The Assignment

Complete the following using this front-end project and the StackExchange API:

Part 1 - question list

Create a simple question list page (`/questions`) that lists 20 questions from StackOverflow that are interesting or have activity. Documentation around querying StackExchange for questions can be found here. Clicking on a question should navigate to `/questions/:question_id` (see part 2 below).

Part 2 - question page

Create a question details page (`/questions/:question_id`) that queries the StackExchange API for question details and recreates the standard StackOverflow question page. You don't need to recreate every single element of the standard question page - see the image `StackOverflowExample.png` in this folder for details on which areas of the page you should recreate and which you can disregard. Specifically:

- DO use BootstrapCSS style definitions. We're not going to be nitpicking you on pixel alignment here since Bootstrap styles won't be identical to StackOverflow's styles, but DO try and get it close.
- DO use AngularJS components, directives, filters, and services where applicable.
- DO create the elements for the `share` , `edit` , `flag` links (below each answer) but DON'T worry about adding the functionality behind them.
- DO show the comment box when the user clicks "Add Comment" but DON'T worry about submitting to the server.
- DON'T worry about the hover effect tooltip cards for people and tags.

Requirements

Write production-quality code - we want to see how your best code looks.

Return your solution in the form of a zip archive. *Please exclude your `/node_modules` and `dist` directories.*

You may utilize any existing internet resource that you like, but don’t, for example, post a question on a forum soliciting help. When sending us your solution, please tell us what types of online resources you used (e.g. StackOverflow code samples, existing work you found on GitHub, etc.)

The StackExchange API:

Here are some relevant API calls to get you started:

- https://api.stackexchange.com/2.2/questions/327738?site=serverfault
- https://api.stackexchange.com/2.2/questions/327738/answers?site=serverfault
- https://api.stackexchange.com/2.2/questions/327738/comments?site=serverfault
- https://api.stackexchange.com/2.2/questions/327738/related?site=serverfault

The StackExchange API documentation is quite good, reference it as needed.

The StackExchange API has a quota of 300 requests per day, so watch out for that. It will also throttle requests if you make too many too rapidly.

This goes without saying, but...

Please don't get help from anyone else while you work on this project - this should be strictly your own work, though as noted above you may use online resources as long as they are properly attributed.

When you're done with this project, please don't share the assignment or your solution with anyone.

Using the project

We've provided you with a fully-functional front-end stack for this assigment; here's how to use it.

Prerequisites

First, install Node.js if necessary.

Then install the project's dependencies:

```
npm install
```

Compile the code

Use the following command to compile the code.

```
npm run build
```

This command compiles the front-end code into the `/dist` directory. It does **not** start a server that serves the compiled code, it only compiles it.

Run the app locally

The project includes a simple webpack development web server.

To start the server:

```
npm run start
```

The app will be available at http://localhost:8080/.

You must compile the project before the first time you start the development server (because the compile process generates `index.html`).

Continuous Development

The project is configured with live reload so just start coding and your browser will get updated with the latest JS/HTML/CSS. There are a few exceptions to this rule - for example, edits to `index.ejs` will require you to restart your dev-server.

Run unit tests

The app comes with unit tests. These are written in Jasmine, which we run with the Karma Test Runner test runner. We provide a Karma configuration file to run them (the file `karma.conf.js`).

The unit tests are found next to the code they are testing; e.g. the unit test file for `foo.js` is named `foo_test.js`.

To run unit tests:

```
npm run --silent test
```

The `--silent` suppresses spurious node errors after test failures.

To run unit tests continuously in watch mode (where file changes trigger tests):

```
npm run test-watch
```

Lint

We use `eslint` for linting - the configuration can be found in `eslintrc.json`. To lint:

```
npm run --silent lint
```

The `--silent` suppresses spurious node errors when linting fails.

Happy Coding!