

Hackathon 3 day 5

Detailed Testing Report

Objective: Day 5 focused on preparing the marketplace application for real-world deployment. The tasks included comprehensive testing, error handling, backend integration refinement, and performance optimization.

Key Learning Outcomes

- Comprehensive testing of functional and non-functional aspects of the application.
- Implementation of robust error handling mechanisms.
- Performance optimization for speed and responsiveness.
- Ensuring cross-browser compatibility and device responsiveness.
- Submission of professional testing documentation, including CSV-based reports.

Functional Testing

- Conducted extensive functional tests to ensure that core features of the marketplace worked as intended:
 - **Product Listing Pages:** Verified that all product data was displayed accurately.
 - **Search:** Tested the search bar for accurate results.
 - **Cart Operations:** Ensured users could add, update, and remove items from the cart seamlessly.
 - **Dynamic Routing:** Checked individual product detail pages for correct data rendering and navigation.

- **Check out page:** Insured that checkout page collects user info properly, deploys it to the firebase database and routes user to the order confirmation page.

Error Handling Implementation

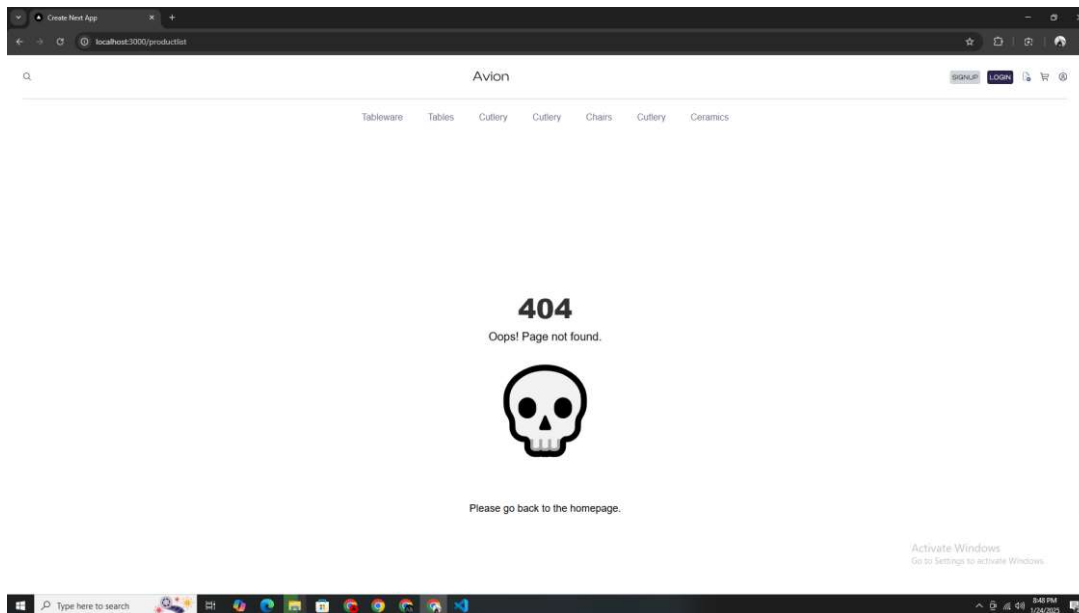
Implemented robust error handling mechanisms:

- **Network Failures:**

Added fallback messages such as "Unable to load products. Please try again later."

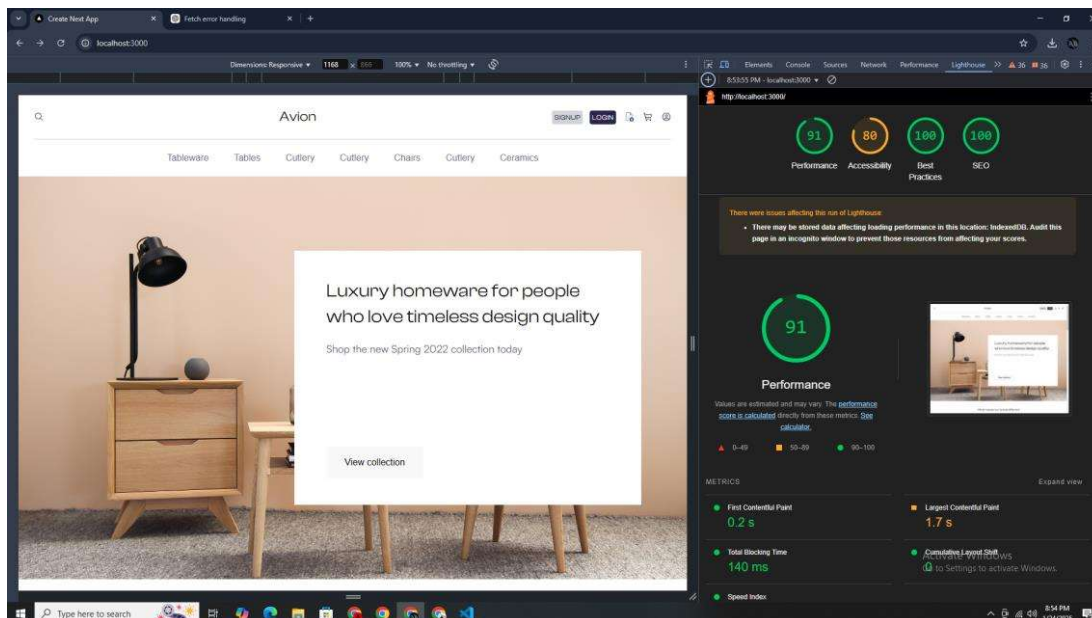
- **Invalid Data:** Displayed appropriate error messages for missing or incorrect data.
- **Unexpected Server Errors:** Logged errors and showed user-friendly fallback UI.

```
50 children: React.ReactNode;
51 }> {
52   const sanityres = await client.fetch(`*[_type == "product"]{
53     _id,
54     name,
55     "slug": slug.current,
56     "imageUrl": image.asset->url,
57     price,
58     quantity,
59     tags,
60     description,
61     features,
62     dimensions,
63     category->{
64       name,
65       "slug":slug.current
66     }
67   }`);
68   if (!sanityres || sanityres.length === 0) {
69     return <div>Product not found.</div>;
70   }
71   return (
72     <html lang="en" className={` ${satoshiFont.variable} ${clashFont.variable} ${integralCFFont.variable}`}>
73       <body
74         className={` ${geistSans.variable} ${geistMono.variable} antialiased`}
75       >
76         <AppWrapper><Navbar product={sanityres}/>{children}</AppWrapper>
77       </body>
78     </html>
79   );
80 };
```



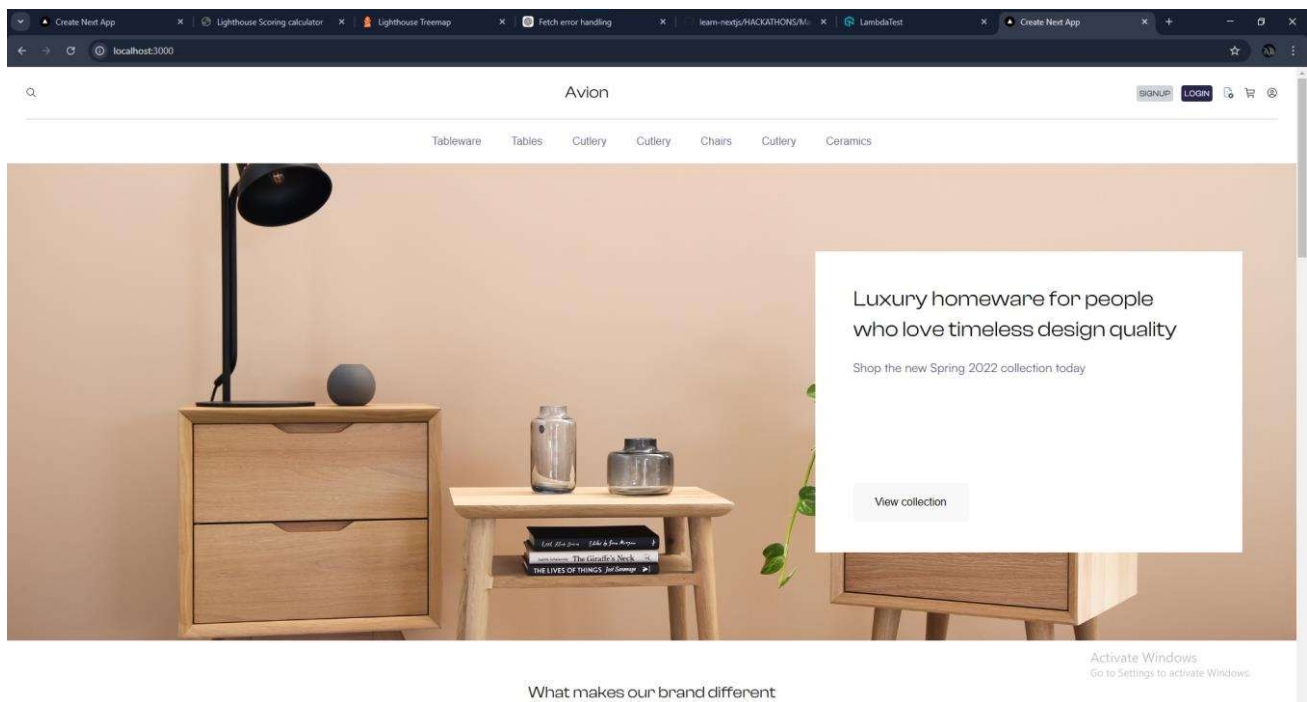
Performance Optimization

- **Code Optimization:**
 - Minimized unused CSS and JavaScript.
 - Enabled caching for static assets.
- **Performance Testing:**
 - Used Lighthouse to identify bottlenecks.
 - Achieved 0.2 on speed index.

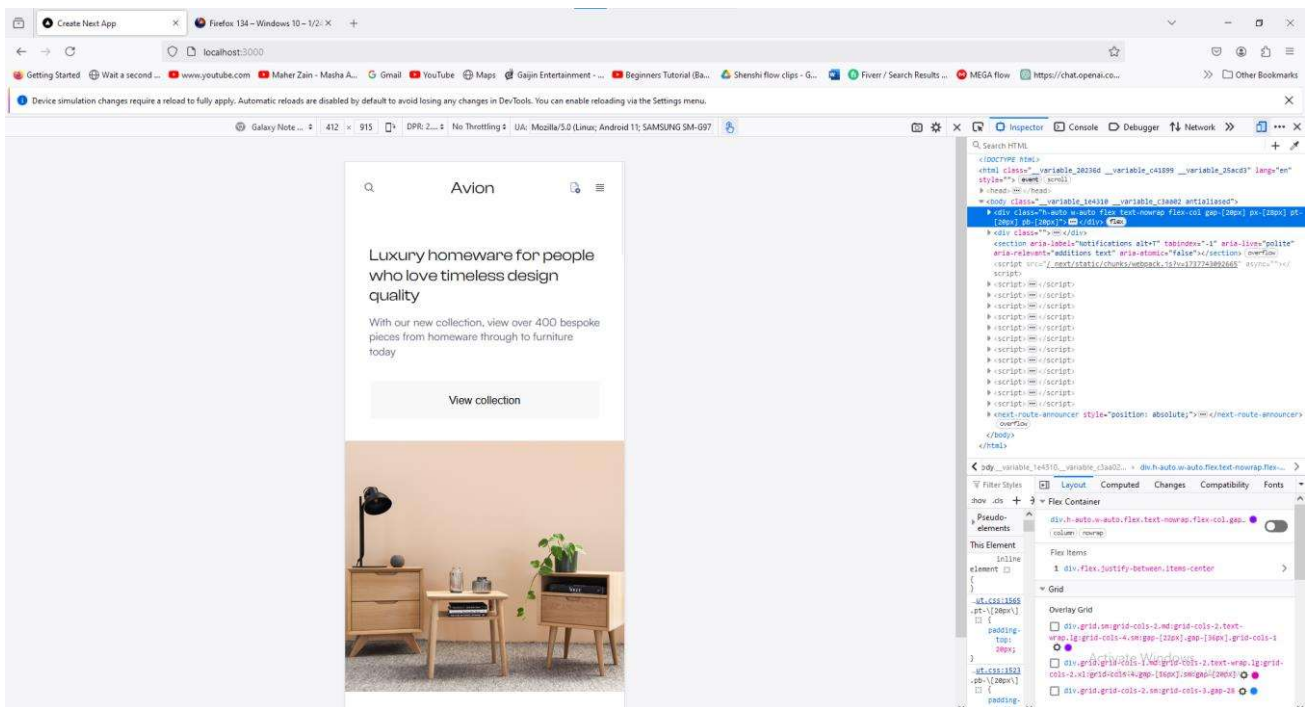
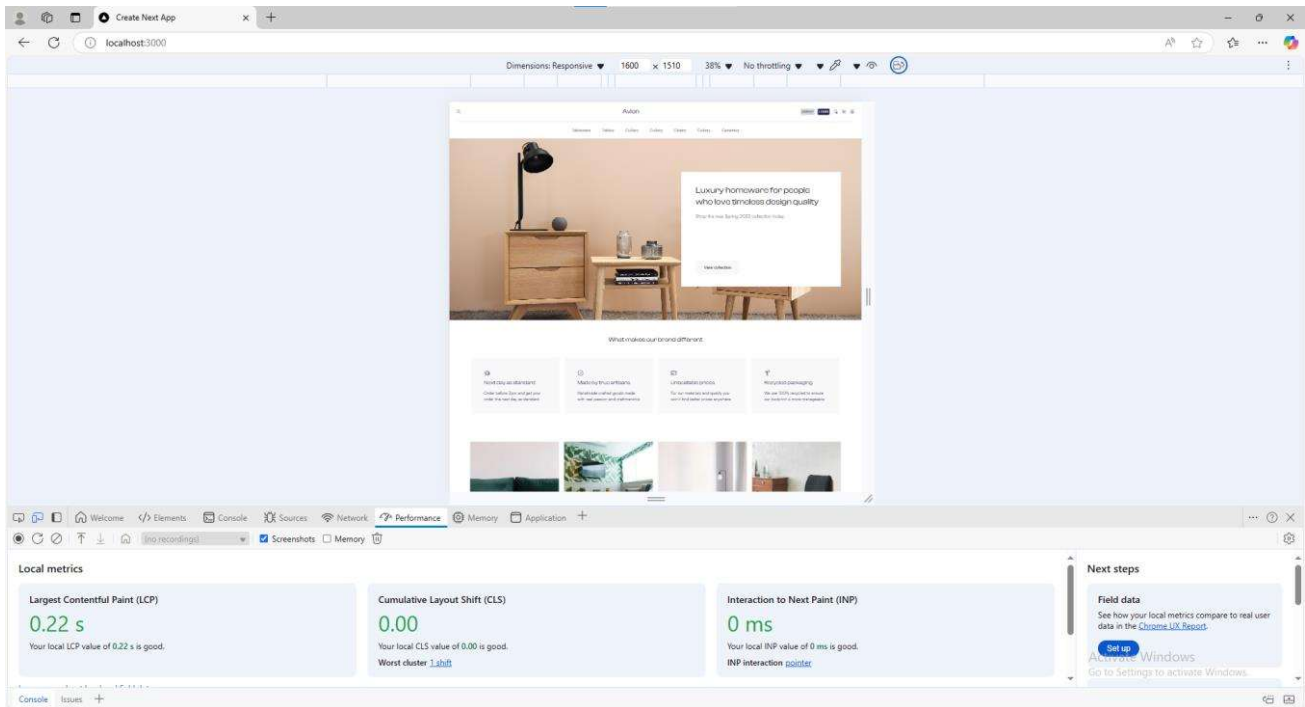


Cross-Browser and Device Testing

- Tested the application on major browsers:
 - Chrome, Firefox, and Edge.
- Verified responsiveness across devices:
 - Desktop, tablet, and mobile.
 - Used browsers dev tool to simulate various devices and screen sizes.



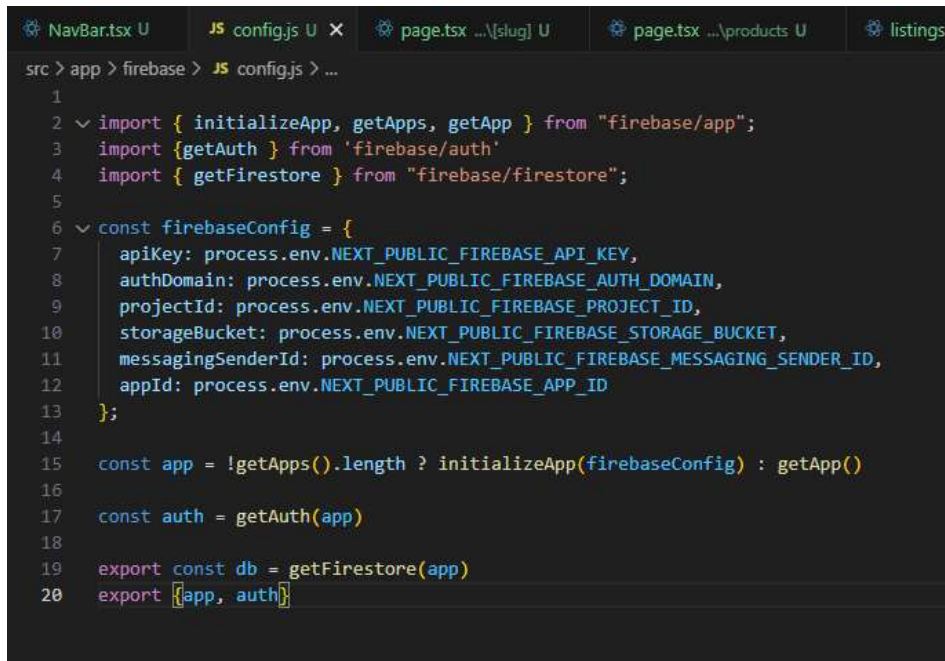
(Continue to next page...)



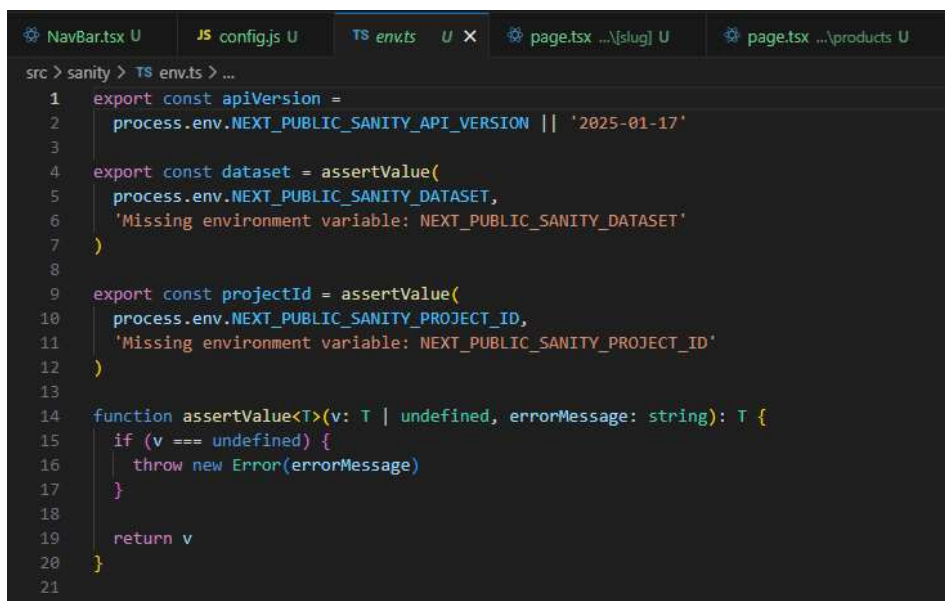
Security Testing

Measures Taken:

- Validated input fields to prevent injection attacks.
- Used HTTPS for secure communication.
- Stored sensitive API keys in environment variables.



```
src > app > firebase > JS config.js > ...
1
2 import { initializeApp, getApps, getApp } from "firebase/app";
3 import { getAuth } from "firebase/auth";
4 import { getFirestore } from "firebase/firestore";
5
6 const firebaseConfig = {
7   apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
8   authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
9   projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID,
10  storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
11  messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID,
12  appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID
13 };
14
15 const app = !getApps().length ? initializeApp(firebaseConfig) : getApp()
16
17 const auth = getAuth(app)
18
19 export const db = getFirestore(app)
20 export { app, auth }
```



```
src > sanity > TS env.ts > ...
1 export const apiVersion =
2   process.env.NEXT_PUBLIC_SANITY_API_VERSION || '2025-01-17'
3
4 export const dataset = assertValue(
5   process.env.NEXT_PUBLIC_SANITY_DATASET,
6   'Missing environment variable: NEXT_PUBLIC_SANITY_DATASET'
7 )
8
9 export const projectId = assertValue(
10  process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
11  'Missing environment variable: NEXT_PUBLIC_SANITY_PROJECT_ID'
12 )
13
14 function assertValue<T>(v: T | undefined, errorMessage: string): T {
15   if (v === undefined) {
16     throw new Error(errorMessage)
17   }
18
19   return v
20 }
21
```

User Acceptance Testing (UAT)

- Simulated real-world scenarios, including browsing, searching, and checkout workflows.
- Collected feedback from peers and made necessary adjustments.

Csv Testing Report

Custom Test Cases Report

	A	B	C	D	E	F	G	H	I
1	Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
2	TC001	Validate home page functionality	Open home page > Verify displayed content	Home page loads with all sections visible	Home page displays as expected	Passed	Low	-	No issues found
3	TC002	Validate product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	Low	-	No issues found
4	TC003	Test product detail page	Click on a product > Verify details	Product details displayed correctly	Product details displayed correctly	Passed	Medium	-	Works as expected
5	TC004	Validate cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	High	-	Cart works correctly
6	TC005	Test wishlist functionality	Add product to wishlist > Verify wishlist	Wishlist updates with added product	Wishlist updates correctly	Passed	Medium	-	Wishlist works as expected
7	TC006	Test add to cart functionality	Add product to cart > Verify cart count	Cart count updates correctly	Cart count updates correctly	Passed	High	-	Works as expected
8	TC007	Test remove from cart functionality	Remove product from cart > Verify cart count	Cart count decreases correctly	Cart count decreases correctly	Passed	High	-	Remove functionality works
9	TC008	Validate login functionality	Enter credentials > Click login	User logs in successfully	User logged in successfully	Passed	High	-	No issues found
10	TC009	Validate signup functionality	Enter details > Click signup	User account created successfully	Account created successfully	Passed	High	-	Signup works as expected
11	TC010	Validate checkout process	Add products to cart > Proceed to checkout	Checkout process completes successfully	Order placed successfully	Passed	Critical	-	Checkout process smooth
12	TC011	Validate order confirmation page	Place an order > Verify confirmation page	Order confirmation page displays correctly	Confirmation page displays as expected	Passed	Medium	-	Confirmation page works

Checklist for Today's Tasks:

- Functional testing ✓
- Error handling ✓
- Performance optimization ✓
- Cross-browser testing ✓
- Security testing ✓
- Documentation ✓