### 

### 

### 

### 

### **Test Plan Document for Craftyfy Handmade Craft E-Commerce Website**

**Created By:**Shana Yasmin S

**1. Introduction**

The **Craftyfy** is a full-stack web application designed for showcasing and selling handmade crafts. It features a dual-role login system for users (customers) and admin (store managers). This test plan outlines the approach for testing key functionalities such as user authentication, product management, cart functionality, order tracking, and payment integration.

The goal is to ensure the platform’s functionalities meet the specified requirements and ensure optimal performance and stability before deployment.

### **2. Objective**

The objectives of this test plan are to:

* **Validate functional correctness** for user registration, login, product management, cart operations, and order tracking.
* **Test the performance** of the application under different loads to ensure scalability and stability.
* **Verify data integrity** during user activities such as purchasing and adding products.
* **Ensure smooth user and admin experiences**, with secure, role-based access to features.
* **Report defects** and suggest improvements to enhance the application's functionality and usability.

### **3. Features to be Tested (In-Scope)**

#### **Functional Testing:**

* **User Registration & Login:**
  + Verify users can register, log in securely with valid credentials.
  + Check if error messages are displayed for invalid login attempts.
* **Product Management (Admin):**
  + Verify the admin can add, update, and delete products in the inventory.
  + Ensure that product changes reflect on the frontend after updates.
* **User Shopping Experience:**
  + Test the user’s ability to browse products by category.
  + Ensure items can be added to the cart, modified, and removed as needed.
* **Order Management:**
  + Verify users can track the order status.
  + Admin should be able to manage and update order statuses (e.g., processing, shipped, delivered).
* **Payment Integration:**
  + Ensure that users can securely complete transactions via Razorpay.
  + Validate payment gateway integration for successful purchases.
* **Admin Dashboard:**
  + Verify the admin can view the order list, manage users, and view detailed transaction data.

#### **Performance Testing:**

* **Load Testing:**
  + Ensure the platform can handle a large number of users browsing products, adding items to the cart, and completing purchases simultaneously.
* **Stress Testing:**
  + Test the system's behavior under extreme load conditions (e.g., thousands of users interacting at the same time).
* **Scalability Testing:**
  + Verify if the application scales effectively with an increasing number of users and orders.

### **4. Features Not to be Tested (Out-of-Scope)**

* **Advanced Security Penetration Testing**: Beyond role-based access control, no external security testing will be conducted.
* **Third-Party Integrations**: Any integrations outside the provided functionality (e.g., marketing platforms) are out of scope.
* **Mobile Testing**: The primary focus will be on the web-based platform. Mobile responsiveness will be tested, but not the functionality of mobile apps.

### **5. Test Levels and Test Types**

#### **Test Levels:**

* **System Testing:** Ensure that the complete application (frontend and backend) works as intended.
* **Acceptance Testing:** Confirm that the application meets the functional and non-functional business requirements.
* **Performance Testing:** Check the application’s behavior under varying loads to assess scalability.

#### **Test Types:**

* **Functional Testing:** Test the workflows of the system, including registration, login, product management, cart functionality, and order processing.
* **Performance Testing:** Ensure the system performs well under load (e.g., number of concurrent users, payment transactions).

### **6. Exit Criteria**

* **Test Execution:** At least 95% of test cases must pass, with no critical defects unresolved.
* **Bug Resolution:** Critical and high-severity bugs must be resolved and re-verified before deployment.
* **Requirement Compliance:** All key user stories and acceptance criteria must be met.

### **7. Suspension Criteria**

* **Critical Defects:** Defects that prevent core functionality (e.g., login failure, cart malfunction).
* **API Failures:** Server or API issues preventing testing progress.

### **8. Test Deliverables**

* **Test Plan Document**: This document (overview of the test strategy, objectives, scope, and approach).
* **Test Cases Document**: Detailed test steps for verifying each feature and functionality.
* **Automation Scripts**: Scripts for automated tests (if applicable).
* **Bug Reports**: Reports of identified issues along with steps for reproduction.

### **9. Test Environment**

* **Operating System:** Windows 10/11, macOS
* **Browsers:** Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions)
* **Network:** Stable internet connection (Wi-Fi, LAN, or Mobile Data)
* **Environments:** Staging Environment for testing before production

### **10. Estimation**

* **Test Planning & Documentation**: 1 week
* **Manual Test Case Execution**: 1 weeks
* **Automation Setup & Execution (if applicable)**: 1 week
* **Bug Reporting & Fix Verification**: Ongoing throughout the testing process

### **11. Roles and Responsibilities**

* **Manual Tester:** Execute test cases, log defects, and verify fixes.
* **Automation Tester (if applicable):** Develop and run automated tests for repetitive tasks like registration and login.
* **Test Manager:** Oversee the testing process, report status, and ensure all testing goals are met.
* **Developer (Backend & Frontend):** Address defects found during testing, assist with debugging, and support the testing team.

### **12. Risks and Mitigation Strategies**

| **Risk** | **Mitigation** |
| --- | --- |
| **Performance Issues under load** | Perform load and stress testing in a staging environment. |
| **Inconsistent data in cart after logout** | Ensure data persistence between sessions with cookies or local storage. |
| **Payment Gateway failures** | Perform multiple payment tests with different scenarios. |
| **Role-based access issues** | Test with multiple user roles (Admin and User) to validate access. |
| **Slow API responses under high load** | Use mock data where applicable for initial tests. |

### **13. Test References**

* **User Stories**: The functional features documented in the project scope (e.g., user login, admin dashboard).
* **API Documentation**: Backend API endpoints for managing products, orders, users, etc.

### **End of Document**