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 couple of hours. If you are unsure of anything being asked, feel free to get in touch and ask us any questions at dan.norton@oneflow.io and andrew@oneflow.io.

Task 1: Coding Challenge

Prerequisites

Please note that this will require JavaScript, Express.js and front-end 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. Implements a service that stores the following data-set: https://gist.github.com/thekiwi/ab70294c8d7ab790d9b6d70df9d3d145
    b. Serves the list of episodes to the front-end via a route
    c. Allows filtering of the results with an optional 'season' query string
3. Using a front-end framework of your choice, create a simple web app that:
    a. Makes an API request to the above route to fetch the episodes
    b. Displays the episodes (as thumbnail & title) in a grid
    c. Has a text-box to enable client-side filtering of the episodes by title
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

A developer has written some code and submitted a pull request. The PR adds functionality to enable users to invite other users to their personal shops. The code is shown below.

- **req** and **res** are the express request and response objects
- **superagent** is an NPM module that makes http requests
- "**User**" and "**Shop**" are mongoose models

Task

Analyse the code 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 unexpected behaviour?
- How might you refactor this code to:
    - Make it easier to read
    - Increase code reuse
    - Improve the testability
    - Minimize unhandled exceptions

Once both tasks are complete...

Commit and push your code from Task 1 and your analysis from Task 2 to your new GitHub 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);
    });
};
```