TEST ANALYSIS DOCUMENTATION

Project: POS System
Team Member: Abdulaziz Bezan

---

1. EXECUTIVE SUMMARY

Analysis of three methods from CartUI.java for BVT, Decision Table, and Code Coverage testing.

Methods: calculateTotal(), addToCartBtnActionPerformed(), create_invoiceActionPerformed()

---

2. METHOD 1: calculateTotal()

Purpose: Sums up cart for totel

2.1 Boundary Value Testing

Test Case ID	Input (cartTable state)	Expected Output	Test Type	Reason
BVT-CALC-1	0 rows (empty table)	totalofcart = "0.0"	Boundary	Minimum row count
BVT-CALC-2	1 row, value="10.00"	totalofcart = "10.00"	Boundary	Single item calculation
BVT-CALC-3	1 row, value="0.01"	totalofcart = "0.01"	Boundary	Minimum price value
BVT-CALC-4	1 row, value="999999.99"	totalofcart = "999999.99"	Robustness	Maximum practical price

Test Case ID	Input (cartTable state)	Expected Output	Test Type	Reason
BVT-CALC-5	2 rows: "100.50" + "200.75"	totalofcart = "301.25"	Typical	Multiple item sum
BVT-CALC-6	Multiple rows with decimals	Correct sum with 2 decimal places	Accuracy	Decimal precision
BVT-CALC-7	Value with 3+ decimals: "10.555"	Rounded/truncated appropriately	Robustness	Decimal formatting

- This BVT tests the implicit boundaries of calculateTotal() by checking cart states (empty, single item, multiple items) and price values (minimum, maximum, decimals). The method works for basic cases, but lacks clear input parameters with defined ranges, making proper boundary testing difficult. Improvements would include adding explicit parameters with validation for better testability!

2.2 Decision Table Testing

Condition	Rule 1	Rule 2	Rule 3
C1: Cart has items?	Yes	Yes	No
C2: All values numeric?	Yes	No	N/A
Action: Calculate sum	✓		
Action: Handle error		✓	

Condition	Rule 1	Rule 2	Rule 3
Action: Display 0.0			✓

- This decision table shows the method's logic paths: normal calculation when cart has numeric items, error handling for non-numeric data, and zero display for empty carts. The table reveals missing error recovery, non numeric values cause crashes instead of graceful handling. The logic is simple, but brittle without input validation.

  (note, the final submission will provide the input user test values)

2.3 Code Coverage Analysis

Coverage Type	Elements to Cover	Test Cases Needed	Status
Statement	6 code statements	Empty cart, 1 item, multiple items	Partial
Branch	4 decision points	Loop (0/1+ times), parse (success/fail)	Incomplete
Condition	4 boolean conditions	rowCount >0 (T/F), parsing (T/F)	Partial
MC/DC	3 independent effects	Row count → loop, parsing → total, total → UI	Achievable

- This table shows calculateTotal() needs tests for all loop and parsing scenarios. Missing error handling prevents full branch coverage. MC/DC is achievable by testing each condition's independent effect on the output.

---

3. METHOD 2: addToCartBtnActionPerformed()

Purpose: Adds product to cart.

3.1 Boundary Value Testing

Test Case ID	Input State	Expected Output	Test Type	Reason
BVT-ADD-1	No product selected (selectedRowIndex = -1)	Error message appears	Boundary	Invalid selection
BVT-ADD-2	Product selected, quantity = "1"	Item added to cart	Boundary	Minimum quantity
BVT-ADD-3	Product selected, quantity = "0"	Error/not added	Boundary	Invalid quantity
BVT-ADD-4	Product selected, quantity = "999"	Item added	Robustness	Large quantity
BVT-ADD-5	Product selected, quantity = "1.5"	Error/not added	Invalid	Non-integer quantity
BVT-ADD-6	Product selected, quantity = "-5"	Error/not added	Invalid	Negative quantity
BVT-ADD-7	Product selected, quantity = "abc"	Error message	Invalid	Non-numeric input

- This BVT tests selection states (selected/not selected) and quantity boundaries (valid/integer/positive). The method handles basic cases, but lacks validation for decimal/negative quantities which may cause calculation errors. Input validation before parseInt() would improve robustness.

3.2 Decision Table Testing

3.3 Code Coverage Analysis

Statement Coverage: [ ]
Branch Coverage: [ ]
Condition Coverage: [ ]
MC/DC Coverage: [ ]

---

4. METHOD 3: create_invoiceActionPerformed()

Purpose: Generates invoice

4.1 Boundary Value Testing

4.2 Decision Table Testing

4.3 Code Coverage Analysis

Statement Coverage: [ ]
Branch Coverage: [ ]
Condition Coverage: [ ]
MC/DC Coverage: [ ]

---

5. TEST RESULTS SUMMARY