Here are a list of a few things 18F would look for when evaluating vendor-submitted code for the challenge, in no particular order.

**Please note: This is not a complete list, guide, or checklist for evaluating a vendor prototype. There are red flags that could exist that are not covered here, and there are good practices and positive signs that are not contained in this list. This list is not meant to train an evaluator, nor is it meant to replace a high quality developer who is familiar with the languages and technologies used.**

1. Is there a defined code style? This should be apparent from the project configuration (e.g., the presence of a .jshintrc file tells me that the project is configured to use the JSHint linter, which implies at the minimum that the project is using the default JSHint style, which is a positive mark).
2. Is the code style completely custom, or is it derived from a well-known style? The most commonly used styles already cover the ground of making sure code is readable (and thus more maintainable), so there’s no reason a vendor should write their own from scratch.
3. Is the code style enforced? Having a style is pointless if it isn’t followed, and a linter tool can ensure that developers are adhering to the style. Code that doesn’t adhere to the style should never be checked in.
4. Does the code have comments? Comments will help other developers understand how the code works, reducing maintenance costs.
5. Does the code use well-known, maintained libraries and frameworks? In the example we looked at, the vendor used one of the most popular libraries in the world, express. Vendors should not reinvent functionality that already exists in well-maintained libraries due to the extra maintenance costs.
6. Does the code “flow” in a normal way? In the example we looked at, it was difficult to tell how the application started. It was not typical for an express app. This adds to the maintenance cost as new developers will have to spend extra time understanding the app’s unique flow.
7. Does the code have custom security logic? In the example we looked at, the vendor used a few well-known security libraries. Security is notoriously difficult to get right so vendors should *always* use existing libraries for the bulk of their security-related work.
8. Does the code have tests, and is the coverage adequate? Tests help ensure the code does what it’s supposed to, and help protect against future code changes breaking existing functionality.
9. Do the tests make sense? It is important to check some of the tests to make sure they actually test the system in a sensible way. Tests that don’t exercise the system properly can easily miss bugs.
10. Do the tests follow standard conventions? There are an endless number of ways to write and run tests, but the industry has come up with fairly standard tools (test runners, assertion libraries, etc.) to help. The vendor’s tests should adopt these rather than do entirely custom testing.
11. Is sensitive information (e.g., private keys, database passwords, etc.) read from a configuration file or the local environment? This information should ***not*** be in the code, nor should it be part of the source control, but should come from configuration or the environment.
12. Is it using a well known IaaS/PaaS or containerization? The app should be deployed on something like Heroku, Google App Engine, Windows Azure, or Amazon AWS, or should be developed with containerization in mind, like Docker.
13. Are they using automated configuration management tools (good) or manually configuring the server (bad)? Some example configuration management tools are Puppet, Chef, or Otto.
14. Does it at least mostly adhere to the 12 Factor Application methodology? <https://12factor.net/>)
15. Did they follow standard git good practices and usage patterns?
16. Does it have automated code quality analysis, and does that give the code a good score? For example, there are no errors and few warnings, or if using a tool like Code Climate, it has a high rating like 90% or 3.0 (GPA scale).