Day 6 - Deployment Preparation and Staging Environment Setup: Accomplished Tasks

On Day 6, I successfully prepared my marketplace for deployment by completing all necessary steps to set up a staging environment, ensuring the application is ready for production.The purpose was to prepare the marketplace for deployment by setting up a staging environment, configuring hosting platforms, and ensuring the application's readiness for a production-like environment. This stage builds on the testing and optimization efforts from Day 5.

Accomplishments

1. Hosting Platform Setup

- Selected **Vercel** as the hosting platform for its seamless integration and deployment capabilities.
- Connected my **GitHub repository** to Vercel, configured the build settings, and deployed the application to the staging environment.
- Ensured all build processes ran smoothly without errors.
- 🌐 **Live URL**: Visit My Website

---

2. Environment Variable Configuration

- Created a secure `.env` file to manage sensitive data such as API keys and project IDs.
- Uploaded these environment variables securely to the Vercel dashboard, ensuring their integration into the deployment process.

---

3. Deployment to Staging

- Successfully deployed the marketplace to a fully functional staging environment.
- Verified that all essential features, including product listings, cart operations, and search functionality, worked as expected in the staging environment.

---

4. Staging Environment Testing

- Conducted **Functional Testing** using Cypress to verify workflows, interactions, and API responses.

- Performed **Performance Testing** with Lighthouse to analyze site speed, load times, and responsiveness. Achieved a high-performance score.
- Executed **Security Testing** to ensure input validation, HTTPS implementation, and secure handling of sensitive data such as API keys.
- Documented all test cases in a professional CSV report, including Test Case ID, description, steps, expected results, actual results, status, and remarks. Sample test results:
    - TC001: Product listing functionality – Passed.
    - TC002: API error handling – Passed.
    - TC003: Cart operations – Passed.
    - TC004: Responsiveness – Passed.
    - TC005: Homepage loading speed – Passed
    - TC006: Product filtering functionality – Passed
    - TC007: Cart addition from product page - Passed
    - TC008: Checkout functionality - Passed

Test Cases Report

Test Case Id	Description	Steps	Expected Result	Acutal Result	Status	Remarks	
TC001	Product listing functionality	Navigate to the product page. Verify all products are displayed correctly.	Products displayed as per database records.	Products displayed correctly.	Passed	No issues.	
TC002	API error handling	Disconnect API. Refresh the page.	Show fallback message.	Fallback message displayed gracefully.	Passed	Handled gracefully.	
TC003	Cart operations	Add items to the cart. Verify item count updates in the cart.	Cart updates correctly with added items.	Cart updated correctly.	Passed	Works as expected.	
TC004	Responsiveness	Resize browser window. Check layout adjustments.	Layout adjusts properly for all screen sizes.	Layout adjusted properly.	Passed	Responsive verified.	
TC005	Homepage loading speed	Load the homepage. Measure the loading time using Lighthouse.	Page loads within 2 seconds.	Page loaded in 1.8 seconds.	Passed	Optimal performance.	
TC006	Product filtering functionality	Apply filters (e.g., category, price). Verify filtered product list.	Filters apply correctly, and results update.	Filters applied correctly. Results updated.	Passed	No issues.	
TC007	Cart addition from product page	Add product to cart directly from the product page.	Product is added to the cart.	Product added to the cart successfully.	Passed	Works perfectly.	
TC008	Checkout functionality	Complete checkout workflow (cart > shipping > payment).	Order placed successfully.	Order placed without errors.	Passed	Payment processed	

Summary of Test Cases

1. **Product Listing Functionality**: Verified that all products were displayed accurately on the product page.
   Status: Passed.
2. **API Error Handling**: Tested application behavior when the API is disconnected, ensuring fallback messages are displayed.
   Status: Passed.
3. **Cart Operations**: Ensured the cart updated correctly when items were added, with accurate item counts displayed.
   Status: Passed.
4. **Responsiveness**: Confirmed the layout adjusted appropriately for different screen sizes and resolutions.
   Status: Passed.
5. **Homepage Loading Speed**: Measured and validated homepage loading times using Lighthouse, achieving optimal performance under 2 seconds.
   Status: Passed.
6. **Product Filtering Functionality**: Tested filters (e.g., category and price) to ensure the filtered product lists updated correctly.
   Status: Passed.
7. **Cart Addition from Product Page**: Verified that products could be added to the cart directly from the product page.
   Status: Passed.
8. **Checkout Functionality**: Tested the complete checkout workflow, ensuring successful order placement, including payment processing.
   Status: Passed.

Conclusion

All key functionalities were thoroughly tested in the staging environment, with every test case passing successfully. This ensures the marketplace is robust, responsive, and ready for deployment in a production environment.

Detailed Performance Analysis

- **Performance:** 85
- **Accessibility:** 87
- Best Practices: **100**
- **SEO:** 100

Security Measures Implemented

1. Secure User Authentication

- Enforced **strong password policies** requiring minimum complexity and length.
- Integrated **Firebase Authentication** for secure login and session management.
- Implemented **rate-limiting** on login attempts to prevent brute force attacks.

---

2. Data Protection

- Secured data transmission by implementing **HTTPS** with an SSL/TLS certificate.
- Stored sensitive data like API keys and database credentials in **environment variables**.
- Encrypted sensitive user information, such as passwords, using **bcrypt**.

---

3. Input Validation and Sanitization

- Validated all user inputs to prevent invalid or malicious data from entering the system.
- Sanitized inputs to protect against **Cross-Site Scripting (XSS)** attacks.
- Used **parameterized queries** to prevent **SQL Injection** vulnerabilities.

---

4. API and Backend Security

- Restricted access to APIs with secure **API keys** and role-based access control (RBAC).
- Implemented **rate limiting** to avoid **DDoS attacks** on backend APIs.
- Ensured all API communications occur over **HTTPS**.

---

5. Cross-Site Scripting (XSS) Protection

- Used proper escaping and sanitization techniques for user-generated content.
- Added security headers via **Helmet.js** in the backend to block potential XSS attacks.

---

6. Cross-Site Request Forgery (CSRF) Protection

- Implemented **CSRF tokens** to validate requests coming from trusted sources.
- Ensured forms and requests were protected from unauthorized cross-origin attacks.

---

7. Secure File Uploads

- Restricted allowed file types for uploads to prevent malicious file execution.
- Validated and sanitized uploaded file content before storage.

---

8. Regular Updates

- Kept all frameworks, libraries, and dependencies up-to-date to ensure security patches are applied promptly.

---

9. DDoS and Bot Protection

- Configured **Cloudflare** for DDoS protection and to filter malicious traffic.
- Added **CAPTCHAs** to forms to deter bot attacks.

---

10. Content Security Policy (CSP)

- Defined a strict **Content Security Policy** to control resources loaded on the website and prevent untrusted scripts.

---

11. Secure Session Management

- Configured cookies with `HttpOnly`, `Secure`, and `SameSite` attributes to protect session data.
- Implemented session expiration and required re-authentication for sensitive actions.

---

12. Monitoring and Error Tracking

- Integrated **Sentry** for real-time monitoring and logging of application errors.
- Set up alerts for suspicious activities, such as repeated failed login attempts or abnormal API usage.

---

5. Documentation Updates

- Created a detailed `README.md` file summarizing all six days of activities, including:
  - Deployment steps.
  - Test case results.
  - Performance analysis reports.
- Organized all project files in a well-structured GitHub repository:
  - `documents/` folder for reports and test cases.
  - `src/` folder for the source code.
  - `public/` folder for static assets.
  - Included the `README.md` file in the root directory for easy navigation.

Expected Outputs Achieved

1. Deployed and validated the staging environment for the marketplace.
2. Securely configured environment variables in the hosting platform.
3. Documented test case results and performance analysis in the repository.
4. Maintained a clean and structured GitHub repository, meeting professional standards.
5. Completed and published a README.md file summarizing the project activities.

Conclusion of Day 6

Day 6 marked a significant milestone in the project as I successfully deployed the marketplace to a staging environment, ensuring it functions seamlessly in a production-like setting. By configuring a secure hosting platform, managing environment variables, and conducting extensive testing, I validated the application's readiness for deployment. Additionally, I documented all processes, organized project files, and created a comprehensive README.md file, setting a strong foundation for future production deployment. This phase has ensured that the marketplace is secure, optimized, and prepared to deliver a seamless experience to end users.

Submission

- Deployed Staging Environment: ✅ Completed and functional.
- GitHub Repository: ✅ Uploaded with all required files, folders, and documentation.
- Submission Form: ✅ Submitted before the deadline.

With Day 6 completed, my marketplace is now deployment-ready and tested in a production-like environment. This marks a significant milestone in ensuring a seamless and professional end-user experience.
