KSD Code Exercise

Instructions

The following is the code exercise that we give our candidates. The objective here is to understand your process and not necessarily to create a full-fledged working system. You can build your solution using any web-based technologies you wish as long as it uses a relational database on its backend (MySQL, Sqlite, Postgres, etc) and HTML, CSS, and Javascript on its frontend. A modern PHP framework would be preferred (Symphony, Laravel, CodeIgniter, etc), but ultimately the idea is to help us evaluate how you go about solving problems in a short amount of time and to give us a sense of how you organize, build, test, document, etc. Use something you're comfortable with.

There is no time-limit on the exercise; you can spend as much or as little time on it as you wish. We realize we're all extremely busy, so we don't want to have you spend too much time on this. If time is a constraint, you can lay out the framework to the components so that we get a sense of how you would approach the problem and solve it, with basic functionality in place. Please also keep track of your hours. Generally a basic application like this should be possible to build in about a day using existing frameworks and open-source libraries, but obviously that's subjective. We would like to avoid you spending multiple days on this, so if this takes much longer please let us know how far you got.

Your solution (both front and back-end code) must be committed to a github account - please do not restrict it in any way. Send us the github URL for the project once you have the code up to a good enough point that you want us to look at it, and also send a working URL. The application may be hosted on any target of your choice: your own web server, Google App Engine, Heroku, or whatever else you're comfortable with or want to play with and learn. Most services offer a free tier.

Any ambiguity or questions you may come across when building this is up for you to decide how your application behaves. Accounting on your own for inconsistencies in a specification given to you will be important in this position.

While the exercise may seem overly long, we want to reiterate that it's about how you think about the problem scope rather than wasting your time building a full-fledged application. Keep it simple; it doesn't need to be perfect.

Exercise Description

The objective is to create a simple database-driven web application for keeping track of inventory.

1. Each item in the inventory will have a name, a serial number, a type, the room it is located in, the city it is located in, and the date it was added. For example:

   Name: John's Desktop Computer
   Serial Number: 238-1338-22
   Type: Desktop
   Room: 251
   City: Kansas City
   Date: 2015-05-01

   The name, serial number, and room are arbitrarily given by the user of the application when adding or updating entries (they can be anything). The type and city should be selected from a list of available options. For example:

Types: Desktop, Laptop, Television, and DVD Player
Cities: Kansas City, Topeka, and Wichita

The date should be automatically assigned by the application when the entry is added.

2. The application should provide ways of adding, updating, or removing entries from the inventory.

3. The main page of the application should provide a means of viewing all items currently in the inventory, along with ways to filter or trim the list down to find desired entries. What filters are available or the flexibility given to the user for this ability is left up to you. Do what you think is the most user friendly and most helpful. Try to anticipate what a user may want to find, and build your application so that finding it is as seamless as possible.

There is no need for a login or permissions system, etc. Concentrate on the 3 tasks above. The application does not need to be perfect.

Front End
The design of the user interface is left up to you. Feel free to use any Javascript or CSS libraries/frameworks you like (such as Bootstrap or Foundation). It does not need to be perfect or necessarily pretty. Functionality, intuitiveness, and usability are much more important.

Back End
As mentioned, please use a modern web framework (PHP preferred) of your choosing along with a relational database you feel comfortable working with (MySQL preferred). You can write the SQL queries yourself, or use model-based abstractions from an ORM you like.