COMP 2136 - Software Quality Assurance

Assignment 1: Software Testing Techniques & Test Case Design

Testing the GBC-Cart Coupon Code & Discount Module

Submitted by: Ben Morrison - 101572409

Course: COMP 2136 - Software Quality Assurance
Instructor: Andrew Rudder
Due Date: Friday, October 17, 2025

George Brown College

Part 1: High-Level Test Plan (Strategy)

1. Objective

Ensure the **Coupon Code & Discount Module** for the GBC-Cart online checkout system meets all functional requirements and delivers a reliable user experience by validating coupon code formatting, discount calculations, minimum purchase enforcement, and user feedback for all scenarios.

2. Scope

2.1 In-Scope

- **Coupon Code Validation (REQ-02):** Format validation (alphanumeric, 4-8 characters, case-insensitive), rejection of invalid formats
- **Discount Application (REQ-03):** Percentage-based (`SAVE10` - 10% off) and fixed-amount (`5OFF` - $5.00 off) discounts
- **Minimum Purchase Enforcement (REQ-04):** `SAVE10` requires $50.00 minimum; `5OFF` has no minimum
- **User Feedback (REQ-05):** Success ("Discount Applied!") and error messages ("Invalid Coupon Code")
- **Business Rules (REQ-06):** One coupon per order, cart total cannot go below $0.00
- **User Interface (REQ-01):** Coupon code input field presence and visibility

2.2 Out-of-Scope

Payment processing, user authentication, shopping cart operations, shipping/tax calculations, order history, email notifications, mobile app functionality, performance/load testing, security testing, and browser compatibility testing.

3. Test Approach

Black-Box Testing (Primary Focus):

- **Equivalence Partitioning:** Divide input domains into valid/invalid classes; select representative test data for comprehensive coverage
- **Boundary Value Analysis:** Test edge cases (e.g., $50.00 vs. $49.99 for `SAVE10`, 4 vs. 3 character codes)
- Design 15 test cases covering positive, negative, and boundary scenarios with full requirement traceability

White-Box Testing:

- Achieve statement coverage by executing all statements in coupon validation and discount calculation logic
- Achieve branch coverage by testing all TRUE/FALSE paths in decision points (focus on `applyCoupon()` method)

Grey-Box Testing:

- Leverage database schema knowledge to test coupon data retrieval, expiration handling, and edge cases (NULL values, connection failures)

---

4. Success Criteria

1. **Test Execution:** 100% of test cases executed; all critical-path cases pass
2. **Requirement Coverage:** All requirements (REQ-01 through REQ-06) validated by at least two test cases
3. **Code Coverage:** Minimum 90% statement coverage, 100% branch coverage for critical decision points
4. **Defect Resolution:** Zero high-severity defects; medium-severity defects documented and triaged
5. **User Experience:** All messages display correctly; cart totals update accurately; system handles invalid inputs gracefully
6. **Stakeholder Sign-Off:** Product Owner and QA team approval for production deployment

Part 2: Black-Box Test Case Design

Equivalence Partitioning (EP) Analysis

EP Analysis Table

Input	**Equivalence Class**	**Valid/Invalid**	**Example(s)**
Coupon Code Format	Alphanumeric, 4-8 chars, any case	Valid	`SAVE10`, `5OFF`
Coupon Code Format	Too short (<4 chars)	Invalid	`ABC`, `12`
Coupon Code Format	Too long (>8 chars)	Invalid	`VERYLONGCODE`
Coupon Code Format	Special characters	Invalid	`SAVE@10`, `5-OFF`
Coupon Code Format	Empty/null	Invalid	`""`, `null`
Coupon Code Format	Only special chars/spaces	Invalid	`!@#$`, `"  "`
Coupon Code Format	Leading/trailing whitespace	Invalid	`SAVE10`, `5OFF`
Coupon Code Existence	Exists in database (`SAVE10`, `5OFF`)	Valid	`SAVE10`, `5OFF`
Coupon Code Existence	Does not exist	Invalid	`FAKE99`, `NOTREAL`
Cart Subtotal (`SAVE10`)	At least $50.00	Valid	$87.42, $123.67
Cart Subtotal (`SAVE10`)	Less than $50.00	Invalid	$49.99, $25.00
Cart Subtotal (`5OFF`)	Greater than $0.00	Valid	$12.34, $7.89
Cart Subtotal (`5OFF`)	Equal to $0.00	Invalid	$0.00
Cart Subtotal	Odd-cent values	Valid	$87.33, $123.67
Discount Impact	Leaves positive balance	Valid	Cart $87.42 with $8.74 discount
Discount Impact	Equals cart total	Valid	Cart $5.00 with $5.00 discount
Discount Impact	Exceeds cart total (capped at $0)	Valid	Cart $3.00 with $5.00 discount

Boundary Value Analysis (BVA)

Input	Boundary Value	Valid/Invalid	Expected Behavior
Coupon Code Length	3 chars (`SAV`)	Invalid	Show "Invalid Coupon Code"
Coupon Code Length	4 chars (`SAVE`)	Valid*	Process coupon if code exists
Coupon Code Length	8 chars (`SAVE1234`)	Valid*	Process coupon if code exists
Coupon Code Length	9 chars (`SAVE12345`)	Invalid	Show "Invalid Coupon Code"
Cart Subtotal (`SAVE10`)	$49.99	Invalid	Show "Invalid Coupon Code" (minimum not met)
Cart Subtotal (`SAVE10`)	$50.00	Valid	Apply 10% discount, new total: $45.00
Cart Subtotal (`SAVE10`)	$50.01	Valid	Apply 10% discount, new total: $45.01
Cart Subtotal (`5OFF`)	$0.00	Invalid	Show "Invalid Coupon Code" (no items in cart)
Cart Subtotal (`5OFF`)	$0.01	Valid	Apply $5.00 discount, total capped at $0.00
Cart Subtotal (`5OFF`)	$5.00	Valid	Apply $5.00 discount, new total: $0.00
Cart Subtotal (`5OFF`)	$5.01	Valid	Apply $5.00 discount, new total: $0.01
Discount Cap (`5OFF`)	Cart: $3.00, $5.00 off	Valid	New total capped at $0.00 (not negative)
Discount Cap (`5OFF`)	Cart: $12.34, $5.00 off	Valid	New total: $7.34

*Valid format, but will show "Invalid Coupon Code" if the code doesn't exist.

Test Cases

Test Case Table (15 test cases)

TC-ID	Req(s)	Description	Steps	Test Data	Expected Result
TC-001	03,04,05	Verify 10% percentage discount (`SAVE10`) correctly applies to cart containing $87.42 when minimum $50 threshold is met	1. Checkout 2. Subtotal $87.42 3. Apply `SAVE10`	$87.42, `SAVE10`	Discount applied, total $78.68

TC-ID	Req(s)	Description	Steps	Test Data	Expected Result
TC-002	03,04,05	`SAVE10` rejected below minimum	1. Checkout 2. Subtotal $49.99 3. Apply `SAVE10`	$49.99, `SAVE10`	"Invalid Coupon Code", total $49.99
TC-003	03,04,05	`SAVE10` applies at exact minimum	1. Checkout 2. Subtotal $50.00 3. Apply `SAVE10`	$50.00, `SAVE10`	Discount applied, total $45.00
TC-004	03,04,05	`SAVE10` applies just above minimum	1. Checkout 2. Subtotal $50.01 3. Apply `SAVE10`	$50.01, `SAVE10`	Discount applied, total $45.01
TC-005	03,04,05	Verify $5.00 fixed amount discount (`5OFF`) applies to cart subtotal of $12.34 with no minimum purchase requirement	1. Checkout 2. Subtotal $12.34 3. Apply `5OFF`	$12.34, `5OFF`	Discount applied, total $7.34
TC-006	03,06	`5OFF` caps total at $0.00 if discount > subtotal	1. Checkout 2. Subtotal $3.00 3. Apply `5OFF`	$3.00, `5OFF`	Discount applied, total $0.00
TC-007	03,06	`5OFF` reduces total to $0.00 when discount = subtotal	1. Checkout 2. Subtotal $5.00 3. Apply `5OFF`	$5.00, `5OFF`	Discount applied, total $0.00
TC-008	02,05	Coupon code 4 chars accepted (lower boundary)	1. Checkout 2. Subtotal $91.27 3. Apply `5OFF`	$91.27, `5OFF`	Discount applied, total $86.27
TC-009	02,05	Coupon code >8 chars rejected	1. Checkout 2. Subtotal $123.67 3. Apply `VERYLONGCODE`	$123.67, `VERYLONGCODE`	"Invalid Coupon Code", total $123.67

TC-ID	Req(s)	Description	Steps	Test Data	Expected Result
TC-010	02,05	Empty coupon code rejected	1. Checkout 2. Subtotal $87.33 3. Leave code empty	$87.33, (empty)	"Invalid Coupon Code", total $87.33
TC-011	02,03,05	Case-insensitive validation for `SAVE10`	1. Checkout 2. Subtotal $87.42 3. Apply `save10`	$87.42, `save10`	Discount applied, total $78.68
TC-012	05	Non-existent coupon code rejected	1. Checkout 2. Subtotal $87.42 3. Apply `FAKE99`	$87.42, `FAKE99`	"Invalid Coupon Code", total $87.42
TC-013	01,05	Coupon input field present and visible	1. Checkout 2. Observe input field	N/A	Input field visible and enabled
TC-014	03,04	Verify `SAVE10` percentage discount calculation accuracy on high-value cart ($387.50) to ensure proper decimal precision	1. Checkout 2. Subtotal $387.50 3. Apply `SAVE10`	$387.50, `SAVE10`	Discount applied, total $348.75
TC-015	02,03,06	Verify only one coupon can be applied per order (REQ-06)	1. Checkout 2. Subtotal $87.42 3. Apply `SAVE10` 4. Attempt to apply `50FF`	$87.42, `SAVE10`, `50FF`	System rejects second coupon or replaces first coupon. Appropriate error shown.

Coverage Summary

- **Positive scenarios:** TC-001, TC-003, TC-004, TC-005, TC-007, TC-008, TC-011, TC-013, TC-014
- **Negative scenarios:** TC-002, TC-009, TC-010, TC-012, TC-015
- **Boundary tests:** TC-003, TC-004, TC-006, TC-007, TC-008, TC-009
- **All requirements covered:** REQ-01 through REQ-06
- **REQ-06 ("one coupon per order") explicitly tested in TC-015**

Part 3: White-Box and Grey-Box Scenarios

---

1. White-Box Testing (Statement & Branch Coverage)

100% Statement Coverage

The `applyCoupon()` method has 9 statements. To achieve **100% statement coverage**, we need test cases that execute every statement at least once:

Test Case	Covers	Input
TC-012 (Non-existent coupon)	Statements 1, 2	Code=`FAKE99`, Subtotal=$87.42
TC-002 (Below minimum)	Statements 1, 3	Code=`SAVE10`, Subtotal=$49.99
TC-001 (Percentage discount)	Statements 1, 4, 5, 7, 8, 9	Code=`SAVE10`, Subtotal=$87.42
TC-005 (Fixed discount)	Statements 1, 4, 6, 7, 8, 9	Code=`5OFF`, Subtotal=$12.34

Result: These 4 test cases execute all 9 statements, achieving **100% statement coverage**.

100% Branch/Decision Coverage

Three branches exist: `if (couponData == null)`, `if (subtotal < minimumPurchase)`, `if (type == PERCENTAGE)`. To achieve **100% branch coverage**, test both TRUE/FALSE paths:

Branch	Path	Test Case	Input
Branch 1	TRUE	TC-012	Code=`FAKE99`, Subtotal=$87.42
Branch 1	FALSE	TC-001	Code=`SAVE10`, Subtotal=$87.42
Branch 2	TRUE	TC-002	Code=`SAVE10`, Subtotal=$49.99
Branch 2	FALSE	TC-003	Code=`SAVE10`, Subtotal=$50.00
Branch 3	TRUE	TC-001	Code=`SAVE10`, Subtotal=$87.42
Branch 3	FALSE	TC-005	Code=`5OFF`, Subtotal=$12.34
Math.max(0)	Edge case	TC-006	Code=`5OFF`, Subtotal=$3.00

Result: These 7 test cases cover all branch paths, achieving **100% branch coverage**.

---

2. Grey-Box Testing Scenario

Expired Coupon Testing

Database Knowledge: Coupons table has columns: `code`, `type`, `value`, `min_purchase`, `expiry_date` (DATETIME).

Scenario: Test expired coupon validation using database schema knowledge.

Grey-Box Advantage:

- **Black-Box:** Limited to testing "expired coupons" without insight into how expiration is stored or validated
- **Grey-Box:** Insert test coupons with specific expiry dates, verify SQL queries include expiry validation, test time boundaries (e.g., 11:59:59 PM on expiry date)

Test Case TC-GB-001:

- **Description:** Verify system rejects coupon expired yesterday
- **Prerequisites:** `INSERT INTO Coupons VALUES ('EXPIRED10', 'PERCENTAGE', 0.10, 50.00, '2025-10-13 23:59:59')`
- **Test Data:** Code=`EXPIRED10`, Subtotal=$87.42, Current Date=2025-10-14
- **Expected Result:** "Invalid Coupon Code" displayed, total remains $87.42
- **Verification:** Query database confirms `expiry_date < CURRENT_TIMESTAMP` in SQL

End of Document