🚲

Factorial Tech Challenge

Bicycle Shop

You're tasked with building a website that allows Marcus, a bicycle shop owner, to sell his bicycles.

Marcus owns a growing business and now wants to sell via the internet. He also tells you that bicycles are his main product, but if the business continues to grow, he will surely start selling other sports-related items such as skis, surfboards, roller skates, etc. It would be a nice bonus if the same website allowed him to sell those things as well.

What makes Marcus's business successful is that customers can completely customize their bicycles. They can select many different options for the various parts of the bicycle. Here is an incomplete list of all the parts and their possible choices, to give an example:

- Frame type: Full-suspension, diamond, step-through
- Frame finish: Matte, shiny
- Wheels: Road wheels, mountain wheels, fat bike wheels
- Rim color: Red, black, blue
- Chain: Single-speed chain, 8-speed chain

On top of that, Marcus points out that there are some combinations that are prohibited because they are not possible in reality. For example:

- If you select "mountain wheels", then the only frame available is the full suspension.

- If you select "fat bike wheels", then the red rim color is unavailable because the manufacturer doesn't provide it.

Also, sometimes Marcus doesn't have all the possible variations of each part in stock, so he also wants to be able to mark them as "temporarily out of stock" to avoid incoming orders that he would not be able to fulfill.

Finally, Marcus tells you how to calculate the price that you should present to the customer after the customization of the bicycle. Normally, this price is calculated by adding up the individual prices of each part that you selected. For example:

- Full suspension = 130 EUR
- Shiny frame = 30 EUR
- Road wheels = 80 EUR
- Rim color blue = 20 EUR
- Chain: Single-speed chain = 43 EUR

Total price: 130 + 30 + 80 + 20 + 43 = 303 EUR

However, the price of some options might depend on others. For instance, the frame finish is applied at the end over the whole bicycle, so the more area to cover, the more expensive it gets. Because of that, the matte finish over a full-suspension frame costs 50 EUR, while applied over a diamond frame it costs 35 EUR.

These kinds of variations can always happen and they might depend on any of the other choices (not only two), so Marcus asks you to consider this because otherwise, he would be losing money.

This code exercise consists of defining a software architecture that could satisfy the requirements described above. In particular:

- **Data model**: What data model would be best to support this application? Could you describe it? Include table specifications (or documents if it's a non-relational database) with fields, their associations, and the meaning of each entity.

- **The description of the main user actions in this e-commerce website**. Explain how they would work in detail.
    - **The product page**: This would be a read operation, performed when you need to display the page of a product (a bicycle) for the customer to purchase. How would you present this UI? How would you calculate which options are available or not? How would you calculate the price depending on the selections made by the customer?
    - **The "add to cart" action**: Following the previous point, the product page should have a button to "add to cart" after the customer has made some specific selection. What happens when the customer clicks this button? What is persisted in the database?
- **The description of the main workflows from the administration part of the website, where Marcus configures the store**.
    - **The creation of a new product**: What information is required to create a new product? What changes in the database after performing this action?
    - **The addition of a new part choice**: How can Marcus introduce a new rim color, for example? Can you describe the UI? What changes in the database after this action?
    - **Setting up prices**: How can Marcus change the price of a specific part (like the diamond frame type) or specify that some combinations of choices have particular prices? How does the UI look? How does the database change to store this information?

We expect you to provide the main core model of the solution: a set of classes/functions/modules in the language of your choice that can describe the main relationships between entities and any supporting materials you find useful (database schemas, diagrams etc). Please make it as lightweight as possible: no need to use web frameworks or provide the finished solution, the goal is to see how do you model and code the domain logic.

For any other specification of the system that's not directly stated in the exercise, feel free to interpret it as you see best.