WooCommerce Logic Skills Assignment (Logic-Focused)

Goal: Evaluate problem-solving, code quality, and understanding of WordPress/WooCommerce internals by building a small but non-trivial feature as a standalone plugin and submitting it on GitHub.

The Task: Cart Rules Engine (Single Feature)

Build a WordPress plugin named `wbcom-cart-rules` that adds a *rule-based discount engine* to WooCommerce. The engine should apply dynamic cart adjustments based on configurable logic. Implement **exactly these three rule types** to keep scope crisp:

1. **Tiered Quantity Discount**
   If a customer buys ≥ X items from a selected product category, apply Y% discount to those items only.
2. Example: Category **"T-Shirts"**, thresholds: `5 → 5%`, `10 → 10%`, `20 → 15%`.

3. **Spend Threshold Reward**
   If cart subtotal (before coupons/taxes/shipping) ≥ A, add a free product B to the cart automatically (only one) and show a notice.

4. If product B is out of stock or already in cart, show an informative message.

5. **First-Time Customer Offer**
   If the shopper has no completed orders with this site, apply a one-time ₹/₹% discount (configurable) to the cart.

6. Should never stack on subsequent orders for the same user/email.

   All three rules must be **independently toggleable** and **composable** (i.e., can be active together), and the final discount must be consistent/deterministic.

Functional Requirements

- Provide an **Admin UI** under **WooCommerce → Cart Rules** to configure:
- Enable/disable each rule.
- Category and tier thresholds for Rule 1 (multiple tiers supported, editable UI).
- Subtotal threshold and free product selector for Rule 2.
- Discount type (fixed/percent) and amount for Rule 3.
- Show a clear **cart/checkout line item** (e.g., `Cart Rules Discount`) with a breakdown (which rules contributed how much) via a tooltip or expandable note.
- Display **customer-facing notices** when a rule becomes active/inactive while editing the cart (AJAX refresh compatible).
- Prevent double benefits with coupons that provide the same product/discount (explain logic in comments/doc).

Technical Requirements

- **Architecture:** OOP, namespaced (`WBCOM\CartRules`), single responsibility classes (e.g., `Rules\TieredQuantity`, `Rules\SpendThreshold`, `Rules\FirstTimeCustomer`, `Admin\SettingsPage`, `Services\CartAdjuster`).
- **Hooks:** Use appropriate WooCommerce hooks/filters such as `woocommerce_cart_calculate_fees`, `woocommerce_before_calculate_totals`, `woocommerce_add_to_cart_validation`, `woocommerce_cart_item_name`, and WordPress settings APIs.
- **Data & State:** Store settings via the Settings API (no custom DB tables). Cache reads where sensible. Idempotent calculations on cart refresh.
- **Security:** Nonces for admin forms; sanitize & escape; capability checks (`manage_woocommerce`).
- **Performance:** Avoid N+1 queries; prefer batch product/category queries; minimal repeated cart iteration.
- **Compatibility:** WooCommerce 8+ and WordPress 6.5+. PHP 8.1+. Works without other plugins.
- **i18n:** Text domain `wbcom-cart-rules` with translation functions.

Testing Requirements

- **Automated tests** using **PHPUnit** (WP & WC test bootstrap). Provide:
- Unit tests for each rule's core decision logic (input → expected discount).
- An integration-style test that simulates a cart with mixed categories and verifies combined results & messages.
- Include a minimal **GitHub Action** workflow that runs PHPUnit on push/PR for PHP 8.1 and 8.2.

UX Requirements

- Settings page: clean, accessible, grouped sections, helpful inline descriptions/examples.
- Frontend: concise notices (e.g., "Spend ₹500 more to get a free Mug"). Update on cart fragments refresh.

Deliverables (GitHub)

Create a **public GitHub repository** named `wbcom-cart-rules` containing:

```
wbcom-cart-rules/
├─ wbcom-cart-rules.php           (main plugin bootstrap)
├─ src/
│  ├─ Admin/SettingsPage.php
│  ├─ Rules/TieredQuantity.php
│  ├─ Rules/SpendThreshold.php
│  ├─ Rules/FirstTimeCustomer.php
│  ├─ Services/CartAdjuster.php
```

```
│  └─ Support/Helpers.php
├─ tests/
│  ├─ bootstrap.php
│  ├─ unit/RuleTieredQuantityTest.php
│  ├─ unit/RuleSpendThresholdTest.php
│  ├─ unit/RuleFirstTimeCustomerTest.php
│  └─ integration/CartCompositionTest.php
├─ languages/wbcom-cart-rules.pot
├─ readme.md                        (install, config, decisions, trade-offs)
├─ CHANGELOG.md
├─ .github/workflows/phpunit.yml
├─ composer.json                    (autoload: psr-4 WBCOM\\CartRules\\ =>
src/)
└─ phpcs.xml                        (WordPress coding standards)
```

Readme must include: - Setup steps (Composer autoload, `wp-env` or WP-CLI/WP local stack, WooCommerce install). - Sample configuration for quick review (e.g., JSON or screenshot of settings). - Explanation of calculation order, conflict handling, and any edge cases considered.

Acceptance Criteria (What we will test)

1. Plugin activates without fatal errors or notices.
2. Each rule works independently with correct math and messages.
3. When combined, rules yield a deterministic, explained total (document the order of application; e.g., percent discounts after item-level adjustments, then free item insertion).
4. Admin settings persist; capability checks enforced; forms protected by nonces.
5. PHPUnit tests pass locally and in CI.
6. Code style passes `phpcs` with WordPress rules (allowing reasonable exclusions documented in `phpcs.xml`).

Evaluation Rubric (100 pts)

- **Correctness & Logic (30):** Rule behavior, edge cases, deterministic composition.
- **Code Quality (20):** OOP design, readability, SOLID, namespacing.
- **WooCommerce Mastery (15):** Correct hooks/filters, cart/fees usage, fragment updates.
- **Security & Reliability (10):** Nonces, sanitization/escaping, capability checks.
- **Performance (5):** Efficient queries & cart iteration.
- **Testing (10):** Coverage of core logic + passing CI.
- **Docs & DevEx (5):** Readme clarity, setup, decisions.
- **Git Hygiene (5):** Commits, PR-ready structure.

Timebox & Submission

- **Estimated Timebox:** 5–7 hours total. (It's fine to push partial work; explain trade-offs in the readme.)

- **Submit:** GitHub repo URL + a short note summarizing your approach and any compromises.

Notes

- If you want to test only logic and structure (not endurance), you can **limit the task** to **2 rules (Tiered Discount + Spend Threshold)** and **skip CI setup**. This reduced version should take about **3–4 hours**.

Tips & Hints (non-mandatory)

- Prefer computing line-item adjustments via `woocommerce_before_calculate_totals` (for per-item) and cart fees via `woocommerce_cart_calculate_fees` (for cart-wide). Ensure tax compatibility by setting fee as taxable where appropriate.
- For first-time customer detection, consider both logged-in users and guest checkout emails (`wc_get_orders` with `billing_email`). Cache minimal results in the session for the request lifecycle.
- Use `WC()->cart->get_cart_contents_count()`, product category checks via terms, and subtotal from `WC()->cart->get_subtotal()` (pre-fee, pre-discount) — document your chosen definitions.

What to Submit Back to Us

1. **GitHub link** to the repo.
2. A **screenshot/GIF** of: settings page, a cart where rules trigger, and passing tests.
3. A short **approach note** (≤300 words) in the readme.

This assignment is designed to surface real-world reasoning (not just copy-paste). Keep the scope tight, focus on clean logic and clear explanations.