DAY 6 - DEPLOYMENT PREPARATION AND STAGING ENVIRONMENT SETUP

Prepared by Ameen Alam

First published: PKT 11:08 AM, January 15, 2025

Document Revision Information

Version	**Date**	**Amendment**	**Author**
1.0	11:08 AM, January 15, 2025	Initial release of Day1	Ameen Alam
1.1	03:00 PM, January 15, 2025	Added Examples in Day1	Ameen Alam
2.0	01:00 PM, January 16, 2025	Day 2 Release	Ameen Alam
2.1	02:00 PM, January 17, 2025	Self-Validation Checklist	Ameen Alam
3.0	04:00 PM, January 17, 2025	Day 3 Release	Ameen Alam
3.1	08:30 PM, January 17, 2025	Minor correction in link	Ameen Alam
4.0	04:00 AM, January 19, 2025	Day 4 Release	Ameen Alam
5.0	02:30 AM, January 20, 2025	Day 5 Release	Ameen Alam
5.1	03:00 AM, January 20, 2025	Test Case Report Sample	Ameen Alam
6.0	03:10 AM, January 21, 2025	Day 6 Release	Ameen Alam

Table of Contents

Day 5 Recap:

Day 5 focused on testing and backend refinement to ensure all marketplace components were functioning as intended. Students conducted:

1. Functional testing to verify workflows, such as product listings, cart operations, and API interactions.
2. Performance testing using tools like Lighthouse to analyse speed, responsiveness, and load times.
3. Security testing to validate input fields, secure API keys, and ensure HTTPS implementation.
4. Documentation of both successful and failed test cases in a professional CSV format.
5. Comprehensive reporting to identify pending issues for future resolution.

By the end of Day 5, students had a clear understanding of their marketplace's readiness and the gaps to address in subsequent stages

Day 6 - Deployment Preparation and Staging Environment Setup

Objective:

Day 6 focuses on preparing your marketplace for deployment by setting up a staging environment, configuring hosting platforms, and ensuring readiness for a customer-facing application. Building on the testing and optimization work from Day 5, this stage emphasizes ensuring the marketplace operates seamlessly in a production-like environment. Students will also learn about industry-standard practices for managing different environments like non-production (TRN, DEV, SIT) and production (UAT, PROD, DR).

Key Learning Outcomes:

1. Set up and configure a staging environment for your marketplace. This includes:
   - Selecting a hosting platform such as **Vercel** or Netlify.
   - Connecting your **GitHub** repository to the platform.
   - Configuring build and deployment settings to ensure successful staging builds.

   - Setting up environment variables securely within the hosting platform.
   - Validating the application functionality in a production-like environment to identify and resolve any pre-deployment issues.
2. Understand professional environment management, including TRN, DEV, SIT, UAT, and PROD stages.
3. Conduct staging environment testing and document results.
4. Create professional deployment documentation, including performance and test case reports.
5. Organize all project files and documents in a structured GitHub repository. This includes maintaining a clear folder hierarchy (e.g., documents/, src/, public/, etc.) and using consistent naming conventions for files and folders. Ensure each file is placed in a logical location and provide a README.md file summarizing the project structure and contents.

Professional Environment Types:

1. **TRN (Training):** Used for onboarding and practice.
2. **DEV (Development):** The environment for writing and testing code locally.
3. **SIT (System Integration Testing):** Validates integrations between systems.
4. **UAT (User Acceptance Testing):** Allows stakeholders to test functionality and validate requirements.
5. **PROD (Production):** The live, customer-facing environment.
6. **DR (Disaster Recovery):** A backup environment for critical situations.

Key Areas of Focus:

1. Deployment Strategy Planning

- Choose a hosting platform like **Vercel (Recommended)**, Netlify, AWS, or Azure for staging.
- Finalize the application's interaction with backend services such as Sanity CMS and third-party APIs.

2. Environment Variable Configuration

- Secure API keys, database credentials, and sensitive data using .env files.
- Configure environment variables in the hosting platform for secure deployment.

3. Staging Environment Setup

- Deploy the application to a staging environment to test it in a production-like setting.
- Validate that deployment builds successfully and the site loads correctly.

4. Staging Environment Testing

- Conduct functional, performance, and security testing. For functional testing, use tools like Cypress to test workflows and interactions, and Postman to validate API responses. Performance testing can be done with **Lighthouse** or **GTmetrix** to analyze load times, speed, and responsiveness. For security testing, validate input fields to prevent **SQL injection**, ensure **HTTPS** is enabled, and check for proper handling of sensitive data like **API keys**.
- Verify responsiveness and error handling.
- Document all test results and unresolved issues.

5. Documentation Updates

- Create a README.md file summarizing all six days of activities.
- Include all reports, test cases, and deployment instructions in a structured GitHub repository.

Steps for Implementation:

Step 1: Hosting Platform Setup

1. **Choose a Platform:**
    - Use platforms like **Vercel** or Netlify for quick deployment.
    - For advanced configurations, consider AWS or Azure.
2. **Connect Repository:**
    - Link your GitHub repository to the hosting platform.
    - Configure build settings and add necessary scripts for deployment.

Step 2: Configure Environment Variables

1. **Create a .env File:**

Include sensitive variables like API keys and tokens.

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

2. **Upload Variables to Hosting Platform:**
   - Use the hosting platform's dashboard to securely add environment variables.

Step 3: Deploy to Staging

1. **Deploy Application:**
   - Deploy the application to a staging environment through the hosting platform.
2. **Validate Deployment:**
   - Ensure the build process completes without errors.
   - Verify basic functionality in the staging environment.

Step 4: Staging Environment Testing

1. **Testing Types:**
   - **Functional Testing:** Verify all features, such as product listing, search, and cart operations.
   - **Performance Testing:** Use Lighthouse or GTmetrix to analyze speed and responsiveness.
   - **Security Testing:** Validate input fields, HTTPS usage, and secure API communications.
2. **Test Case Reporting:**
   - Document all test cases in a CSV file with fields like Test Case ID, Description, Steps, Expected Result, Actual Result, Status, and Remarks. For example:

Test Case ID	**Description**	**Steps**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
TC001	Validate product listing	Open product page > Verify products	Products displayed	Products displayed	Passed	No issues found
TC002	Test API error handling	Disconnect API > Refresh page	Show fallback message	Fallback message shown	Passed	Handled gracefully
TC003	Check cart functionality	Add item to cart > Verify cart	Cart updates correctly	Cart updates correctly	Passed	Works as expected

TC004	Test responsiveness layout	Resize browser window > Check layout	Layout adjusts properly	Layout adjusts properly	Passed	Responsive verified

3. **Performance Testing:**
    - Submit a performance report generated by tools like Lighthouse in your GitHub repository.

Step 5: Documentation Updates

1. **Create README.md:**
    - Summarize all project activities, including deployment steps and test case results.
2. **Organize Project Files:**
    - Ensure all files from Days 1 to 6 are in a structured folder hierarchy.

Expected Output:

1. A fully deployed staging environment for the marketplace.
2. Environment variables securely configured.
3. Test case and performance reports documenting staging tests.
4. All project files and documentation organized in a GitHub repository.
5. A professional README.md file summarizing project activities and results.

Submission Requirements:

Submission Form Link:

https://forms.gle/nA5Lv867KpaV659r7

What to Submit:

1. Staging environment deployed Link.
2. A new GitHub repository with:
    - A documents folder containing all Day 1 to Day 6 documents.
    - Test case report in CSV format.
    - Performance testing results.
    - Organized project files.

    - A README.md file summarizing all project activities/Folder Structure.

- **Deadline:** 22 January 2025 at 11:59 AM (Afternoon).

Checklist for Day 6:

Deployment Preparation:

- ✓ ✗

Staging Environment Testing:

- ✓ ✗

Documentation:

- ✓ ✗

Form Submission:

- ✓ ✗

Final Review:

- ✓ ✗

FAQs:

1. **Why is a staging environment necessary?**
    - A staging environment allows you to test your application in a production-like setting without affecting live users.
2. **What should my test report include?**
    - Include all test cases (passed and failed) with details like Test Case ID, Description, Steps, Expected Result, Actual Result, Status, and Remarks.
3. **How do I document performance testing?**
    - Use tools like Lighthouse or GTmetrix to generate a performance report and include it in your GitHub repository.
4. **What if I find major issues during staging tests?**
    - Focus on documenting the issues for now. Resolving them can be part of post-hackathon activities.
5. **What is the purpose of the README.md file?**
    - It provides an overview of your project, deployment steps, and results, making it easier for others (including clients) to understand your work.