E-Commerce Checkout Flow (Until Order Placement)

Validate the complete journey of a user placing an order, up to the point of order submission, with making a real purchase using “Cash on delivery”.

Scope of Testing

1. **User Login/Session**
   a. Valid login with correct credentials.
   b. Invalid login (wrong password, empty fields).
   c. Session handling (stay logged in across cart and checkout).
2. **Product Discovery**
   a. Search for a product by exact name, partial name, and category.
   b. Verify product details (title, price, stock availability).
   c. Add item(s) to cart and confirm cart updates correctly.
3. **Shopping Cart**
   a. Increase/decrease product quantity.
   b. Remove item from cart.
   c. Validate subtotal and recalculated total.
4. **Checkout Flow**
   a. Proceed to checkout from the cart.
   b. Add/select shipping address.
   c. Apply valid and invalid coupon codes.
   d. Verify tax/shipping calculations.
5. **Payment Page (but stop before actual transaction)**
   a. Ensure payment methods are listed (card,direct bank transfer, COD).
6. **Edge Cases / Negative Testing**
   a. Try checking out without adding a product.
   b. Try checking out with an out-of-stock item.
   c. Try skipping address details.
   d. Refresh/reload during checkout steps.

Test Data:
URL:https://mridul-demo.acodez.ca/wp-admin/
Username:demo
Password:demo

Deliverables Expected

• **Test Plan** (what to test, scope, assumptions).
• **Test Cases** (positive + negative + edge cases).
• **Automation script using Cypress tool** (upto order placing).
• **Bug Report** (with severity & priority).
• **Suggestions** (UX, validations, performance).

Part 1 – API Testing (Functional + Automation)

Objective

Evaluate ability to design, execute, and automate API tests for core e-commerce features.

Mock API Base URL

Use the following free mock API for testing (no signup required):

https://fakestoreapi.com/

Endpoints to Test

Endpoint	Method	Description
`/products`	GET	Fetch all products
`/products/{id}`	GET	Fetch single product details
`/carts`	GET	Get all carts
`/carts`	POST	Create a new cart (mock checkout)
`/users`	GET	Fetch list of users

Tasks

1. **Create API test cases** for the above endpoints.
    - Include positive and negative cases.
    - Example: invalid product ID, missing fields, etc.
2. **Automate the tests** using:
    - **Postman/Newman**, or
    - **Python (requests + pytest)**, or

    - **Java (RestAssured)**

3. **Validate**:

    - Status codes (200, 201, 404)
    - Response time (< 1000 ms)
    - JSON schema (keys and data types)
    - Key business logic (e.g., product price > 0)

4. Generate an **HTML or JSON report** (Newman, pytest-html, etc.).

Deliverables for Part 1

1. Postman Collection / Code files
2. Test Report (HTML / PDF / Excel)
3. Screenshot or brief summary of results (passed/failed cases)

Part 2 – Performance Testing (Load Test)

Objective

Assess ability to design and execute a simple load test using a mock API.

Task

1.Use **Apache JMeter** or **k6** to load-test the following API:

`https://fakestoreapi.com/products`

2.Simulate **50 virtual users** making GET requests for products.

3.Run the test for **1 minute** with a 10-second ramp-up time.

4.Collect and analyze:

- Average response time

- 90th percentile response time
- Error rate
- Throughput (requests/sec)

5.Generate a **performance report** (JMeter HTML or k6 summary).

Deliverables for Part 2

1.JMeter or k6 test script file (`.jmx` or `.js`)
2.HTML or console report output
3.Short summary (1–2 paragraphs) explaining:

- What you observed
- Any performance issues
- Recommendations for improvement