Day6-DEPLOYMENT PREPARATION AND STAGING ENVIRONMENT SETUP

1. Setting Up and Configuring the Staging Environment

For Day 6, my primary task was to set up and configure a staging environment for my niche e-commerce marketplace. This involved several key steps:

Hosting Platform Selection

I chose **Vercel** as the hosting platform due to its seamless integration with **Next.js** and ease of deployment.

GitHub Repository Connection

- Connected my project's GitHub repository to Vercel to enable automatic deployments.
- Ensured proper repository permissions and branch selection for staging.

Build and Deployment Configuration

- Configured build settings to ensure smooth and automated staging builds.
- Defined the correct build commands and output directories.
- Enabled automatic deployments on push to the staging branch.

Secure Environment Variable Setup

- Added API keys and sensitive credentials securely within Vercel's environment variable settings.
- Ensured that no sensitive data is exposed in the codebase or logs.

Validation of Application Functionality

- Deployed the application to the staging environment to test production-like behavior.
- Checked database connectivity and UI responsiveness.
- Debugged and resolved minor pre-deployment issues related to API response times and UI rendering.

3. Staging Environment Testing and Documentation

I conducted several tests to ensure the staging environment was fully functional:

- **Performance Testing** – Verified load times and response times for key pages.

- **Functionality Testing** – Ensured checkout flow, authentication, and product fetching worked correctly.
- **API Testing** – Validated all API endpoints, including Stripe and ShipEngine integrations.
- **Error Handling Testing** – Simulated edge cases to check error handling.

Findings from the testing process were documented for future reference and issue resolution.

Conclusion

Successfully setting up and testing the staging environment has enhanced my understanding of deployment workflows. This setup ensures a smooth transition to production while minimizing potential deployment risks. The documentation serves as a comprehensive guide for future deployments and team collaboration.