**Dukaan – Testing Strategy & Test Plan**

**1. Introduction**

The purpose of this document is to define the **testing approach and plan** for the *Dukaan* application, which enables customers to browse and compare products from nearby shops, shopkeepers to manage inventory, and delivery partners to handle order fulfillment.

This combined Testing Strategy & Plan outlines **objectives, scope, approach, tools, resources, schedule, responsibilities, risks, and deliverables** to ensure the application meets its functional and non-functional requirements.

**2. Objectives**

The objectives of testing are to:

1. Verify that all customer, shopkeeper, and delivery partner modules work as per specified requirements.
2. Detect defects at the earliest possible stage and ensure their timely resolution.
3. Validate critical workflows such as **cart management, checkout, order preparation, and delivery tracking**.
4. Ensure the system provides a consistent and user-friendly experience across browsers and devices.
5. Confirm system integration and proper flow of data among different user roles.
6. Demonstrate quality assurance practices in alignment with SDLC for academic evaluation.

**3. Scope**

**In-Scope**

* Functional testing of:
  + Customer features (browse shops, add to cart, checkout, compare prices, track orders).
  + Shopkeeper features (add/edit products, manage orders, check payments).
  + Delivery partner features (view orders, accept/reject, update status, profile).
* Authentication (login, logout, error handling).
* Usability and UI consistency across modules.
* Compatibility across major browsers (Chrome, Edge, Firefox) and mobile (Android).

**Out-of-Scope**

* Large-scale performance/load testing.
* Advanced penetration security testing.
* Real payment gateway integrations (simulated only).

**4. Testing Approach**

Testing will follow the **V-Model** with corresponding verification and validation phases.

**Levels of Testing**

1. **Unit Testing**
   * Performed by developers.
   * Verifies correctness of small components like “Add to Cart” button or product form validation.
2. **Integration Testing**
   * Ensures data flow across modules.
   * Example: When a customer places an order, it should appear in the shopkeeper’s dashboard and notify the delivery partner.
3. **System Testing**
   * End-to-end validation of customer → shopkeeper → delivery workflow.
   * Includes functional and UI testing.
4. **User Acceptance Testing (UAT)**
   * Performed by classmates acting as real users.
   * Confirms whether Dukaan meets user expectations and academic project goals.

**5. Test Deliverables**

* Test Strategy & Plan (this document)
* Detailed Test Cases (with ID, steps, input, expected output)
* Test Data (dummy product catalogs, sample users)
* Bug Reports (documented in GitHub Issues)
* Test Execution Reports
* Test Summary Report (after final testing)

**6. Test Environment**

| **Component** | **Details** |
| --- | --- |
| Frontend | HTML, CSS, JavaScript prototype (browser-based) |
| Backend (if simulated) | JSON-based dummy data, mock APIs |
| Database | Local JSON files / simulated ER model |
| Tools | Browser DevTools, GitHub Issues, Google Sheets for case tracking |
| Devices | Windows Laptop (Chrome, Edge, Firefox), Android mobile browser |

**7. Testing Tools**

* **GitHub Issues** → Bug tracking and task assignment
* **Excel** → Test case management
* **Browser Developer Tools** → Debugging, UI testing
* **Dummy Data JSON** → Controlled environment for product/shop/order testing

**8. Roles & Responsibilities**

| **Role** | **Responsibility** |
| --- | --- |
| Scrum Master | Ensure testing activities align with sprint goals |
| Developers | Perform unit tests, fix bugs |
| Testers (Team members) | Write and execute test cases, log defects |
| Product Owner | Validate UAT, approve final acceptance |

**9. Test Schedule**

| **Phase** | **Planned Dates** | **Activities** |
| --- | --- | --- |
| Test Planning | 10 – 15 Aug 2025 | Strategy, test case design |
| Unit Testing | 12 – 16 Aug 2025 | Developers test individual modules |
| Integration Testing | 16 – 18 Aug 2025 | Customer–Shopkeeper–Delivery data flow |
| System Testing | 18 – 20 Aug 2025 | End-to-end scenarios |
| UAT | 21 – 22 Aug 2025 | Peer testing and feedback |
| Final Test Summary | 23 Aug 2025 | Document results for demo |

**10. Risks & Mitigation**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Limited testing time | Medium | Prioritize critical workflows (login, order, checkout, delivery). |
| Device/browser differences | Medium | Test on multiple browsers + at least one Android device. |
| Missing features in prototype | High | Use stubs/dummy data to simulate workflows. |
| Inconsistent team contribution | Medium | Assign clear testing responsibilities. |

**11. Entry & Exit Criteria**

**Entry Criteria**

* Functional modules developed and available on local environment.
* Test data prepared.
* Test cases reviewed.

**Exit Criteria**

* All high-severity defects fixed and retested.
* Minimum 90% test cases executed successfully.
* All critical workflows verified end-to-end.
* UAT feedback incorporated.

**12. Reporting & Tracking**

* **Daily Updates** in team meetings.
* **Defects** tracked using GitHub Issues with labels (bug, priority, status).
* **Test Execution Results** logged in Excel.
* **Final Summary Report** presented before demo (24 Aug 2025).

**Bug Template**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Bug ID** | BUG-001 |
| **Title** | Short, clear title of the bug (e.g., *Cart total not updating after quantity change*) |
| **Module / Feature** | (e.g., Customer – Cart, Shopkeeper – Inventory, Delivery – Dashboard) |
| **Severity** | Critical / High / Medium / Low |
| **Priority** | High / Medium / Low |
| **Environment** | (Browser: Chrome 139 / Edge 128, OS: Windows 10 / Android 14, etc.) |
| **Preconditions** | Steps or conditions required before reproducing (e.g., user must be logged in, product added to cart) |
| **Steps to Reproduce** | 1. Step one 2. Step two 3. Step three |
| **Expected Result** | What should have happened |
| **Actual Result** | What actually happened |
| **Screenshots / Logs** | (Attach screenshot, console log, or error message if available) |
| **Status** | Open / In Progress / Fixed / Retested / Closed |
| **Reported By** | (Your Name / Tester Name) |
| **Reported Date** | (DD-MM-YYYY) |
| **Assigned To** | (Developer / Team Member Name) |