**Amazing - Code Challenge Solution**

Read Me

Author: Kamal Saluja

Date: 11-Feb-2019

Table of Contents

[Document Objective 3](#_Toc876873)

[Problem statement: 3](#_Toc876874)

[Assumptions 3](#_Toc876875)

[Rules Configured: 3](#_Toc876876)

[Solution Structure 5](#_Toc876877)

[Steps to setup solution 6](#_Toc876878)

[Test Solution 6](#_Toc876879)

## Document Objective

Objective of this document is to provide information about the solution for coding challenge. It contains details about problem, proposed solution structure, technical stack & key features (ES 6), sample data and some nice to have features i.e. can be further included with the solution.

## Problem statement:

Amazing Co is a managed marketplace for experiences and helps customers with unique and amazing experiences for everyday activities and special occasions.

Solution is required to produce technology platform for registering events details with promotional offers. Such offers should be configurable and can be modified in future easily.

## Assumptions

Few assumptions have been made during development to ensure desired function to run efficiently.

1. Currently four types of events provided in coding challenge are hard-coded in API layer which can be wrapped with database layer.
2. Based on ruleset provided in requirements, below is the matrix of rules is configured with proposed solution. Currently solution is running with one rule per offer however it can be implemented to produce combo(s) of offers per product (event).

## Rules Configured:

Below are the rules configured to support the calculation of price after discounts. Currently this is configured in application (Config -> Default.json) file, where item\_<<prodId>> is configured for below items with rules mentioned in below table.

{ prodId: 1, eventName: 'Kids Party', price: 220 },

{ prodId: 2, eventName: 'Wine Tour', price: 440 },

{ prodId: 3, eventName: 'Team Building', price: 800 },

{ prodId: 4, eventName: 'Picnic', price: 110 }

|  |  |
| --- | --- |
| Rule | Description |
| fixedDiscount | If this is rule is set to ‘true’, final fixed discount will be deducted on per minimum quantity.  E.g.  As per offer, if ‘Team Buildings has base price of $800 and has offer of ‘Buy 2 for $1500’. Rule is configured as below  "item\_3": {  "fixedDiscount": 1500,  "minimum": 2,  "addQuanity": false,  "percentOnSingle": false,  "percentOnAll": false  },  If user order 5 ‘Team Buildings’ the price will be calculated as  Total Price = 5 \* 800 = $ 4,000  Total Price after discount = 2 \* 1500 + 800 = $3,800 |
| Minimum | This describes minimum quantity to be ordered to claim offer |
| addQuantity | This will add extra one item on every minimum quantity (i.e. buy 2 get 1)  E.g. ‘Picnics’ has base price of $110 and has offer ‘Buy 2 get 1 free’, rule is configured as below  "item\_3": {  "fixedDiscount": false,  "minimum": 2,  "addQuanity": true,  "percentOnSingle": false,  "percentOnAll": false  },  If user order 5 ‘Picnics’ the price will be calculated as  Total Price = 5 \* 110 = $ 550  Total Quantities (offer) = 7 (after adding two more quantities) |
| percentOnSingle | This will % discount on one item after minimum order  E.g. ‘Kids Party’ has base price of $220 and has offer ‘Buy5 and get 20% on 5th’, rule is configured as below  "item\_3": {  "fixedDiscount": false,  "minimum": 2,  "addQuanity": false,  "percentOnSingle": 20,  "percentOnAll": false  },  If user order 5 ‘Kids Party’ the price will be calculated as  Total Price = 5 \* 220 = $ 1,100  Total Price (after discount) = 1,100 – 44 = $1,056 |
| percentOnSingle | This will % discount on overall order after minimum order  E.g. ‘Kids Party’ has base price of $220 and has offer ‘20% on overall’, rule is configured as below  "item\_3": {  "fixedDiscount": false,  "minimum": 2,  "addQuanity": false,  "percentOnSingle": false,  "percentOnAll": 20  },  If user order 5 ‘Kids Party’ the price will be calculated as  Total Price = 5 \* 220 = $ 1,100  Total Price (after discount) = 1,100 – 220 = $880 (20% on overall order) |
|  |  |

## Solution Structure

Node JS is used to implement the solution for provided problem. Below are some key java script files are created.

|  |  |
| --- | --- |
| **File / Folder** | **Description** |
| /index.js | Entry point to kick application server, host express settings and API endpoints |
| /Bootstrap | Contains files (express-settings & components) to bootstrap API endpoints. |
| Components / Controller / LeaveController.js | This file is used to build couple of API endpoints   1. Http Get – getAllProducts() i.e. fetch all products from database. 2. Http Post – takeOrder() i.e return order quotation to display offer details / calculated price on requested order.   Idea behind creating such folder structure is to place all endpoints under Controller folder and then create index.js under Components folder which can be helpful at time of bootstrapping. It will allow single import statement to load all modules from controllers.  e.g. ***const controllers = require(‘../Controller’)*** will point to index.js that will load all modules at single line. |
| /Models | Contains classes to be instantiated by API endpoints to prepare results. All offer logic is encapsulated under models. |
| Services / ProductService.js | This file contains class i.e. responsible to make connection to database and fetch products details. Currently it’s hard-coded.  { prodId: 1, eventName: 'Kids Party', price: 220 },  { prodId: 2, eventName: 'Wine Tour', price: 440 },  { prodId: 3, eventName: 'Team Building', price: 800 },  { prodId: 4, eventName: 'Picnic', price: 110 } |
| Services / Index.js | This file provides single instance for Product Service. |
|  |  |

## Steps to setup solution

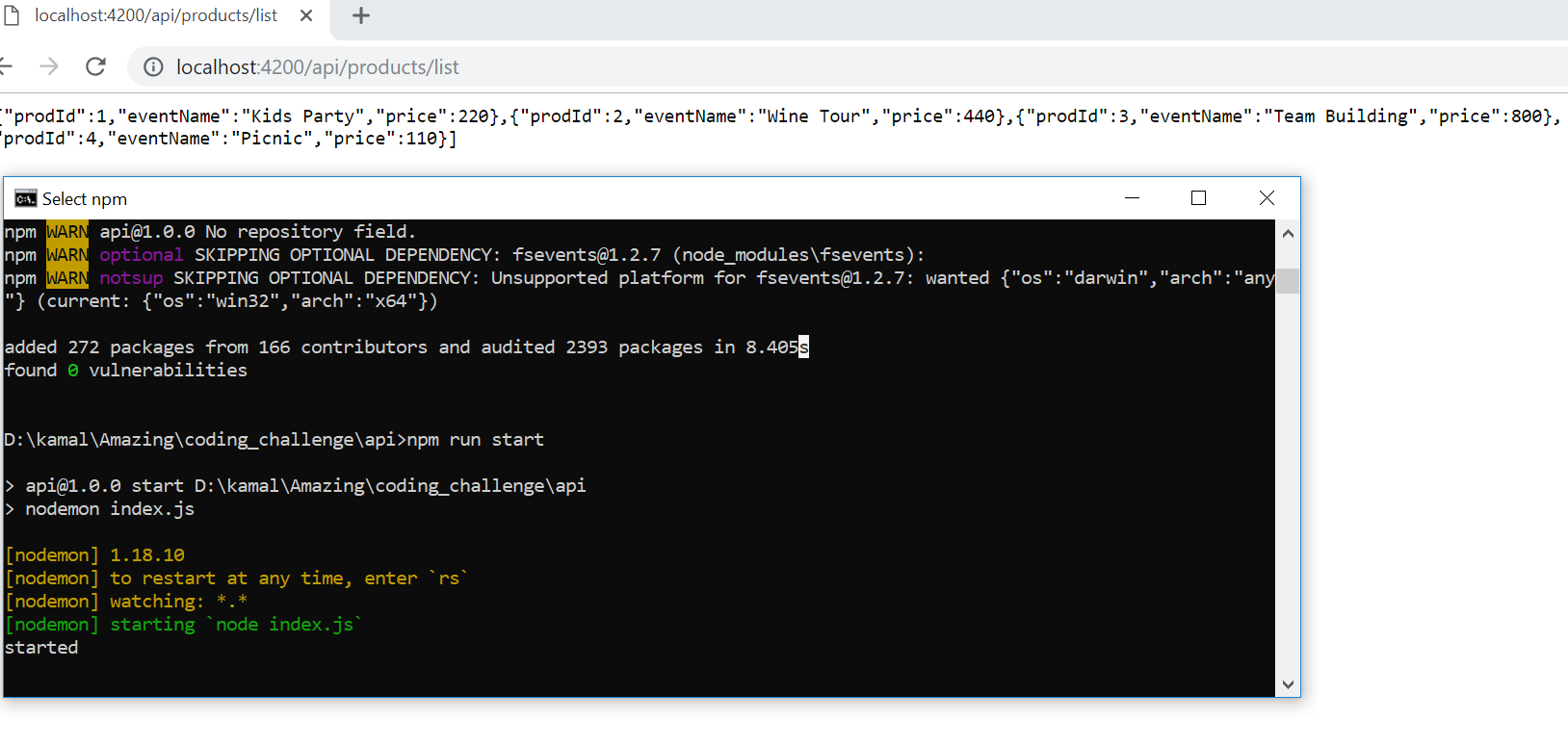
Below are the steps to execute the solution and see the results.

1. Clone repository from url <https://github.com/kamal-kamal/coding_challenge>
2. run ‘npm install’ to install dependencies
3. run ‘npm run start’ to start the project on port 4200
4. Open browser to <http://localhost:4200/api/products/list> to see if endpoint working & fetches products list.
5. Discount offers calculation can be executed Postman (please see *Test Solution* section below).

## Test Solution

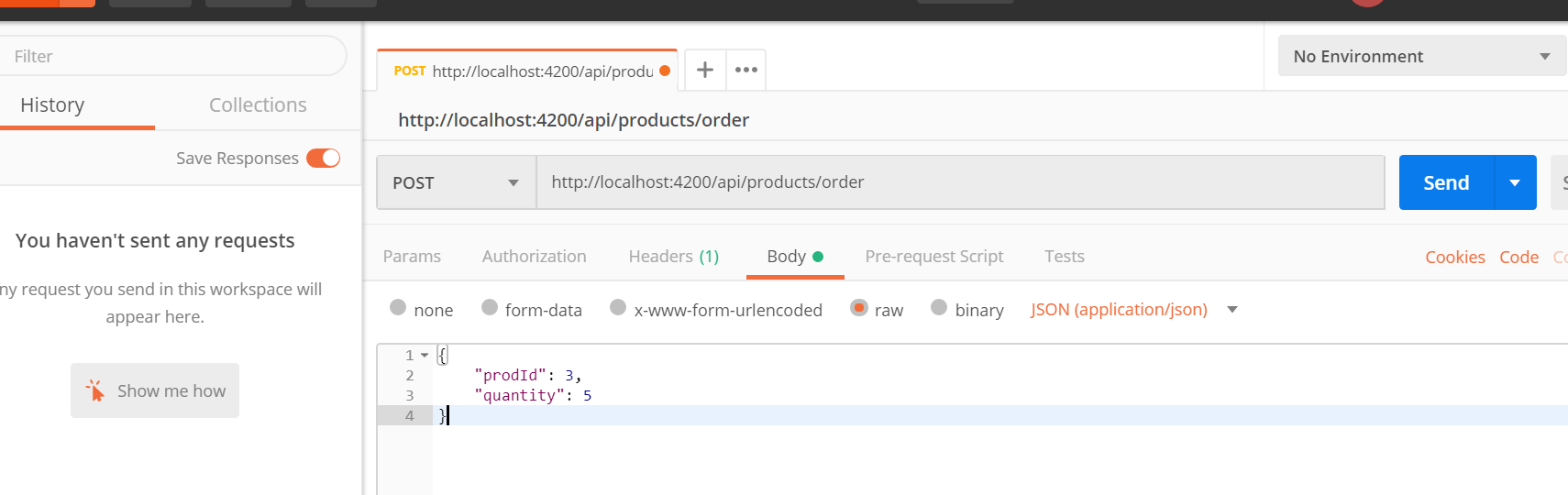
Scenario 1: list out all products

Open any browser and enter url <http://localhost:4200/api/products/list>

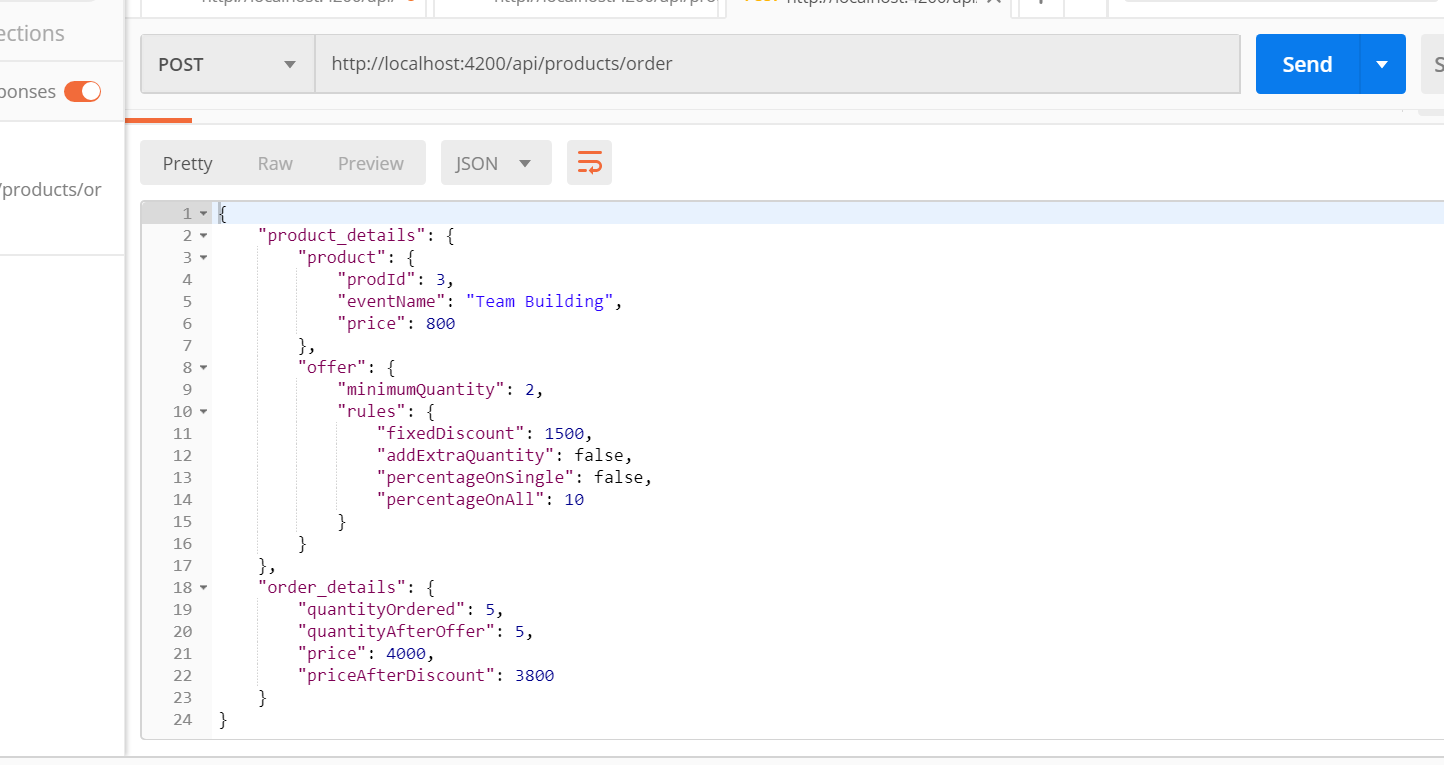


Scenario 2:

Post request via postman.



Output:



Product\_details: Json snippet provides the details about product

Offer: Json snippet provides details about rules configured for the product id

Order\_detais: provides the information about price, discount and revised price / quantity after applying the rules.

Here in this example the configuration is as below

Product requested:

{ prodId: 3, eventName: 'Team Building', price: 800 },

Quantity requested: 5

"item\_3": {

"fixedDiscount": 1500,

"minimum": 2,

"addQuanity": false,

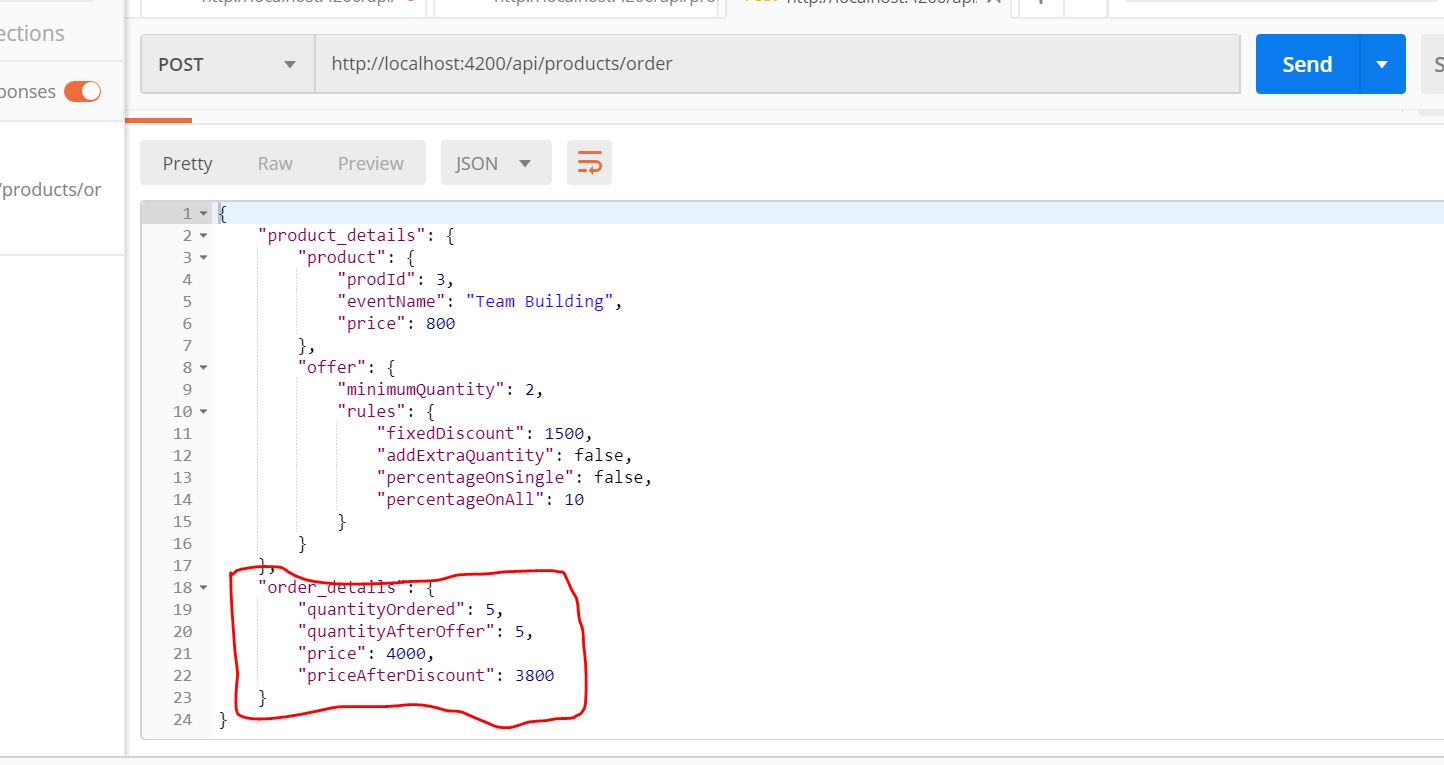
"percentOnSingle": false,

"percentOnAll": false

},

And posting request, calculation has resulted total price after discount 3,800

(Fixed discount on 4 items = 3000 + 1 item = 800 i.e. 3,800)



Similarly other rules can be tested by changing configuration in application.