Action-Based Testing (ABT) Cheat Sheet

A Logical Approach to Software Testing

Core Principles

1. The Testing Paradox

- **The Problem:** Unstructured testing leads to complexity. The more you test without structure, the harder it is to maintain.
- **The Solution:** Organize tests into modular, reusable actions.

2. The Three Layers of Logic

Separate the “what” from the “how”.

Layer	Name	Purpose	Example
1	**High-Level Actions**	Business Goals	`purchase_product("Laptop")`
2	**Mid-Level Actions**	Functional Steps	`login()`, `add_to_cart()`
3	**Low-Level Actions**	UI Mechanics	`click("submit_btn")`, `type("user", "bob")`

3. Naming Conventions (The Language of Logic)

- **Rule: Verb-Noun**
- **Why:** Creates predictable, readable sentences.
- **Examples:**
  - ✅ `check_balance`
  - ✅ `create_invoice`

- ❌ `balance_check` (Noun-Verb)
- ❌ `invoice_new` (Noun-Adjective)

4. Test Modules (Architecture)

- **Business Objects:** Tests for specific entities (e.g., `Invoices`, `Customers`). Focus on CRUD.
- **Business Flows:** Tests for processes spanning multiple objects (e.g., `Order Fulfillment`).

5. Interface Definitions (The Rosetta Stone)

- **Purpose:** Decouple logical names from technical IDs.
- **Example:** `"Submit Button"` → `#btn_submit_v2`

Common Anti-Patterns (What NOT to do)

- **The "Enter, Enter, Click" Fallacy:** Recording every keystroke. *Fix: Use higher-level actions.*
- **The "Clueless" Test:** Wandering without a goal. *Fix: Define a clear purpose.*
- **The "Swiss Army Knife":** Actions doing too much. *Fix: Keep actions focused on one task.*

Generated for the "Ask Marilyn About Software Testing" Course