Day 6

Introduction:

For the successful deployment of my online marketplace, I have chosen **Vercel** as the hosting platform for its seamless integration with modern web applications and its ability to quickly deploy and manage environments. I have also integrated **Sanity CMS** for content management to provide flexibility in handling product listings, categories, and other dynamic content. By securing sensitive data and configuring environment variables properly, I have ensured a robust and secure deployment pipeline for the application. This report outlines the key steps taken to set up the staging environment, configure environment variables, and conduct necessary testing to validate the deployment.

Key Areas of Focus

1. Deployment Strategy Planning

- **Hosting Platform**: I have chosen **Vercel** for its optimized deployment capabilities, integration with GitHub, and support for serverless functions, making it ideal for fast, scalable deployments.
    - **Why Vercel**: Vercel offers efficient deployment with zero configuration, automatic scaling, and automatic preview URLs for each Git commit, making it an excellent choice for staging environments.
- **Backend Services**:
    - **Sanity CMS Integration**: Sanity CMS is used to manage dynamic content like product details, categories, and pricing, allowing for easy content updates without needing code changes. The CMS is integrated with the frontend, ensuring a smooth content management experience.

2. Environment Variable Configuration

- **.env File Setup**:
    - I created a `.env` file to securely store sensitive environment variables such as API keys and database credentials. This file includes the following:
        - `NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id`
        - `NEXT_PUBLIC_SANITY_DATASET=production`

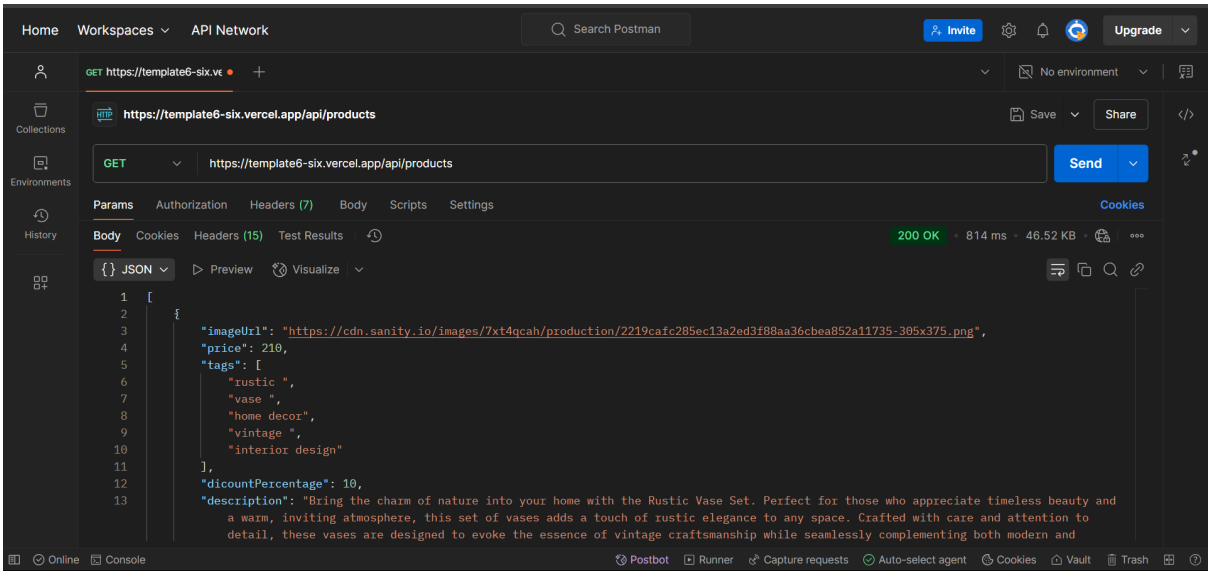
3. Staging Environment Setup

- **Deploy to Staging**:
    - The application was deployed to **Vercel's staging environment** by linking my GitHub repository to Vercel. The build was completed successfully.
- **Validate Deployment**:
    - The site was validated in the staging environment to ensure that the deployment was built correctly. I verified basic functionality, such as product listings, the shopping cart, and payment flow, ensuring that all features were working as expected in a production-like environment.

Prepared By: Mariyam Asif

4. Staging Environment Testing

- **Functional Testing**:
  - I conducted functional testing using **Postman** to validate API responses. The tests focused on verifying product listing APIs, cart APIs, and payment processing endpoints.
  - I tested critical features such as product search, adding products to the cart, and processing checkout transactions to ensure seamless user experiences.



- **Responsiveness and Error Handling**:
  - I manually verified the responsiveness of the platform on different devices and screen sizes, ensuring that the UI adjusts correctly.
  - Error handling was thoroughly tested to ensure that edge cases like API failures or missing content are handled gracefully, providing users with clear messages and a smooth experience.

5. Test Case Reporting

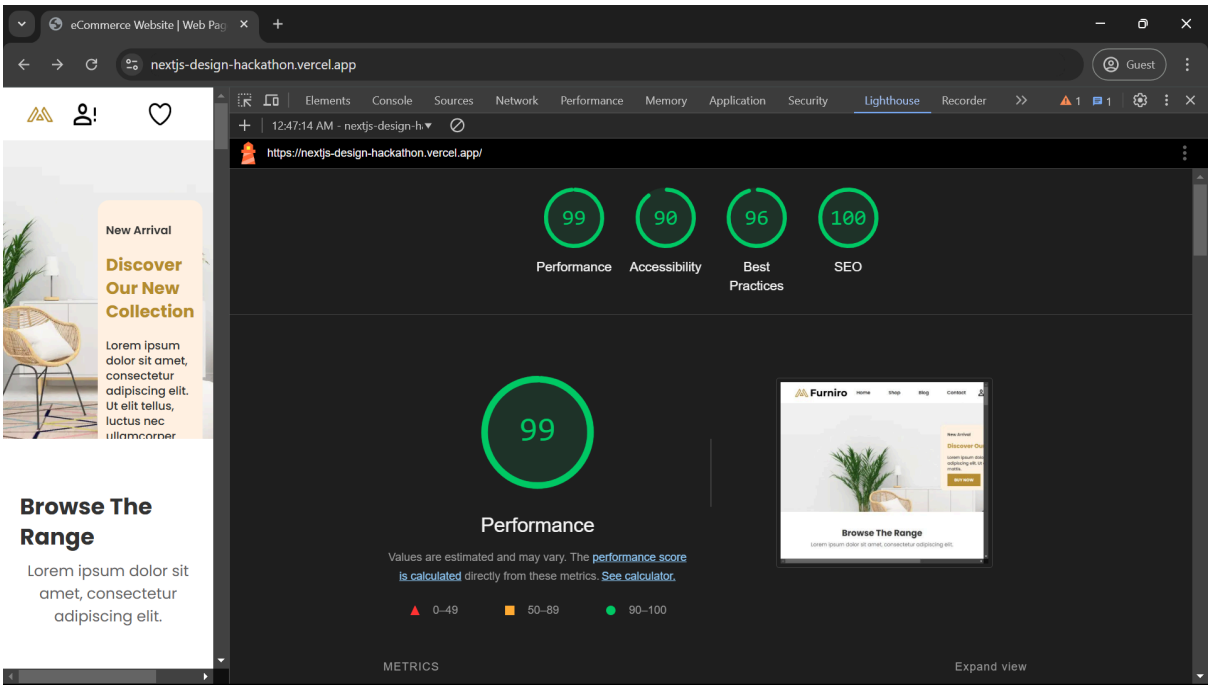
- **Test Case Reporting**:
  - I documented the test cases and results in a structured format, including steps, expected results, and actual results. This documentation will help track the performance and ensure all functionalities work as intended.

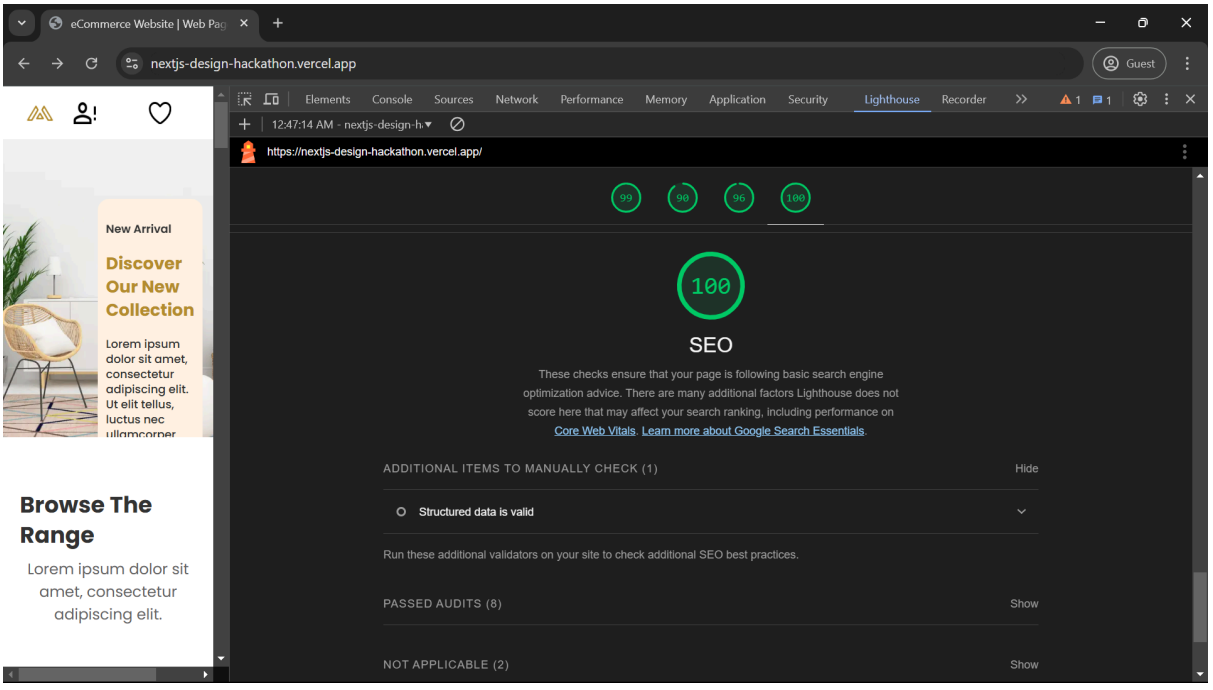
Test Case ID	Test Case Description	Test Steps	Expected Result
TC001	Validate product listing page	Open product page > Verify products	Products displayed correctly
TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message
TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product
TC004	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size
TC005	Test product search functionality	Enter a product name in search bar > Press search button	Relevant products should be displayed
TC006	Test API performance	Use Postman to test API endpoints	Get a successful response and check respon
TC007	Test fallback UI on empty cart	Add no items to cart > Open cart page	Display "No items in cart" message
TC008	Test cross-browser functionality	Open page in Chrome, Firefox, Safari, and Edge > Verify page layout	Page layout should be consistent across all b

Prepared By: Mariyam Asif

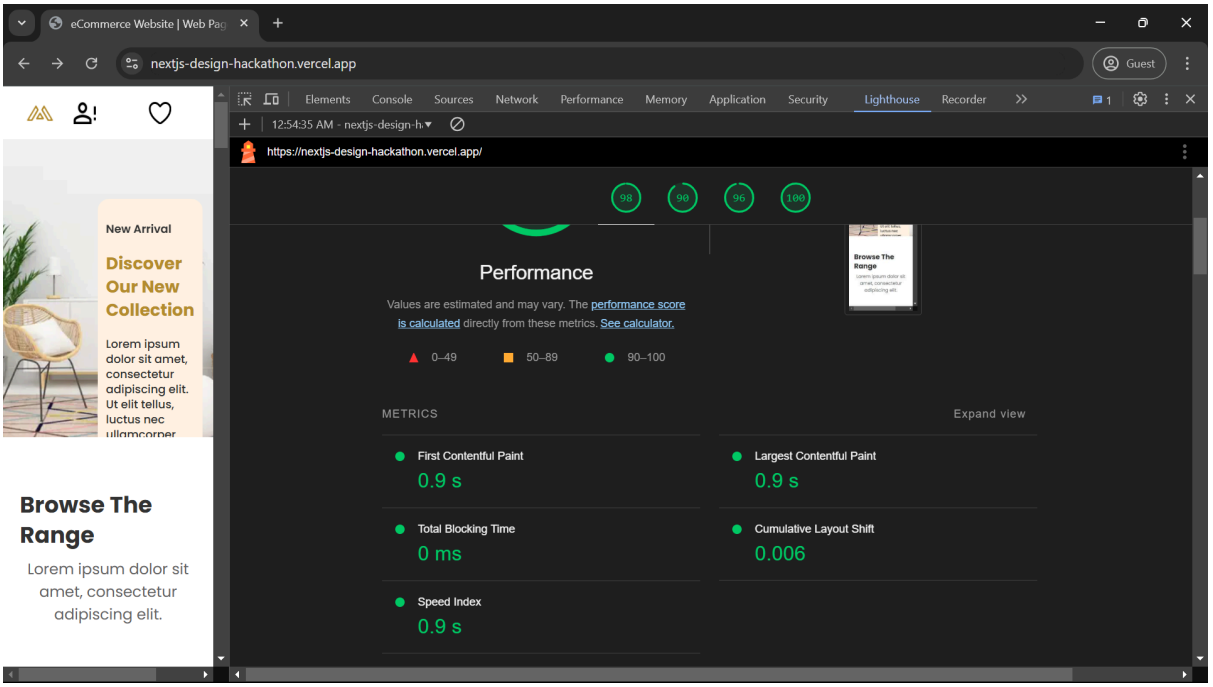
6. Performance Testing Report:

I performed **performance testing** using **Lighthouse** to analyze the speed, responsiveness, and overall performance of the application. The key metrics from the Lighthouse report include the following:





Prepared By: Mariyam Asif



Conclusion:

This deployment process has successfully set up a robust staging environment for the online marketplace, ensuring secure configuration, smooth deployment, and thorough testing. With the performance reports and test cases documented, the application is now ready for further development and eventual production deployment. The next steps will involve refining features and preparing for a full-scale launch.

Prepared By: Mariyam Asif