Coupon Management

Assignment Name: Coupon Management *<< Please mention this name in Submission Form >>*
Role: Software Developer
Goal: Build a **Simple Coupon System** for an e-commerce use case..

---

1. Problem Overview

Build a **Simple Coupon System** for an e-commerce use case.

Your system should:

1. Provide an API to **create coupons** with **eligibility rules**.
2. Provide another API that, given a **user + cart input**, returns the **best matching coupon** for that user (or no coupon if nothing applies).

You can implement this as:

- A simple **HTTP service** (preferred), in any language/framework you like.

Persistence (database) is **optional**. In-memory storage or simple JSON is fine for this assignment.

---

2. Domain: E-commerce Coupon Concepts

2.1 Coupon Definition

Each coupon should have at least:

- `code` – unique coupon code (e.g. `"WELCOME100"`)
- `description` – short human-readable description
- `discountType` – `"FLAT"` or `"PERCENT"`
- `discountValue`
    - If `FLAT` → amount off (e.g. `100` means ₹100 off)
    - If `PERCENT` → percentage (e.g. `10` means 10% off)
- `maxDiscountAmount` (optional; relevant for `%` discounts)
- `startDate` and `endDate` (coupon valid only in this window)
- `usageLimitPerUser` (optional; e.g. `1` = can be used once per user)
- `eligibility` – object describing who/what this coupon applies to (see below)

You may extend this schema if needed, but document any change in the README.

2.2 Supported Eligibility Attributes

Design an `eligibility` object with the following **supported fields** (all are optional; if not provided, that condition is ignored):

User-based attributes

- `allowedUserTiers`: list of allowed user tiers
  e.g. `["NEW", "REGULAR", "GOLD"]`
- `minLifetimeSpend`: minimum total historical spend of the user (e.g. `5000`)
- `minOrdersPlaced`: minimum number of past completed orders (e.g. `3`)
- `firstOrderOnly`: boolean; if `true`, coupon is only valid if this is the user's **first order**
- `allowedCountries`: list of country codes (`["IN", "US"]`)

Cart-based attributes

- `minCartValue`: minimum total cart value before discount
- `applicableCategories`: list of product categories for which the coupon can apply (e.g. `["electronics", "fashion"]`; valid if **at least one** item in cart is from these categories)
- `excludedCategories`: categories that **must not** appear in the cart for the coupon to be valid
- `minItemsCount`: minimum number of total items in the cart (sum of quantities)

You do **not** need to implement a real order history/lifetime spend storage.
You can simulate user attributes as part of the input to the "best coupon" API.

---

3. Input Models

You will need at least two main inputs:

3.1 User Context

Example:

```
JSON
{
  "userId": "u123",
```

```
  "userTier": "NEW",          // e.g. NEW, REGULAR, GOLD
  "country": "IN",
  "lifetimeSpend": 1200,      // total spend so far
  "ordersPlaced": 2           // number of completed orders
}
```

3.2 Cart

Example:

```JSON
{
  "items": [
    {
      "productId": "p1",
      "category": "electronics",
      "unitPrice": 1500,
      "quantity": 1
    },
    {
      "productId": "p2",
      "category": "fashion",
      "unitPrice": 500,
      "quantity": 2
    }
  ]
}
```

You should be able to compute `cartValue` from the items.

---

4. APIs to Implement (Minimum)

You should implement at least **two APIs**.

4.1 Create Coupon API

Behavior:

- Store the coupon in memory (or in a simple store).

- `code` should be unique; you may decide how to handle duplicates (reject or overwrite) but document your choice.

  You may optionally add `GET /coupons` to list all coupons for debugging/testing.

4.2 Best Coupon API

Expected behavior:

1. Evaluate **all coupons** currently stored.
2. Filter coupons that are:
    1. Within validity dates (`startDate` ≤ now ≤ `endDate`).
    2. Not exceeding `usageLimitPerUser` for this user.
    3. Satisfying all eligibility criteria (user & cart attributes).
3. For each eligible coupon, **compute the discount amount**:
    1. For `FLAT`, discount = `discountValue`.
    2. For `PERCENT`, discount = `discountValue% of cartValue`, capped by `maxDiscountAmount` if provided.
4. Select the **best coupon** using a clear rule, for example:
    1. Highest discount amount
    2. If tie, earliest `endDate`
    3. If still tie, lexicographically smaller `code`
5. Return the **best coupon** and its computed discount, or `null` if none applies.

---

5. Non-Functional Requirements

We care about:

- **Code clarity & structure** (functions/classes broken down logically, good naming)
- **Clear eligibility evaluation logic**
- **Deterministic “best coupon” selection**
- **Good error handling** for invalid inputs

You **do not** need:

- Authentication
- UI
- Real database (unless you want to)

---

7. README & GitHub Requirements (Important)

Your GitHub repository **must** include a `README.md` with:

1. **Project Overview**
   2–5 lines describing what you built.
2. **Tech Stack**
   Language + framework + any important libraries.
3. **How to Run**
   - Prerequisites (e.g. `Node.js 18`, `Python 3.10`, `Java 17`, etc.)
   - Setup steps (e.g. `npm install`, `pip install -r requirements.txt`)
   - Commands to start the service (e.g. `npm run start`, `python main.py`).
4. **How to Run Tests** (if you add tests)
   - Commands like `npm test`, `pytest`, etc.
5. **AI Usage Note**
   - Briefly mention how you used AI tools, if at all.
   - Share the prompts used

---

8. Evaluation

We will look at:

- **Correctness & edge cases**
- **Clarity of code & structure**
- **Quality of eligibility logic & best-coupon selection**
- **Documentation & ease of running the project**
- **Bonus:** Tests, clean design, and thoughtful handling of trade-offs

---

🔑 Demo Login (Must Be Hard-Coded in Seed Data)

Email	hire-me@anshumat.org
Password	HireMe@2025!

This user **must exist** in your hosted deployment, so reviewers can log in **without registration**.

🏁 Submission Format

Submit this as a comment in the assignment form:

Form Link: forms.gle/Bjmec4ajeJtc7697A

```
XML
Name:
GitHub Repo:
Live Demo Link:
Tech Stack Used:
Notes for Reviewer (optional):
```
