Q-Commerce Website Development - Day 6 Documentation

Table of Contents

Day 5 Recap

Day 5 focused on testing and backend refinement to ensure all marketplace components were functioning as intended. Key activities included:

1. **Functional Testing: Verified workflows such as product listings, cart operations, and API interactions.**
2. **Performance Testing: Used tools like Lighthouse to analyze speed, responsiveness, and load times.**
3. **Security Testing: Validated input fields, secure API keys, and ensured HTTPS implementation.**
4. **Documentation: Created CSV reports of both successful and failed test cases.**
5. **Comprehensive Reporting: Identified pending issues for future resolution.**

By the end of Day 5, students had a clear understanding of their marketplace's readiness and 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. This stage emphasizes testing the marketplace in a production-like environment to ensure seamless operations.

Key Learning Outcomes

1. **Set up and configure a staging environment for your marketplace.**
   - **Select a hosting platform such as Vercel or Netlify.**
   - **Connect your GitHub repository to the platform.**
   - **Configure build and deployment settings to ensure successful staging builds.**
   - **Set up environment variables securely within the hosting platform.**
   - **Validate application functionality in a production-like environment.**
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.**

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.**
- **Use tools like Cypress, Postman, Lighthouse, or GTmetrix for comprehensive testing.**
- **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 and configure build settings.**

Step 2: Configure Environment Variables

1. **Create a .env File: Include sensitive variables like API keys and tokens.**
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 and 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.**

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

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 and folder structure.**
3. **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.**

Umama Rajput