WebStore Product Catalogue

Testing Deliverable - 1

Test Strategy Document

Document Classification : Internal

G10X INC.
7545 IRVINE CENTER DRIVE
IRVINE, CALIFORNIA,
92618 USA

WWW.G10X.COM

GXX INDIA PVT LTD.
WORLD TRADE CENTER,
INFOPARK, COCHIN,
KERALA, INDIA

G10X PVT LTD.
BERKELEY SQUARE HOUSE,
LONDON,
W1J 6BD, UK

G10X GMBH
ESCHBORN 65760
FRANKFURT,
GERMANY

G10X PTE LTD.
30 CECIL STREET,
PRUDENTIAL
TOWER SINGAPORE
049712,
SINGAPORE

G10X DMCC
GOLD TOWER, DMCC
DUBAI, UAE

DOCUMENT CONTROL

DOCUMENT HISTORY

Version	Date	Author	Description/Summary of changes
V 1	22-04-2025	Joel Johnson	Test Strategy Document for Product Catalogue Deliverable

APPROVED BY

Version	Issue Date	Name	Position	Approval Date

Contents

1. INTRODUCTION

This document outlines the testing strategy for the Product Catalogue component of the eCommerce platform. The backend is currently under development, and this strategy will focus primarily on API-level testing, followed by optional database verification. Testing will evolve with the addition of features.

2. OBJECTIVE

The primary objective of this test strategy is to ensure that the **Product Catalogue** module functions as expected in accordance with the high-level business and technical requirements. This module serves as a foundational component for the overall eCommerce experience, enabling customers to discover and explore available products across multiple categories such as agricultural goods, cooked food, and other items.

3. SCOPE

APIs related to the Product Catalogue (fetch categories, fetch product list, product details).

Associated database interactions (read operations).

Tools: Postman and Cypress (initially).

Out of Scope:

- End-to-End (E2E) UI testing
- Other modules (cart, checkout, etc.)

4. TESTING STRATEGY

The testing strategy for the Product Catalogue backend includes both **manual** and **automated** approaches to ensure comprehensive coverage, early defect detection, and maintainability. This dual approach provides flexibility during initial development and sets the stage for scalable, repeatable testing as the system evolves.

Manual Testing

Manual testing will serve as the initial validation layer during the early stages of backend development, particularly for verifying the database and API behaviours before automation frameworks are fully integrated.

- **Database Validation Using pgAdmin**
  - Perform basic CRUD validations directly on the PostgreSQL database.
  - Validate product and category data accuracy, integrity, and relational consistency.
- **Endpoint Validation with Postman**
  - Use Postman to manually test API endpoints for expected status codes and response structures.

- Validate authentication, parameterized queries, and different edge case scenarios.

Automated Testing

Automation focuses on ensuring the repeatability and accuracy of API behaviour over time. This includes both unit and integration-level testing, which are critical for backend microservices.

- **API Unit Test Automation**
  - Automate tests for key API endpoints to validate HTTP status codes, content types, and JSON structure.
- **Scenario-Based API Integration Tests**
  - Combine multiple API calls (e.g., create, read, update, delete) into a single test flow.
  - Validate logical relationships and expected transitions across these operations.
- **API and Database Synchronization Checks**
  - Implement test cases that verify API JSON responses against corresponding database query results.
  - Ensure consistency and correctness between API data exposure and backend records.

5. TEST DATA STRATEGY

Use mocked and seed data sets for common scenarios.

Include test data for:

- Multiple product categories and catalogues
- Different parameters for calling APIs, checking responses based on parameter, which will be stored in fixture files
- Edge cases (no products in category, unavailable items)
- Negative Test Data Fixtures

6. TOOLS & FRAMEWORKS

Type	Tool
API Testing	Postman
Automated API Testing	Cypress
API – Database Integration Tests	Cypress
Database Verification	pgAdmin

7. RISK AND MITIGATION

Risk	Mitigation Strategy
Limited DB access for validation	Request read-only access or coordinate with DB team.
Lack of API Understanding	Work closely with developers and update test cases iteratively.
Changing API specs during development	Use version control for test cases, enable fast updates.
Dependency on backend readiness	Establish early communication with the dev team and align test timelines accordingly.
Test data issues or inconsistencies	Setup test data fixture files that can be modified based on changes as well as changing requirements of the dev team.

8. EXIT CRITERIA

- All planned API test cases executed and passed
- Critical bugs closed
- All endpoints return valid and expected responses
- Basic DB checks match API responses