Technical Task for Junior Java Developer position at Pragmatic Coders

Checkout Component 3.0

Business requirements:

Design and implement market checkout component with readable API. Assume that your application should be a backend tool and will be used by an on-line shop – you should provide only few parts of it's functionality. Customer should be able to:

- Open new empty basket – later the basket will be used to store items,
- Scan items – imagine that on UI side user will be able to select a product of one type and input it's quantity, remember that our goods are priced individually,
- Close basket at any time and return the total price of a number of items.

Your application should provide some extra functionality:

- Discount mechanism – buy 'n' units of one type, and they will cost you 'y' dollars – take a look on 3rd and 4th column in the table below. Take into consideration that shop might want to extend the discount offer in future (eg. buying A and B product will allow customer to take 10% discount) – you should provide a proper mechanism or design pattern.

Also take into account that:

- You should use repository to store state of customer's basket. For the needs of this task it can be 'in memory' database.
- The table below is only an example state of shop's storage at the time of writing your application – you can change or extend it, eg. change names of products to descriptive ones. The initial storage state shouldn't be hardcoded into your application (Java) code - provide additional file (eg. .sql) that will be used by StorageRepository to create an initial state of the repository.

Item	Price	Unit	Special Price
A	40	3	for 70
B	10	2	for 15
C	30	4	for 60
D	25	2	for 40

Technical requirements:

1. Concentrate on well designed, tested, structured and clean code - discount calculation logic is not the main topic of this task.
2. The output of this task should be a project with buildable production ready service, that can be executed independently of the source code.
3. Project has to include all tests, including unit, integration and acceptance tests.
4. The service should expose REST api, without any UI.
5. You can use gradle, maven or any other building tool.
6. It ought to be written in Java 8.
7. If requested, please use Spring or any other framework, otherwise, the choice is yours.
8. Please include a short description of how to build and execute your project in e.g. README.md file. It it's up to you to make your own assumptions about the code and assignment. Each assumption should be captured as part of README.md file, so that we can review them before reading a code. There are no good and bad answers here. You can put there any other information that you would like to share with us like important classes, design overview, the rationale for your choices etc.