Day 5 - Testing, Error Handling, and Backend Refinement - FoodTuck

By uzma mumtaz thursday 9 to 12

Objective

Day 5 focuses on ensuring that the "FoodTuck" restaurant website is rigorously tested, optimized for peak performance, and refined for an intuitive user experience. The main goals include backend integration testing, robust error handling, and seamless readiness for real-world deployment.

Key Learning Outcomes

1. Comprehensive testing of both functional and non-functional aspects.
2. Deployment of robust error handling mechanisms.
3. Enhancement of website performance and responsiveness.
4. Verification of compatibility across multiple browsers and devices.
5. Delivery of professional testing documentation with actionable insights.

Key Areas of Focus

1. Functional Testing

- Validate all core functionalities, including menu listing, search filters, and the ordering workflow.
- Test cart operations such as adding, removing, and updating items.
- Verify dynamic routing capabilities, such as individual menu item detail pages.

2. Error Handling

- Implement clear, user-friendly error messages for scenarios like network failures or server errors.
- Display fallback UI elements (e.g., "No menu items available" when API data is unavailable).

3. Performance Optimization

- Use performance auditing tools such as Lighthouse and GTmetrix to identify bottlenecks.
- Optimize images, reduce unused JavaScript and CSS, and implement caching strategies to enhance performance.

4. Cross-Browser and Device Testing

- Test functionality and layout on popular browsers, including Chrome, Firefox, Safari, and Edge.
- Verify responsiveness across desktop, tablet, and mobile devices.

5. Security Testing

- Ensure proper validation of input fields to prevent injection attacks like SQL or XSS.
- Use HTTPS for secure API communication and store sensitive data in environment variables.

6. User Acceptance Testing (UAT)

- Simulate real-world usage scenarios, such as browsing the menu, placing orders, and checking out.
- Gather feedback to refine usability and identify potential roadblocks.

7. Documentation Updates

- Summarize testing results, fixes, and optimization steps in a clear format.
- Compile comprehensive test reports, including a CSV file for detailed tracking.

Steps for Implementation

Step 1: Functional Testing

- Develop and execute detailed test cases for functionalities like menu listing, search, cart, and checkout processes.
- Validate backend API responses using tools like Postman.

Step 2: Error Handling

- Integrate error-catching mechanisms with informative feedback to the user.

Example Code:

```
 try {
    const data = await fetchMenuItems();
    setMenu(data);
} catch (error) {
    console.error("Failed to fetch menu items:", error);
    setError("Unable to load menu items. Please try again later.");
}
```

Step 3: Performance Optimization

- Compress images with tools like TinyPNG or ImageOptim.
- Implement lazy loading for heavy assets.
- Analyze performance metrics using Lighthouse and address flagged issues, such as reducing render-blocking resources.

Step 4: Cross-Browser and Device Testing

- Utilize BrowserStack or LambdaTest to simulate multiple browser environments.
- Manually test responsiveness on at least one physical mobile device to ensure accurate rendering.

Step 5: Security Testing

- Sanitize input fields and validate user data formats using regular expressions.
- Conduct vulnerability assessments using OWASP ZAP or Burp Suite.

Step 6: User Acceptance Testing (UAT)

- Perform user-like interactions to validate workflows and functionality.
- Incorporate constructive feedback from peers or mentors to enhance usability.

Step 7: Documentation Updates

- Compile all testing outcomes and resolutions into a CSV file.
- Summarize findings in a professional report, detailing challenges and solutions.

Expected Output

By the end of Day 5, the "FoodTuck" website should deliver:

1. Fully tested and validated functional components.
2. Transparent and user-friendly error handling mechanisms.
3. Optimized performance for reduced load times and improved interactivity.
4. Verified compatibility across all targeted browsers and devices.
5. Comprehensive CSV-based testing documentation.
6. A polished and professional report summarizing testing results and optimization efforts.

Submission Requirements

1. **Functional Deliverables:**

   Showcase functional and responsive components through testing logs or reports from tools like Lighthouse and Postman.

2. **Testing Report (CSV Format):**

   Include the following columns:

   - Test Case ID
   - Description
   - Steps
   - Expected Result
   - Actual Result
   - Status
   - Severity Level
   - Remarks

CSV Data:

Test Case ID	Description	Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Verify menu listing displays correctly	Navigate to the homepage, check menu section	Menu items displayed	Menu items displayed	Passed	low	Work as expected
TC002	Test category filter	Select category you desired	Relevant items displayed	Relevant items displayed	Passed	Medium	Work as expected
TC003	Test search functionality	Enter keyword in search bar, press enter	Relevant items displayed	Relevant items displayed	Passed	low	some issue found
TC004	Test cart functionality	Select item then click add to cart button	Items added/removed correctly	Items added/removed correctly	Passed	Low	Need improvement
TC005	Handle API failure gracefully	Simulate API failure, observe error message	"Unable to load menu" message displayed	"Unable to load menu" message displayed	passed	Heigh	Work as expected
TC006	Ensure responsiveness on mobile	Resize browser or test on mobile device	Responsive design adapts	Responsive design adapts	passed	medium	Test successful
TC007	Validate input fields	Enter invalid data in form fields	Error message displayed	Error message displayed	passed	medium	No issue found

Documentation:

- Submit a professionally formatted PDF or Markdown file summarizing testing outcomes.

4. **Repository Submission:**
    - Upload all updated project files to the designated GitHub repository, ensuring a clear folder structure and an updated README file.

Checklist

- Functional Testing Completed
- Error Handling Mechanisms Implemented
- Performance Optimization Steps Finalized
- Cross-Browser and Device Testing Performed
- Documentation Compiled and Submitted