Day 5: Testing, Error Handling, and Backend Integration

Refinement Points

Functional Testing

Functional testing ensures that all features of the application work as expected under normal and edge-case scenarios. This includes testing individual functions, integration of different modules, and verifying that the system behaves as intended. For example, testing user login, form submissions, and data retrieval processes.

Error Handling

Error handling focuses on identifying and managing unexpected conditions during runtime. This includes verifying proper error messages, fallback mechanisms, and logging systems to capture errors. For instance, checking how the application responds to invalid input, network disruptions, or server errors.

Performance Optimizing

Performance optimization involves assessing the application's speed, resource usage, and scalability under varying loads. Key areas include reducing page load times, optimizing database queries, and minimizing memory leaks. Tools like Lighthouse, JMeter, or APM software can help monitor these metrics.

Security Testing

Security testing evaluates the application's resilience against vulnerabilities such as SQL injection, cross-site scripting (XSS), and unauthorized access. Using frameworks like OWASP guidelines ensures the application adheres to industry standards and protects sensitive user data.

User Acceptance

User acceptance testing (UAT) validates that the application meets the end-user requirements and performs as expected in real-world scenarios. This involves engaging users to test functionalities, providing feedback, and ensuring usability aligns with their expectations.

UI Testing

UI testing examines the application's layout, design consistency, and responsiveness across multiple devices and screen sizes. This ensures a seamless user experience, checking for proper alignment, font consistency, and usability on mobile, tablet, and desktop devices.

Best Practices

1. Write clear and reusable test cases.
2. Automate repetitive tests wherever possible.
3. Maintain separate environments for testing and production.
4. Use version control for test scripts.
5. Document all bugs and fixes systematically.

Test Case Table

1	**ID**	**Test Case Description**	**Test Steps**	**Expected Result**	**Actual Result**	**Status**	**Severity Level**	**Remarks**
2	TC001	Validate product listing	Product verified	Successful display products	Successful login	Passed	High	No problem
3	TC002	Check for api error handling	Discount API	Display error message	Display error message	Passed	Medium	successfully
4	TC003	Check cart funcnality	Add to cart	Load under added product	Load under 2 seconds	Passed	High	Optimized
5	TC004	Validate backend integration	Call API endpoints	Correct data returned	Correct data returned	Passed	High	Work as expected

Summary

On Day 5, we focused on comprehensive testing and integration efforts. The team validated core functionalities, optimized performance, and ensured robust error handling. Security tests ensured compliance with best practices, and user acceptance tests highlighted areas requiring minor adjustments. The process reinforced the importance of structured testing and iterative improvements to deliver a reliable product.