# Coding Exercise - Server

Thanks for taking some time from your busy schedule to work through this coding exercise; we certainly appreciate it. Our goal here is to learn about your coding style, use of frameworks/libraries, and approach to development in general - basically, an early look at how good a fit MIW might make for you and vice versa :)

Our servers run Java using Spring Bootstrap/Data JPA/Data REST and use MySQL for storage. The web client is built with Node.js/ReactJS and the Android mobile app is written in its native language. And who knows what the future will bring? Languages/frameworks continue to change; we pride ourselves on hiring great, adaptable developers.

We also value testability at MIW. Since teams do not always have dedicated testers we expect developers to ensure their code executes as flawlessly as possible. We still have project owners, stakeholders, and others to help with acceptance and usability testing - but the main responsibility for quality remains with the developers. We are looking towards having any estimate and deliverable for a feature include unit tests or some form of automated testing.

The exercise on the following page along with the list of deliverables. Please be sure to include answers for the fourth item and any other notes/comments/decisions in a README.md. Feel free to over-communicate in that document. This is your chance to show us what you consider to be great code (and tests)! We really enjoy seeing a passion for development surface early in this process. When you are finished, please reply to this email with a link to your Github (or similar online) repository.

Looking forward to seeing your code and answers!

# The Gilded Rose Expands

As you may know, the Gilded Rose* is a small inn in a prominent city that buys and sells only the finest items. The shopkeeper, Terry, is looking to expand by providing merchants in other cities with access to the shop's inventory via a public HTTP-accessible API.   The Gilded Rose would like to utilize a surge pricing model like Uber to increase the price of items by 10% if they have been viewed more than 10 times in an hour.

## API requirements

- Retrieve the current inventory (i.e. list of items)
- Buy an item (user must be authenticated)

Here is the definition for an item:

```
class Item
{
        public   String   Name {get; set;}
        public   String   Description {get; set;}
        public   int      Price {get; set;}
}
```

A couple questions to consider:

- How do we know if a user is authenticated?
- Is it always possible to buy an item?

Please model (and test!) accordingly. Using a real database is not required.

## Deliverables

1. A runnable system with instructions on how to build/run your application
    a. The application should be built using Java and the Spring framework
    b. The application should be able to be run from the command line without any dependencies or database's required to run your application.  (Other than maven, gradle and Java)
2. A system that can process the two API requests via HTTP
3. Appropriate tests (unit, integration etc)
4. A quick explanation of:
    a. Your application, how you set it up, how it was built, how you designed the surge pricing and the type of architecture chosen.
    b. Choice of data format. Include one example of a request and response.
    c. What authentication mechanism was chosen, and why?

* The Gilded Rose is a refactoring/TDD exercise created by Terry Hughes and Bobby Johnson. You can safely ignore it during the coding exercise.