


# Backend Technical challenge

## 1. Description

Welcome to the Restaurant Backend Challenge!

Imagine you're the developer making the online ordering system for a famous restaurant. Your job, if you accept it, is to make the system even better. The restaurant is known for its tasty pizzas, burgers, pastas, and more.

In this test, you are given a file (  menu.json) that represents the content of the menu, including the following models:

- Category :

Field	Type	Description
id	Number	Unique identifier for the category.
name	String	Name of the category.
products	Array	List of products in the category.

- Product :

Field	Type	Description
id	Number	Unique identifier for the product.
name	String	Name of the product.
price	Number	Price of the product.
happyHourPrice	Number	Price during happy hour for the product.
supplements	Array	List of available supplements for the product.

- Supplement :

Field	Type	Description
id	Number	Unique identifier for the supplement.

name	String	Name of the supplement.
price	Number	Price of the supplement.
happyHourPrice	Number	Price during happy hour for the supplement.

Your goal? Build a strong system using Node.js, Express, and TypeScript to handle online orders smoothly. Let's make ordering food easy and fun for our customers!

Good luck with the challenge!

## 2. Tasks:

- 💡 Create an endpoint to retrieve a list of all categories in the menu, showcasing each category's unique ID along with its name.
- 💡 Implement an endpoint to fetch information about a specific category using its ID. Provide details about the chosen category, including its associated products and their supplements. Make sure to handle any errors with clear and easy-to-understand messages.
- 💡 Build a quick search tool for customers that directly looks for a product by its name. Create an endpoint that searches for products with a specific keyword in their names. The response should be a list (array) of matching products
- 💡 The restaurant manager wants to take some supplements off the menu. Create an endpoint to make this happen. The endpoint should update the product details and return the modified product without the removed supplements. Also, make sure to handle cases where the specified supplement ID doesn't exist in the product's supplements list.
- 💡 A customer is ordering through our online system, sending a request like this:

```
{
  "products": [
    {
      "id": 1555,
      "supplements": [3000, 5000, 400]
    }
  ],
  "happyHoursActive": true
}
```

Your job is to create an endpoint for handling this order. The endpoint should:

- Check that the request has the right structure and necessary fields.
- Confirm that the provided product and supplement IDs match valid items on the menu
- Calculate the total cost of the order, factoring in product and supplement prices
- If it's a happy hour, apply happy hour prices for products and supplements
- Generate a response with the total cost before and after the discount, indicating if a discount was applied, and provide detailed information about the ordered items

Expected Response:

```
{
  "totalCostBeforeDiscount": 39.25,
  "totalCostAfterDiscount": 35.50,
  "discountApplied": true,
  "orderDetails": [
    {
      "product": "Pizza Margherita",
      "price": 10.99,
      "happyHourPrice": 9.89,
      "supplements": [
        {"id": 3000, "name": "Extra Cheese", "price": 1.50, "happyHourPrice": 1.35},
        {"id": 5000, "name": "Pepperoni", "price": 2.00, "happyHourPrice": 1.80},
        {"id": 400, "name": "Olives", "price": 0.75, "happyHourPrice": 0.68}
      ]
    }
  ]
  // ... additional product details if multiple products are ordered
}
```