Quixel Developer Test

This test consists of two sections:

1. A coding project
2. A subjective part

The first part should take you ~3 hours to implement. Basically it's a sanity check to make sure you can actually write and structure code. What ever you can get going in 3 hours should be enough.

For the second part, we would advise that you spend a little time thinking about the questions, perhaps keep them in your head for an afternoon and then do the write-up at your leisure. The actual typing would of course only take you an hour to do. Feel free to include any citations, references, or illustrations that you feel are relevant.

Deliverables

For the first section, you are required to give us a link to a github.com project. The project can be implemented in any language and framework that you choose; we of course do have a strong preference for NodeJS and Python.

The project is expected to be in a functioning form and relatively bug free. It should be independently deployable, a dockerized implementation (having an associated Dockerfile) would be wonderful.

For the second section you are expected to submit a word of markdown file containing your perspective on each of the questions given. Anything around a 1000 words should be enough. We'll have a discussion with you afterwards regarding the solution submitted.

Section I (Coding Project)

You are required to implement a "URL Shortening" web application (https://en.wikipedia.org/wiki/URL_shortening). It should consist of:

- A backend that at least has an HTTP endpoint for submitting a URL to be shortened for which it would emit a functioning shortened URL.
- At least a one-page (index.html) frontend that has enough of a UI to submit a URL and get back a shortened URL.

Note that this part is intentionally left open ended instead of dictating specifics. We would like to see what kind of design decisions and conventions you end up adhering to.

Section II (Subjective Part)

Kindly give your perspective on the following questions:

1. What properties should a shortened URL possess, and what architectural decisions or algorithms would you devise to guarantee those properties?

2. What do you feel are the limitations or vulnerabilities of your submitted solution? What should be done to address these limitations or vulnerabilities?

3. We would like to allow the users to specify an expiration period for the generated URL. How would you accommodate this feature into your solution?

4. We would like to give users a real-time dashboard against each URL generated, which they can use to monitor analytics regarding the usage of that generated URL. What sort of architecture would you propose we implement to do that?

5. We would like to prevent abuse of the service by making sure that only human users can submit URLs to be shortened (as opposed to spam bots and scripts). But we would also like to let other applications (e.g. Slack and Skype etc.) use our service. How do you propose we go about doing that?