SkilloVilla

Our Aim
We envision a world where anyone, anywhere has the power to transform their lives through learning. We train youth to be relevant in the digital economy and also assist them to secure their dream job.

Assignment
A user comes to our platform to check out the prices of our course offerings. He should be able to see the prices on different pages (e.g., the course product page, checkout page, etc.) in a simple and transparent manner.

Your assignment is to design an API that provides pricing information for a course.

The net price for a course is composed of several components: base price, taxes, and other components. The type of these components and their values might differ in different scenarios (For e.g. tax component value can be different for different countries; conversion fees might be applicable if payment is done in USD assuming we consider INR as the base unit etc.).

Each course can also have a different pricing strategy. It can be a free, one-time payment strategy, or a subscription-based model (subscribe for x months), etc. It should be possible to pick and choose such strategies on a course-by-course basis.

- ❑ Database usage is not a P0 requirement. You can choose to simplify it to expedite development time. For eg, in-memory data structures can be used as a datastore. Do however ensure that this layer is optimized for the use case.
- ❑ There is no need to write data onboarding flows. Assume the data is already present in the datastore.
- ❑ It has to be a **Java Spring Boot** app
- ❑ Add a README file that states your
  - ❑ high-level assumptions or any important assumptions
  - ❑ steps or commands on setting up the datastore with some sample data
  - ❑ building and running the code
  - ❑ curl commands to test API
- ❑ Be concise and to the point in the README file but ensure to add the necessary details
- ❑ Add comments in the code wherever you are making any assumptions

The assignment will be judged on the following criteria:

- Code should be functional and should produce a proper outcome

- You will be evaluated primarily on the code-quality, your component breakdown, and the quality of your workflow
- Code should be modular and readable
- Code should easily accommodate new requirements and minimal changes.
- Bonus points for test cases

Submission details

- Submissions should be in **zip** format. Send your submissions before the deadline along with your full name.
  NOTE: Submission in any other format will be directly rejected
- Inside the zip, ensure that the README doc describes how to set up and run the app you've built in a fresh environment
  (assume that the environment on which this will run doesn't have any pre-installed packages, scripts, or dependencies, so add the steps accordingly)