SC Node Task

As we are expanding our capabilities at SocialCops, we have begun to build a multitude of backend microservices to support and simplify our applications.

Your task is to build a simple *stateless* microservice in Nodejs, with three major functionalities -

- Authentication
- JSON patching
- Image Thumbnail Generation

We have no requirements for which frameworks/libraries to use, choose whichever seem best suited for the task!

Required Endpoints

The API should feature the following endpoint functionality -

Public Endpoints

Login

- Request body should contain an arbitrary username/password pair
    - Treat it as a mock authentication service and accept any username/password.
- Return a signed Json Web Token(JWT, https://jwt.io/) which can be used to validate future requests.

Protected Endpoints

The following two endpoints should be protected. The JWT obtained in the “Login” endpoint must be attached to each request. If the JWT is missing or invalid, these endpoints should reject the request.

Apply Json Patch

- Request body should contain a JSON object and a JSON patch object (http://jsonpatch.com/).
- Apply the json patch to the json object, and return the resulting json object.

Create Thumbnail

- Request should contain a public image URL.
- Download the image, resize to 50x50 pixels, and return the resulting thumbnail.

General Requirements

Code Requirements -

- Include a test suite for the microservice.
    - We recommend using Mocha (https://mochajs.org/).
- API should reject invalid request inputs. Test the edge cases!
- Use modern javascript ES6 syntax.

Other Requirements -

- Use Git for version control, and host the project in a Github repository.
- Project should contain documentation with setup and usage instructions.
- Project should install all dependencies with “npm install”, should start the server with “npm start”, and should run the test suite with “npm test”.

Bonus Points -

100% code coverage in test suite.

- We recommend using Istanbul (https://github.com/gotwarlost/istanbul) to generate code test coverage reports.

Extra Documentation

- Include JSdoc comments and/or Swagger specifications to impress us.

Logging / Monitoring

- Integrate a centralized app logging/monitoring system.
- Really, please just don’t use “console.log” as the primary debugging/logging tool.

Javascript Style and Linting

- Use a javascript linter, along with a linting npm script. We like clean code.

Dockerize

- Include a working Dockerfile with the app directory.

How we will judge the task -

- Project organization and code readability (40%)
- API functionality correctness (15%)
- Input validation and error handling (15%)
- API speed and efficiency (10%)
- Documentation (10%)
- Unit Test Coverage (10%)
- Bonus Points (up to +20%)