# JoulesLabs Assessment Test

## Title: E-Commerce Cart & Coupon Service

### Overview

We want you to design and implement a simple **Cart and Coupon Service**.
The service should allow customers to manage their cart, apply coupons, and calculate the total amount after discounts.

Your solution should cover the following requirements.

---

### Requirements

#### Cart

- A customer can add items to the cart.
- A customer can update or remove items from the cart.
- The cart should display:
    - Total price before discount
    - Applied discount amount
    - Final payable amount

#### Coupons

There will be **two types of coupons**:

1. **General Coupon**
    - The customer manually enters the coupon code.
    - If valid, the discount is applied to the cart.
2. **Auto-Applied Coupon**
    - This coupon should be applied automatically.

- The rules depend on conditions like the total cart price or the products in the cart.

---

## Discount Types

Coupons can have two discount types:

1. **Fixed amount** (e.g., $100 off)
2. **Percentage** (e.g., 10% off)

---

## Coupon Rules

Each coupon can have the following rules/attributes:

- Start time and expiry time.
- Maximum discount amount.
- Discount type (fixed or percentage).
- Minimum cart size (number of items).
- Minimum total price required.
- Product-specific restrictions (coupon applies only to certain products).
- Whether the coupon is auto-applied or not.
- Maximum number of total uses (system-wide).
- Maximum number of uses per user.

---

## Task

- Design the data structures/models needed to support the cart and coupon functionality.
- Implement the logic to:
  - Add/update/remove items in the cart.
  - Apply coupons (both manual and auto-applied).

    - Calculate total price, discount, and final payable amount.
- Show how coupon validation works based on the rules above.
- Write unit tests to demonstrate your implementation.

---

## Bonus (Optional)

- Show how you would design this as a microservice with clear APIs (e.g., REST or GraphQL).
- Explain how you would handle concurrency if multiple users are applying the same coupon at the same time.

---

## Note

You can implement or add any additional features if you think they would be useful. We'll appreciate it.

---

## Acceptance Criteria

Your solution will be considered correct if it meets the following criteria:

- Cart supports add, update, and remove operations.
- Cart shows total price, discount, and final payable amount correctly.
- General coupons can be applied manually and validated properly.
- Auto-applied coupons trigger automatically when their rules are met.
- Coupon rules (expiry date, max uses, min cart size/price, etc.) are enforced.
- Both discount types (fixed and percentage) are supported.
- Code is clean, structured, and readable.
- Unit tests cover the main functionality.
- Bonus points: API design, scalability considerations, concurrency handling.

---

## What We Measure and Consider (Updated)

- **Correctness** – Meets requirements and acceptance criteria.
- **Code Quality** – Readable, maintainable, idiomatic.
- **Design Decisions** – Sound data models, clear boundaries, sensible APIs.
- **Critical Thinking & Problem-Solving** –
    - How you decompose the problem and prioritize.
    - Assumptions stated and justified (no guesswork hidden).
    - Options evaluated with trade-offs (complexity, performance, DX).
    - Clear reasoning for chosen approach vs. alternatives.
    - Handling ambiguity and edge cases proactively.
    - Debugging strategy and how you verify correctness (logs, tests, assertions).
- **Testing** – Meaningful unit tests with good coverage of rules/edge cases.
- **Error Handling** – Robust validation, graceful failures, clear messages.
- **Scalability & Extensibility** – Easy to add new rules/types; performance-aware.
- **Documentation** – Setup/run steps, design rationale, examples.
- **Developer Experience** – Clean repo, helpful scripts, sensible project structure.
- **Creativity/Extra Features** – Thoughtful extras beyond the brief.

Want me to drop this into the full brief and send the finalized one-pager version too?

---

→ Deliver your solution in a GitHub repository or as a zip file containing code and instructions to run it.
→ You can use any language or framework you're comfortable with.