Marketplace Staging Deployment and Testing

This repository contains the code and documentation for the **Marketplace Application**. The goal of this project is to prepare the marketplace for deployment by setting up a staging environment, conducting tests, and ensuring that everything is ready for production. This README summarizes the key activities, testing results, and deployment steps.

Table of Contents

Project Overview

The project focuses on preparing the **Marketplace Application** for deployment by:

- Setting up a **staging environment** to test the application in a production-like setting.
- Performing **functional**, **performance**, **security**, and **responsiveness** testing.
- Configuring the application's environment variables and deployment settings.
- Ensuring everything is secure, responsive, and fully functional before deploying to production.

Key Sections Explained:

- **Project Overview**: A brief description of what the project is and its purpose.
- **Deployment Steps**: Detailed instructions on how to deploy the marketplace, configure environment variables, and set up the hosting platform.
- **Testing**: Describes the testing types performed (Functional, Performance, Security, Responsiveness), and includes examples of test cases with results.
- **Folder Structure**: A visual representation of the project folder structure.
- **Test Case Reports**: Links to the documentation of test cases (e.g., CSV or PDF files).
- **Performance Report**: References the full performance report PDF.
- **Conclusion**: A summary of the project and its conclusion.

Deployment Steps

1. Choose Hosting Platform

For the staging environment, **Vercel** was chosen for its ease of use and automatic deployments.

2. Connect GitHub Repository

https://day-4-ruddy.vercel.app/

- Set up build settings in the hosting platform (e.g., selecting the framework and build commands).

3. Configure Environment Variables

- A `.env` file is used to store sensitive environment variables (API keys, credentials, etc.):

```
NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id
NEXT_PUBLIC_SANITY_DATASET=production
API_KEY=your_api_key
```

-These environment variables are securely uploaded to the hosting platform through the dashboard.

4. Deploy the Application

Deploy the application through the platform (e.g., Vercel automatically deploys the application on each GitHub push).
Ensure the build process completes successfully and verify the application works as expected.
Environment Configuration
Development (DEV)
The DEV environment is for local development where code is written and tested.
Staging (SIT/UAT)
The staging environment mimics the production environment and is used for final testing before deployment.
Production (PROD)
The PROD environment is the live, customer-facing version of the application.
Disaster Recovery (DR)
The DR environment is a backup that can be used in case of critical issues.
Testing
Functional Testing
Functional testing ensures that key features of the marketplace work as expected.

-Test Case 1: Validate product listing

Steps: Navigate to the product listing page and check if products are displayed correctly.
Expected Result: Products should be listed with their correct names, images, and prices.
Status: Passed
-Test Case 2: Test cart functionality

Steps: Add an item to the cart and verify that the cart updates accordingly.
Expected Result: Cart should show the added item correctly.

Status: Passed
Test Case 3: Test search functionality

Steps: Enter a search term in the search bar and verify the results.
Expected Result: Search results should show the correct products.
Status: Passed
Performance Testing
Performance testing was performed using Lighthouse to ensure the application is fast and responsive.

Lighthouse Performance Report:

Performance Score: 100/100
Accessibility Score: 100/100
SEO Score: 100/100
Best Practices Score: 100/100

Security Testing

Security testing ensured that the application is free from vulnerabilities:

HTTPS: The application is using HTTPS for secure communication.
Input Validation: Form inputs are sanitized to prevent SQL injection.
Sensitive Data: API keys and other sensitive information are properly secured.
Responsiveness Testing
Responsiveness testing ensures the application adapts to various screen sizes:

-Desktop: Layout adjusts properly.
-Tablet: Layout is responsive, and touch elements are usable.
-Mobile: The layout adapts correctly for smaller screens, and functionality is preserved.

Folder Structure

/documents
├── performance-report.pdf
├── test-cases.csv
/projects
├── components/
├── app/

/src: Contains the application code, including components, pages, styles, and utility functions.
/public: Stores assets such as images and static files.
/documents: Contains test case reports, performance reports, and other project documentation.
/tests: Contains test scripts and results for functional, performance, and security tests.
Test Case Reports
Test cases have been documented in the documents/test-cases.csv file. Here is an example:

Conclusion

This project successfully sets up a staging environment for the Marketplace Application. The application passed functional, performance, security, and responsiveness tests, and is now ready for final user acceptance testing (UAT) and production deployment.