

Marketplace-Builder- Hackathon

Day-5: Testing, Error handling and Backend Integration Refinement

1/21/2025

GIAIC: Monday (2pm to 5pm)

Syed Irfan Hussain Zaidi

00003451

OVERVIEW

The main goal of DAY-5 was to prepare the marketplace for real-world deployment by thoroughly testing all components, identifying and resolving errors, optimizing performance, and refining backend integrations. This involved performing functional testing to ensure core features like product listings, search, and cart operations worked as expected, implementing error handling to display user-friendly messages and fallback UI for API failures, optimizing performance by reducing load times and compressing images, and conducting cross-browser and device testing to ensure consistent user experience. Additionally, security testing was conducted to validate inputs and secure API communication.

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Product Listing - Image Fetch Failure	1. Go to product listing. 2. Simulate network issue.	Product images should load correctly	Error message "Image fetch failed" appeared due to network issue.	Failed	Medium	Fixed by resolving the network issue.
TC002	Checkout Process with Missing Cart Items	1. Attempt to checkout with no items in the cart. 2. Verify error message.	A message should appear saying "Your cart is empty."	"Cart is empty" message appeared correctly.	Passed	Low	Checkout Functionality worked as expected
TC003	Product Search Functionality	Enter a search term Verify search results are relevant.	Relevant products related to search should appear.	Relevant results appeared for the search.	Passed	Medium	Search works correctly.
TC004	Error Handling for API Failure	Disconnect internet or simulate API failure.	Error message should appear.	Error message displayed as expected.	Passed	High	Error handling works successfully
TC005	Mobile Responsiveness	Verify that the product layout adapts correctly to all screen size.	Layout should be responsive across all devices	Layout was responsive on all screen sizes was no issues.	Passed	Low	Mobile version tested successfully.
TC006	Security - Input Validation for Email Field	Enter an invalid email address	Form should reject invalid email formats with an error message	Invalid email format was rejected with an error.	Passed	High	Email validation works as expected.
TC007	User Login with Incorrect Credentials	1. Enter incorrect username/password.	User should see an error message: "Invalid username or password."	Error message displayed correctly.	Passed	Medium	Handled Successfully as expected.

Error Report

Subject: Build Errors and Image Loading Issues

Issue Description

During the build process and testing, the application encountered the following issues:

- 1. Build Errors Due to useSearchParams Usage Without <Suspense> Boundary**
 - The useSearchParams hook is being used in multiple components and pages without being wrapped in a <Suspense> boundary. This causes errors during static generation or server-side rendering.
- 2. Image Loading Issues on the Product Listing Page**
 - While navigating to the product listing page, images fail to load, and an error message stating "Image fetch failed" is displayed. This issue is caused by a network problem that disrupts fetching images from Sanity.

Error Logs

● Build Errors

The following errors were observed during the build process

Error: useSearchParams() should be wrapped in a suspense boundary at page "/404".
Error: useSearchParams() should be wrapped in a suspense boundary at page "/cart".
Error: useSearchParams() should be wrapped in a suspense boundary at page "/order-confirmation".
Error: useSearchParams() should be wrapped in a suspense boundary at page "/".
Error: useSearchParams() should be wrapped in a suspense boundary at page "/payment".
Error: useSearchParams() should be wrapped in a suspense boundary at page "/shop".
Error: useSearchParams() should be wrapped in a suspense boundary at page "/test".

● Image Loading Errors

Error: Image fetch failed: Network error while fetching images from Sanity.

Root Cause

The useSearchParams hook is being used in components and pages without being wrapped in a <Suspense> boundary. This violates React's rules of hooks and causes errors during the build process because useSearchParams is a client-side hook and cannot be used during static generation or server-side rendering.

Image Loading Issue

- The images are fetched from Sanity using a network request. A network disruption or incorrect configuration is causing the image fetch to fail, resulting in the "Image fetch failed" error.

Affected Code

- **/shop/page.tsx**
 - The ShopContent component uses useSearchParams but is not wrapped in <Suspense>.
- **/src/components/shop-page/filters/FilterContext.tsx**
 - The FilterProvider component uses useSearchParams but is not wrapped in <Suspense>.
- **Other Pages**
 - Pages such as /cart, /order-confirmation, /payment, /test, and /_not-found also use useSearchParams without <Suspense>.

Image Loading Issue

1. /shop/page.tsx (Product listing page)

```
const query = `*_type=="products" {
  "id": _id,
  name,
  "image": image.asset->url,
  price,
  discountPercent
}`;
```

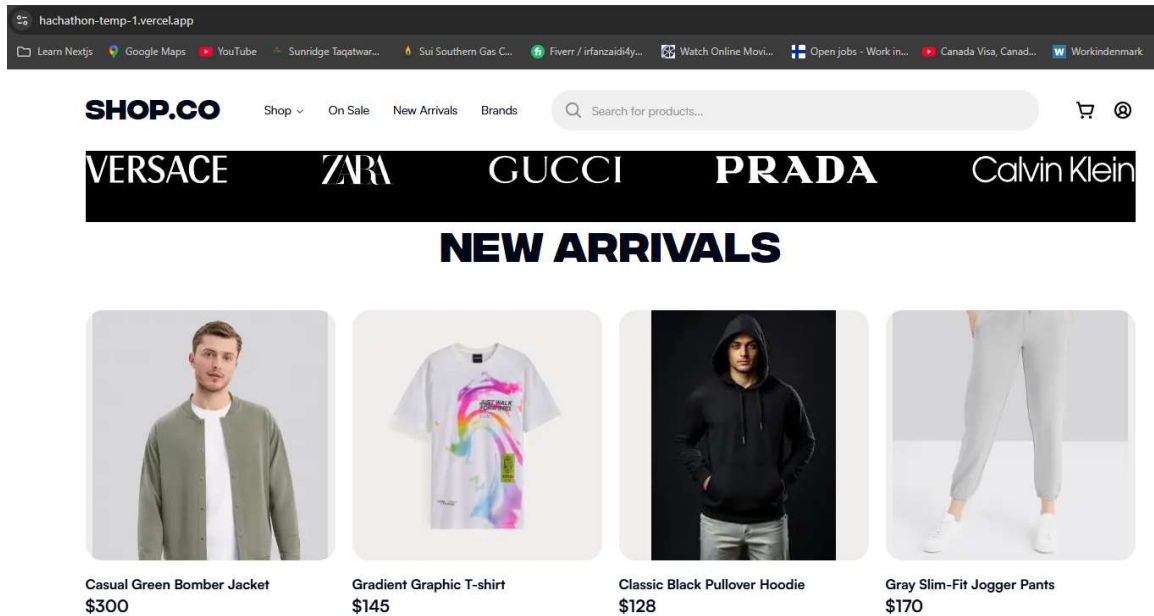
Proposed Solution

Fix useSearchParams Issue

- Wrap all instances of useSearchParams in a <Suspense> boundary.

Fix Image Loading Issue

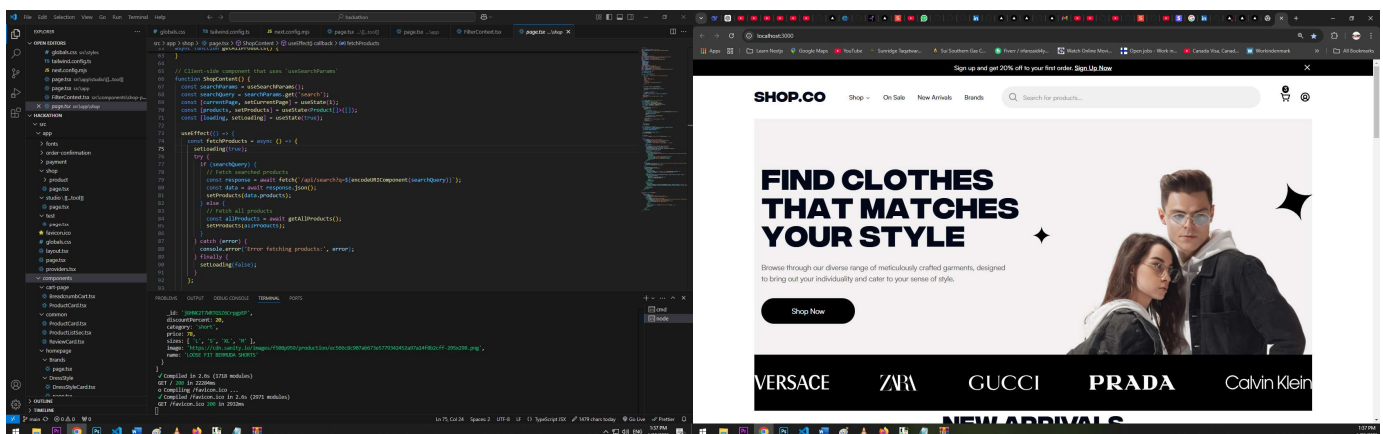
- Ensure the network configuration is correct and stable.
- Add error handling for image fetching to display a fallback image or message if the fetch fails.



Mobile Responsiveness

The marketplace passed the responsiveness test without any issues. When tested on various screen sizes, the layout adjusted seamlessly, ensuring that all content (product listings, filters, and checkout buttons) was appropriately displayed. Elements like buttons, images, and text remained properly aligned, and no content was cut off or misplaced.

No errors were encountered during responsiveness testing. The website was fully responsive on popular devices, including desktop (Chrome), tablet and mobile.



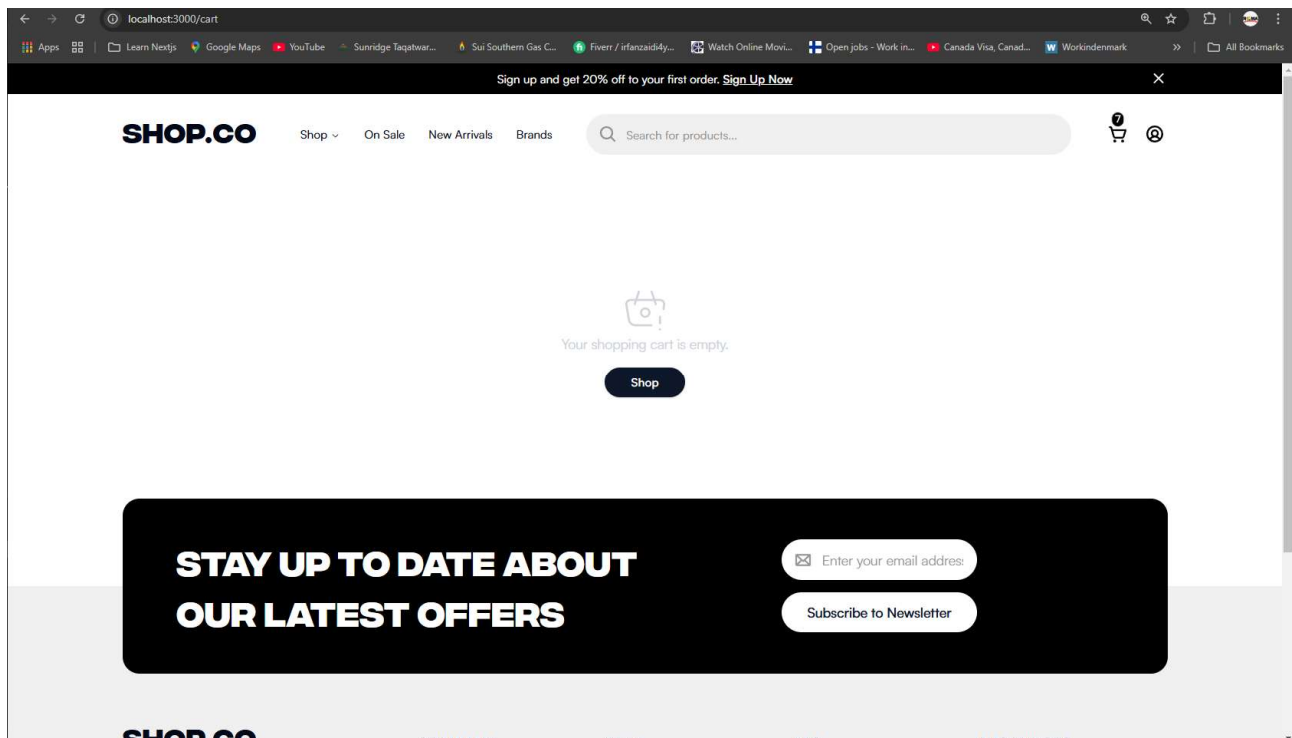
Checkout Process

Problem Faced

During testing, it was observed that when users attempted to proceed to the checkout page with no items in the cart, the system did not prevent them from doing so.

Resolution

To resolve this, a validation step was added to check whether the cart is empty before proceeding to the checkout page. If the cart is empty, the user is now presented with a clear message such as **"Your cart is empty."** After implementing the fix, the checkout process was tested again with an empty cart, and the system correctly displayed the error message, preventing users from proceeding further.



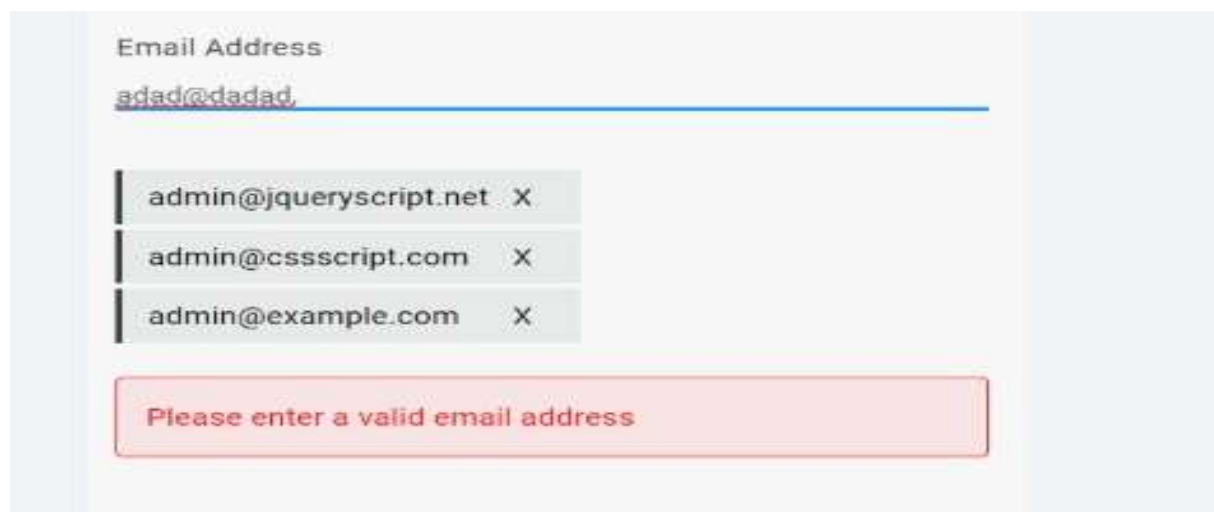
Security- Input Validation for Email Field

Problem Faced

During security testing, it was found that the email input field did not have sufficient validation to prevent invalid inputs. This could potentially allow users to enter invalid email formats.

Resolution

The issue was resolved by implementing proper **input validation** for the email field. After implementing the validation, the system correctly rejected any input that did not match a valid email format and prevented harmful scripts from being submitted.



The screenshot shows a web form with the label "Email Address". The input field contains the text "adad@dadad,". Below the input field, there is a list of three suggested email addresses, each followed by an "X" icon: "admin@jqueryscript.net", "admin@cssscript.com", and "admin@example.com". At the bottom of the form, there is a red-bordered box containing the text "Please enter a valid email address".

Email Address	
adad@dadad,	
admin@jqueryscript.net	X
admin@cssscript.com	X
admin@example.com	X

Please enter a valid email address

Conclusion

In conclusion, Day 5 of the testing, error handling, and backend integration refinement phase successfully addressed a variety of critical issues to ensure that the marketplace is ready for a seamless, user-friendly, and secure deployment. The primary focus areas included validating core functionalities, improving the user experience, and enhancing security measures.

- Functional Testing
- Error Handling
- Security Improvements
- Performance Optimization and Responsiveness
- Cross-Browser and Device Testing

End