Marketplace Testing and Optimization Report

Prepared for: Marketplace Project

Prepared by: Your Name

Date: [Insert Date]

1. Introduction

This document provides a comprehensive report on the testing, error handling, backend integration, and performance optimization of the marketplace project. The report includes details of encountered issues, their solutions, testing methodologies, and recommendations for improvement.

2. Summary of Implemented Features and Fixes

A. Dynamic Product Listing Issues & Fixes

Issue: Product IDs were not dynamically generating, causing errors in routing.

Solution: Implemented Next.js dynamic routing for handling product pages correctly.

B. API Integration Issues & Fixes

Issue: API responses were inconsistent, leading to incorrect data mapping. **Solution:** Added error handling with try-catch blocks and implemented fallback UI elements for failed requests.

C. Performance Optimization Fixes

- Compressed and optimized images using **WebP format**.
- Implemented **lazy loading** for assets.
- Minimized CSS and JavaScript bundle sizes to enhance page speed.

D. Security Enhancements

- Used **HTTPS** for secure API communication.
- Implemented **input validation** to prevent SQL injection and XSS attacks.
- Secured sensitive API keys using environment variables.

3. Testing Methodologies & Tools Used

Test Type	Tool Used	Purpose
API Testing	Postman	Verify API responses and performance
UI Testing	Cypress	Ensure frontend UI behaves as expected
Performance Testing	Google Lighthouse	Identify speed and optimization issues
Security Testing	OWASP ZAP, Burp Suite	Check vulnerabilities and secure APIs

4. Key Testing Results & Findings

Below is a detailed report of encountered testing problems and their solutions:

A. Product Listing Issues

Problem: Dynamic product IDs were not being generated correctly, causing routing errors. **Solution:** Implemented Next.js dynamic routing to ensure products load based on unique IDs.

B. API Integration Challenges

Problem: API responses were inconsistent, leading to incorrect data mapping. **Solution:** Added structured error handling with try-catch blocks and implemented fallback UI elements for failed requests.

C. UI & Responsiveness Issues

Problem: Mobile layout was breaking on smaller screens. **Solution:** Adjusted CSS styles and implemented a mobile-first responsive design approach.

D. Security Vulnerabilities

Problem: API keys were being exposed in frontend code. **Solution:** Moved all sensitive credentials to environment variables using `.env.local`.

E. Performance Bottlenecks

Problem: Page load speed was slow due to unoptimized images. **Solution:** Compressed images and enabled lazy loading for assets.

Detailed Test Results

Test Case ID	Description	Expected Outcome	Actual Outcome	Status
TC-001	Product Listing Loads	Products should load correctly	Products loaded with minor delay	☑ Passed
TC-002	API Error Handling	Should display error message on API failure	Error message displayed correctly	☑ Passed
TC-003	Mobile Responsiveness	Layout should be optimized for mobile screens	So=[me layout issues fixed	⚠ Partial Fix

(For a complete list, refer to the attached CSV file)