id": "00000000-0000-0000-0000-000000000000",

"email": "valid@email.com",

"consents": []

}

If two events are created in this order:

{

"user": {

"id": "00000000-0000-0000-0000-000000000000"

},

"consents": [

{

"id": "email\_notifications",

"enabled": true

}

]

}

{

"user": {

"id": "00000000-0000-0000-0000-000000000000"

},

"consents": [

{

"id": "email\_notifications",

"enabled": false

},

{

"id": "sms\_notifications",

"enabled": true

}

]

}

The resulting user (GET request) will be:

{

"id": "00000000-0000-0000-0000-000000000000",

"email": "valid@email.com",

"consents": [

{

"id": "email\_notifications",

"enabled": false

},

{

"id": "sms\_notifications",

"enabled": true

}

]

}

**Review session**

After receiving your code challenge, we organize a review session with you and a few engineers from DOT Infotech. During the review session, we will:

* Ask you to share you screen and do a quick demo of the app you built
* Ask you to present your project structure and walk us through the code (the different components, the state management, etc.)
* Ask you general technical questions related to your project and backend architecture
* Do an architecture exercise where you will sketch out an architecture (think about APIs, clients, queues, jobs, etc) using <http://draw.io/>, <http://miro.com/> or a similar tool of your choice

A few examples of the topic that we like to discuss in more details:

* API formats and documentation
* Schema management and migrations
* Microservices and loose coupling
* Secrets and encryption
* Queues and event-driven architecture
* Testing