We respect your time; please do not spend more than 2-3 hours on this. We care more about your thought process than the exact answer.

Part 1 - Unit test code review

Peer review is a core part of Cesium's engineering culture. As a Software Developer in Test, you would be expected to help review and improve code and tests written by other developers.

A small project, *dev-in-test.zip*, was included with this project description.

- Run **npm install** to install any required dependencies
- Run **npm run test** to run unit tests under Google Chrome

The source directory contains a single file, *Cartesian3.js*, which represents a 3D point defined by x, y, and z properties.

The specs directory contains a single jasmine test suite for this file, *Cartesian3Spec.js*. Which verifies the Cartesian3 class works as expected.

Your task is to perform a code review of the two JavaScript files and provide feedback on anything you think is missing, incorrect, or needs to be changed. This does not need to be a formal document, a list of bullet points that reference specific line numbers or functions is fine.

To the best of our knowledge, the source code itself is correct and there is no "trick" to this review. However, the unit tests may be lacking.

We recommend spending no more than 20 minutes on this exercise to allow sufficient time for part 2. Please provide your write-up as a Markdown file.

Part 2 - End-to-end testing

A major part of your role at Cesium will be in establishing and maintaining end-to-end test suites.

A small blogging application, *sample-app.zip*, was included with this project description. This is almost identical to the sample application used in the Redux tutorials.

- Run **npm install** to install any required dependencies
- Run **npm start** to launch a local development version of the app
- Run **npm run build** to build a minified and production ready version of the application
- Run **npm run start-build** to serve the production build

Take a few moments to familiarize yourself with the application. It has been modified to generate consistent data across page loads. So while the Post content and notifications may appear random, the same actions will result in the same output each time, which makes it ideal for end-to-end testing.

- The *Posts* tab allows you to create, view, and react to posts.
- The *Users* tab shows the current list of authors
- The *Notifications* tab shows notifications sent from the server. Since this is a client-side only application, all server events are simulated and you must click “Refresh Notifications” to generate data.

Create a Markdown document with a high level end-to-end automated test plan. What tools and techniques would you use? What needs to be covered? The plan should be written so another developer could understand and implement it.

Using the testing framework of your choice, implement as much of your plan as time allows. We do not expect you to have enough time to perform exhaustive testing, so concentrate on showcasing what you can finish to a high quality. You can either modify the codebase provided, or create an entirely new project that uses the application url for configuration.

You can provide your submission for both parts via a zip file or link to a publicly accessible source repository, such asBitbucket, GitLab, or GitHub (preferred).