MarketPlace-Builder-Hackathon

Day-6: Deployment Preparation And Staging Environment Setup

1/21/2025
GIAIC: Sunday(2pm to 5pm)
Qaimudin Khuwaja
00161235

OVERVIEW

On **Day 6**, the focus was on preparing the marketplace for deployment by setting up a **staging environment**, configuring the **hosting platforms**, and ensuring the application's readiness for **real-world use**. This day was critical to simulate the production environment and to catch any potential issues before deploying the application to a live, customer-facing environment. The goal was to test the application in a **production-like setting**, which helps to identify any deployment or configuration issues that might arise during the final deployment to production.

Deployment Strategy Planning

For the staging deployment, I chose Vercel as the **hosting platform** due to its fast and efficient deployment process, which is ideal for both static and dynamic applications. Vercel seamlessly integrates with GitHub, allowing me to link the GitHub repository to the platform for continuous integration and continuous deployment. This setup enables automatic deployments whenever changes are pushed to the repository. I also configured the build settings and deployment scripts in Vercels dashboard, ensuring that the build process, including bundling, environment setup, and resource compilation, was optimized for the staging environment. This approach ensured a smooth deployment process and facilitated testing the application in a production-like setting.

Environment Variables Configuration

I created a **.env file** locally to securely store all sensitive configuration data, including **API keys**, **Sanity CMS credentials**, and other environment-specific variables like **database connections**. This file ensured that sensitive information was not exposed in the codebase. After setting up the file, I securely uploaded these environment variables to the hosting platform's dashboard. This step ensured that the application could access the required credentials and settings in the staging environment, while also keeping sensitive data protected. By using this approach, I ensured the application functioned correctly in the staging environment with appropriate production or staging credentials.

Staging Environment Setup

I deployed the application to a **staging environment** and made sure the build process finished without any errors. After the deployment, I checked that the website was loading properly and tested important features like product listings, product details page, search, checkout, and cart operations. Everything was working as expected, and the application was ready for further testing. I also checked other essential features to make sure the user experience was consistent and error-free. Everything was functioning as expected, which confirmed that the staging deployment was successful and the application was ready for further testing and validation.

Staging Environment Testing

Functional Testing

I performed testing of the application's key features such as browsing products, adding items to the cart, and completing the checkout process. Since the product data is fetched from Sanity CMS, I made sure that the data was properly retrieved and displayed on the site. I used tools like Postman for validating the APIs to ensure that the data was being fetched and processed correctly from Sanity. Everything worked as expected, and the application responded correctly to user actions.

Performance Testing

I used the **Lighthouse** tool for performance testing to check how well the website performs. The tool gave me important details on **performance**, **accessibility**, **best practices**, and **SEO**. It helped me measure things like **page load time**, **page speed**, and how responsive the site is. Lighthouse also pointed out any areas that could be improved to make the site faster and easier for users to navigate. This testing ensured that the site performs well under different conditions and that there are no issues slowing down the user experience.

Security Testing

I performed **security testing** to check for common issues which could make the site vulnerable to attacks. I also made sure that HTTPS was properly set up to protect data during transmission. Additionally, I verified that sensitive information, like API keys, was securely handled and not exposed. Finally, I checked that the site had the necessary secure HTTP headers in place to protect it from potential attacks.

Test Reporting

I documented all the test results and any issues found during testing. The test case report included details like the **Test Case ID**, **steps** followed, **expected results**, **actual results**, and the **status** (whether it passed or failed). I also made note of any **unresolved issues** that could be fixed later. This helped keep track of everything that was tested and any areas that needed attention.

Test Case ID	Description	Steps	Expected Result	Actual Result	Status	Remarks
TC001	Validate Product Listing	Open Product Listing Page > Verify that products are displayed correctly	Products displayed correctly from Sanity CMS	Products displayed correctly	Passed	No issues found
TC002	Validate Product Details Page	Verify product details are fetched from Sanity And display	Product details are displayed correctly	Product details fetched correctly from **Sanity CMS**	Passed	Works as expected
TC003	Test Cart Functionality	Add an item to the cart > Verify the cart is updated correctly	Cart updates with the correct product and details	Cart updated correctly with the added item	Passed	Cart functionality working fine
TC004	Test Checkout Process	Verify that checkout form loads correctly	Checkout page loads with correct product information and form	Checkout page loads with correct product info	Passed	No issues found
TC005	Test Search Functionality	Verify relevant products are displayed according search	Relevant products appear in search results	Relevant products displayed	Passed	Search functionality working fine
TC006	Test Order Placement	Verify order confirmation	Order confirmation page displayed with details	Order confirmation page displayed correctly with details	Passed	Order placement work as expected
TC007	Test Responsiveness	Resize browser window > Verify layout adjusts properly	Layout adjusts to different screen sizes (mobile/tablet/desktop)	Layout adjusts properly on all devices	Passed	Responsive design verified

Checklist for Day 6

Deployment Preparation	✔
Staging Environment Testing	✔
Documentation	✔
Form Submission	✔
Final Review	✔

Prepared By: Qaimudin Khuwaja