Hello Candidate!

Terms

We're extremely excited that you are interested in joining our team. As this is a test project,once you submit the code you agree that it becomes a property of Gifnote.

Purpose

The test project is designed to reflect the way we cooperate, so it includes major stuff you would do on daily basis. We're not looking for a super skilled developer on certain platform, but rather seek more universal people who can easily learn new stuff. That is why we want you to complete the test project in Node.js. The aim is that it should not take more than 1 day (8 hours) to complete this test project for experienced developers. Please submit this project back within 3 days.

Instructions

Implement a RESTful API in Node.js for an AddressBook app following the below specification.
The AddressBook app enables its users to manage a simple contact list by adding new contacts. Commit your source code to a private repository on Bitbucket or Github. When you are done invite milan@strv.com and nolan@getgifnote.com to your repository and Heroku or AWS instances. Also send us the links to your bitbucket repository and deployed backend. For deployment you can use the free plan on Azure, Heroku, AWS or any other hosting.

Good luck!

Backend Test Project - AddressBook

Your task is to implement an AddressBook backend API. Detailed specifications for the test project are provided below. We estimate that you will not need more than a single day at relaxed coding speed to implement it.

Project description

The AddressBook backend will be used by your users to perform the following tasks:

- Register new account
- Login to existing account
- Manage their contacts

Technical details

Your backend should be able to serve both mobile apps and websites using a RESTful API. The following technical requirements are placed on your implementation:

Deployment

- Deploy the application to either Heroku or AWS ECS or EC2

API

- Use Node.js
- Use a well-known framework (preferable Koa or Express) for the API
- HTTP responses should follow best practices in the industry (especially with regard to status code definitions and request/response headers' usage - you may consult RFC 2616 for more information)
- API should communicate with their clients using JSON data structures
- Use stateless authentication - you can use JSON Web Tokens and the authentication token can be permanent.

User accounts

- All user account information should be stored in either a relational database (MySQL, PostgreSQL, etc.)
- Registrations should be done with email and password
- You should implement the following functionality:

    - User registration
    - User login

Contact data

- All your users' contacts should be stored in database
- You should implement only the following functionality on backend:
    - Create a new contact
    - Retrieve a single contract
    - Retrieve list of contracts
    - Delete a contract
    - Update a contract
- Contact data can consist of name of the contact and the phone number

Review process

We would like you to think in terms of backend architecture, development processes, code extensibility and readability, and how you generally deal with the challenges you might face while implementing this API.