Full Stack JavaScript Technical Challenge

In order to be considered for the position, you must complete the following two tasks.

Note: These tasks should take no longer than a few hours. If you are unsure of anything being asked, feel free to get in touch and ask us questions.

Task 1: Coding Challenge

Prerequisites

Please note that this will require JavaScript, Express.js and Vue.js knowledge, as well as an understanding of REST APIs.

You will also need to have Node.js installed to complete this task.

Steps

1. Create a repository on Github for the task.
2. Create an Express.js app that accomplishes the following:
    a. Connects to the iTunes Search API: https://tinyurl.com/itunes-search-api
    b. Pulls back a list of albums for a specified artist
    c. Filters the results so there are no duplicate albums (based on album name)
    d. Serves the filtered results to the front-end via a route
3. Create a Vue.js app that accomplishes the following:
    a. Makes an API request to the above route to fetch the albums
    b. Displays the albums (as thumbnail & title) in a grid
    c. Has a live search box to filter (on client-side) the currently displayed albums
4. Create unit tests as appropriate, with the testing framework of your choice.

But wait…

We are looking for someone who not only completes a project to the specified requirements but also makes use of the newest technologies as well as bringing new ideas to a project. So feel free to add in anything that you would like to share with us during the next stage of the interview process.

Task 2: Analysis Challenge

Prerequisites

Please note that this will require JavaScript, Express.js and Mongoose knowledge, as well as an understanding of REST APIs and best Node.js development practices.

Overview

Below is a Node.js function that a developer has written. It is an express middleware that processes users' invitations to use private shops.

- req and res are the express request and response objects
- superagent is a module that makes http requests and is on npm
- "User" and "Shop" are mongoose models

Step 1

Analyse the function below and provide answers to the following questions:

- What do you think is wrong with the code, if anything?
- Can you see any potential problems that could lead to exceptions
- How would you refactor this code to:
    - Make it easier to read
    - Increase code reusability
    - Improve the stability of the system
    - Improve the testability of the code
- How might you use the latest JavaScript features to refactor the code?

Step 2

Provide a sample refactor with changes and improvements you might make. The refactored code does not have to be executable; it will only be used for discussion.

Once both tasks are complete...

Commit and push your code from Task 1 and your analysis from Task 2 to your new repository. Then send us a link, we will review and get back to you.

Good luck!

Task 2 Code Snippet

```
exports.inviteUser = function(req, res) {
  var invitationBody = req.body;
  var shopId = req.params.shopId;
  var authUrl = "https://url.to.auth.system.com/invitation";

  superagent
    .post(authUrl)
    .send(invitationBody)
    .end(function(err, invitationResponse) {
      if (invitationResponse.status === 201) {
        User.findOneAndUpdate({
          authId: invitationResponse.body.authId
        }, {
          authId: invitationResponse.body.authId,
          email: invitationBody.email
        }, {
          upsert: true,
          new: true
        }, function(err, createdUser) {
          Shop.findById(shopId).exec(function(err, shop) {
            if (err || !shop) {
              return res.status(500).send(err || { message: 'No shop found' });
            }
            if (shop.invitations.indexOf(invitationResponse.body.invitationId)) {
              shop.invitations.push(invitationResponse.body.invitationId);
            }
            if (shop.users.indexOf(createdUser._id) === -1) {
              shop.users.push(createdUser);
            }
            shop.save();
          });
        });
      } else if (invitationResponse.status === 200) {
        res.status(400).json({
          error: true,
          message: 'User already invited to this shop'
        });
        return;
      }
      res.json(invitationResponse);
    });
};
```