

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.
 - **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.

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

Error Handling Implementation

Implemented robust error handling mechanisms:

- Network Failures:

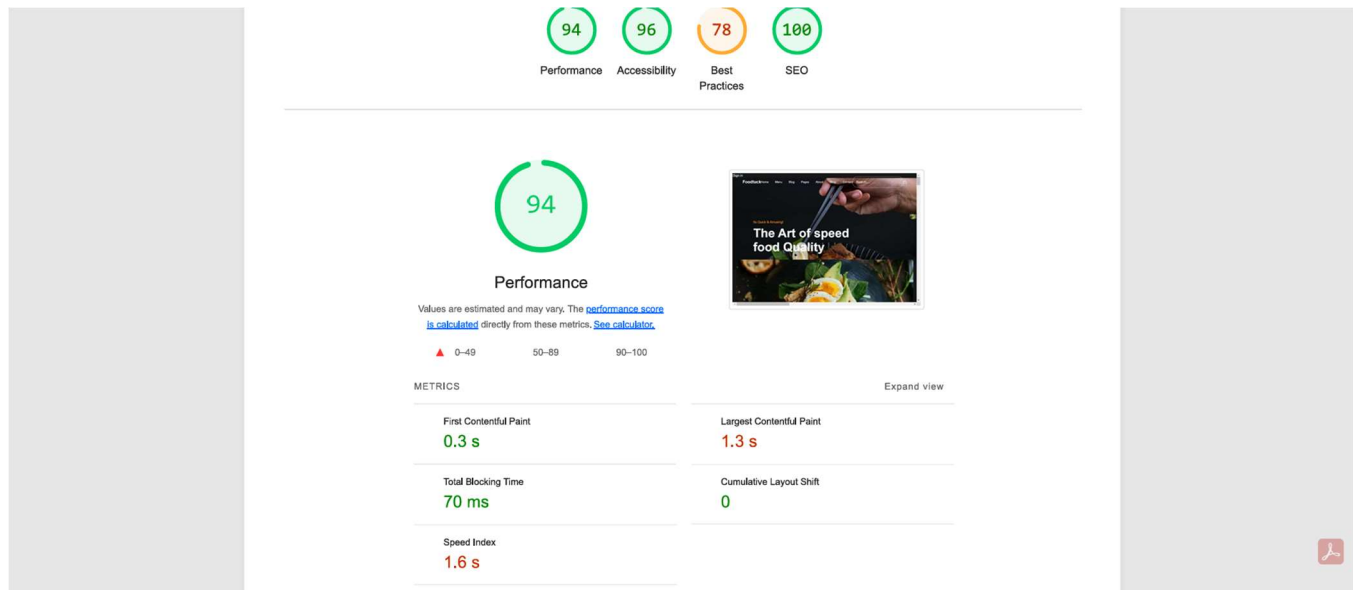
Added fallback messages such as "Unable to load products. Please try again later."

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

```
39
40 import ProductDetails from '@app/components/productDetails';
41 import { client } from '@sanity/lib/client';
42 import React from 'react';
43
44 async function getProduct(slug: string): Promise<Product | null> {
45   const query = `*[ _type == 'food' && slug.current == "${slug}" ] | order(_createdAt asc) {
46     _id,
47     name,
48     category,
49     originalPrice,
50     "discountPrice": price, // Assuming the discount price is stored in 'price'
51     tags,
52     "slug": slug.current,
53     "imageUrl": image.asset->url,
54     description,
55     available
56   }`;
57
58   // Fetch data from Sanity
59   const products = await client.fetch(query);
60   return products.length > 0 ? products[0] : null; // Return product if found, otherwise null
61 }
62
63 const ProductDetail = async ({ params }: { params: { slug: string } }) => {
64   const product = await getProduct(params.slug);
65
66   if (!product) {
67     return (
68       <div className="text-center py-10">
69         <h2 className="text-2xl font-bold text-red-500">Product not found</h2>
70         <p className="text-gray-600">The product you are looking for does not exist or is currently unavailable.</p>
71       </div>
72     );
73   }
74
75   return (
76     <div>
77       </* Render the product details for the fetched product */>
78       <ProductDetails product={product} key={product._id} />
79     </div>
80   );
81 };
```

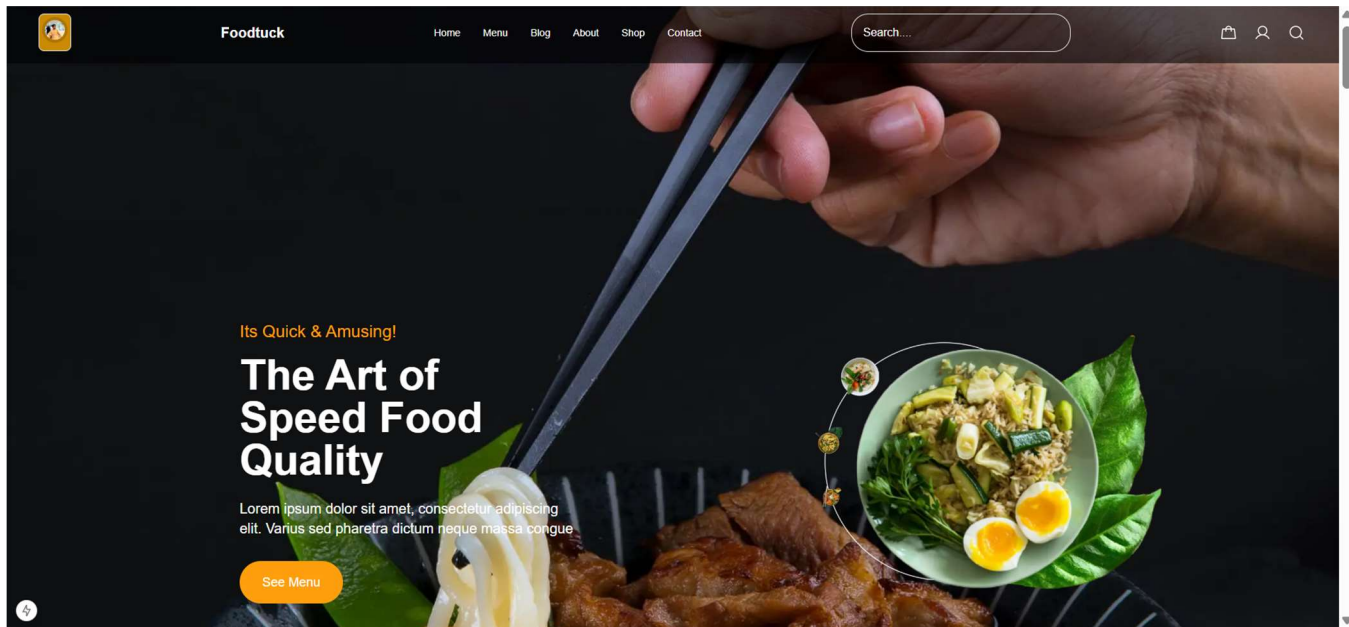
Performance Optimization

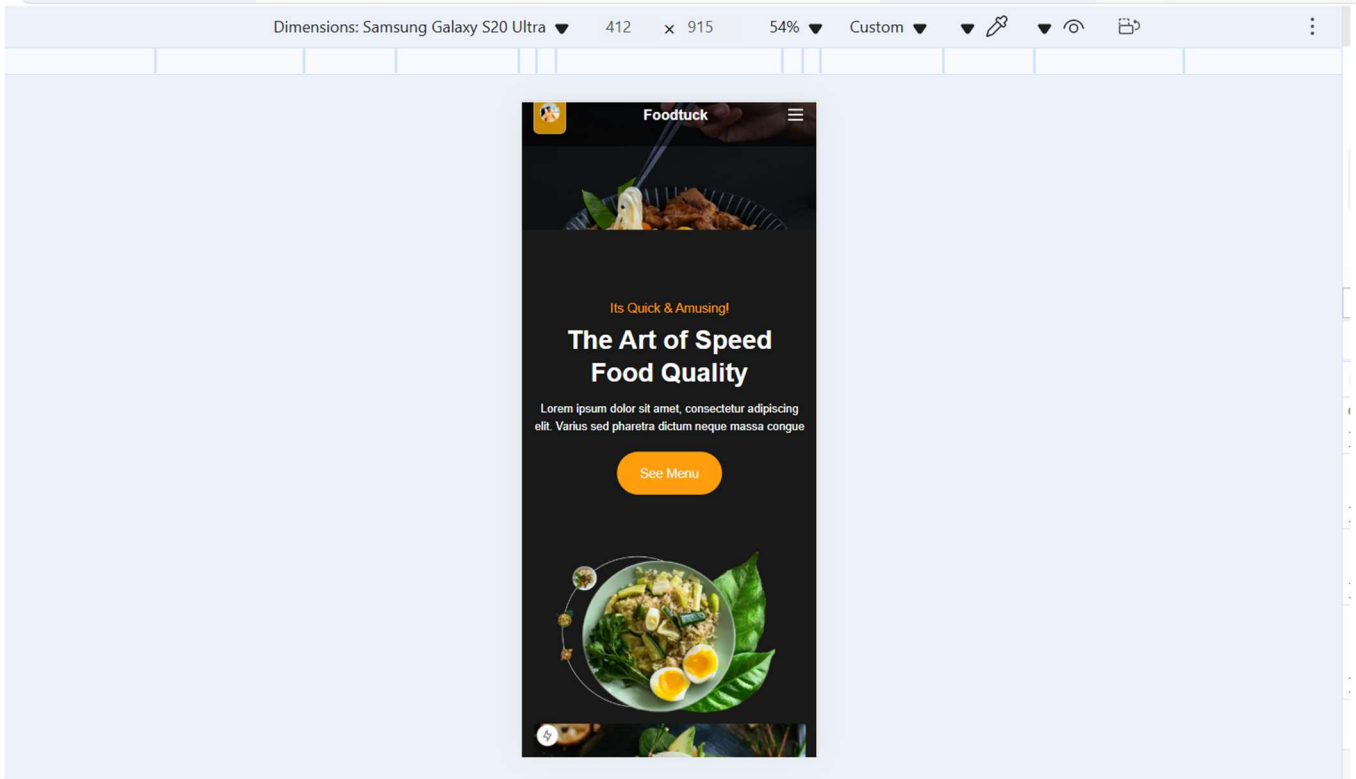
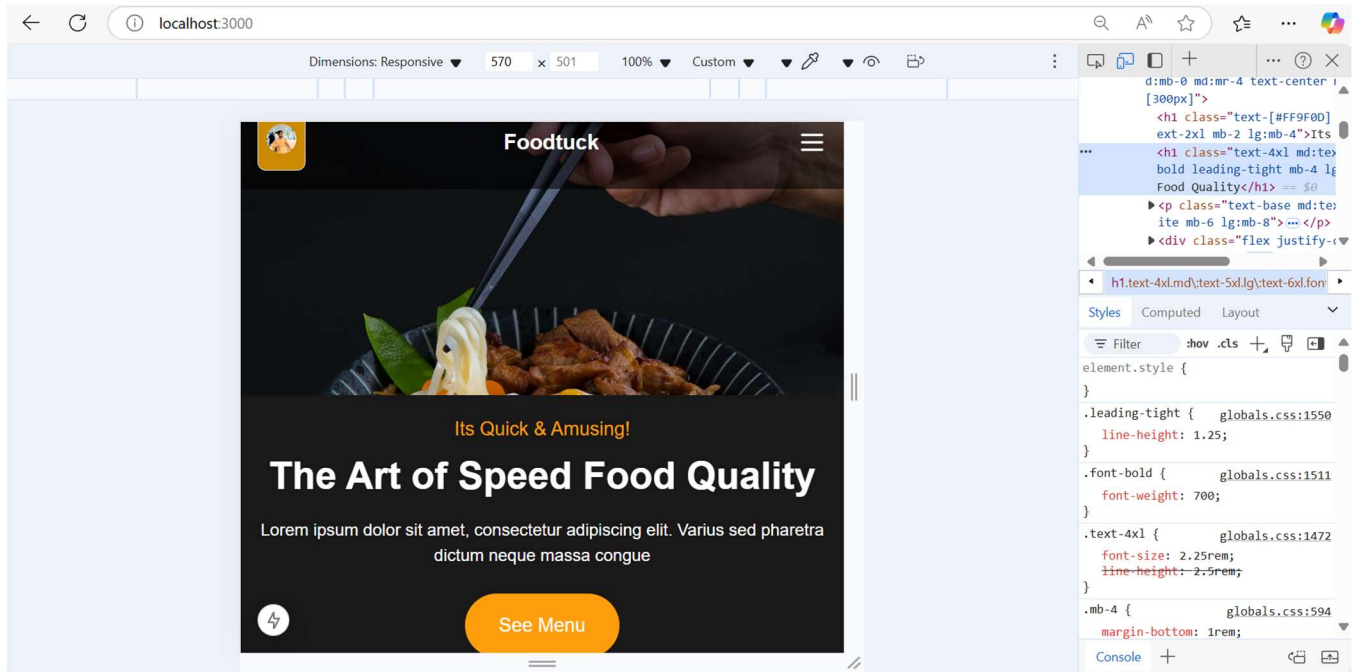
- Code Optimization:
 - Minimized unused CSS and JavaScript.
 - Enabled caching for static assets.
- Performance Testing:
 - Used Lighthouse to identify bottlenecks.Achieved 1.6 on speed index.



Cross-Browser and Device Testing

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

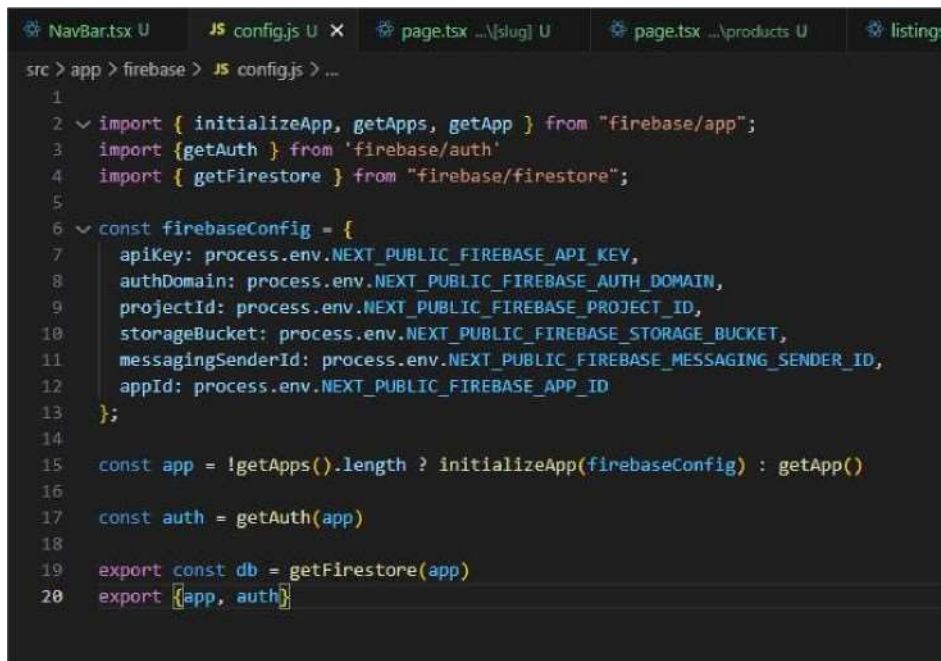




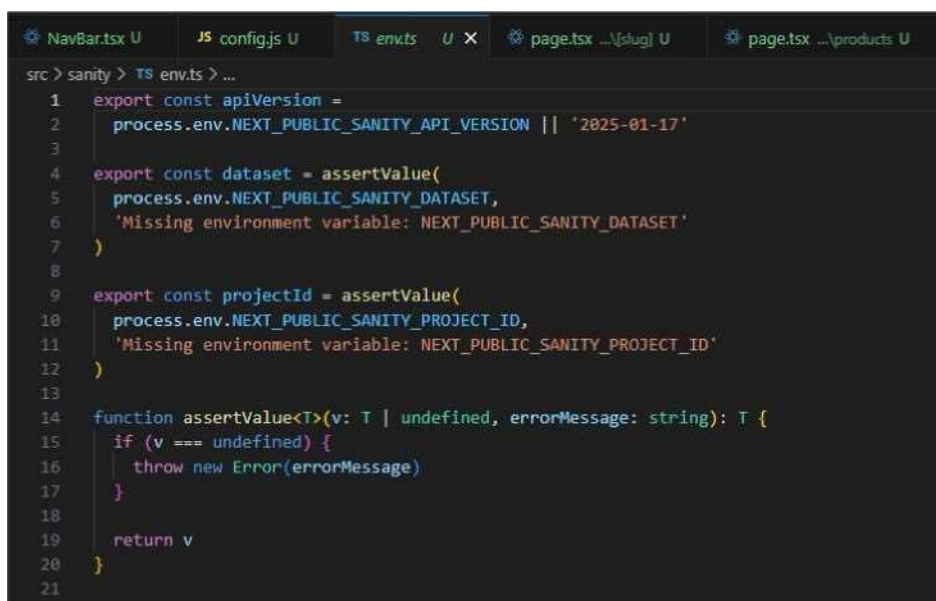
Security Testing

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

	A	B	C	D	E	F	G
1	Test Case Id	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Remarks
2	TC001	validate home page functionality	open home page > display content	home page load with all content	worked as expected	passed	no issue
3	TC002	validate product listing page	open product page > verify products	Product displayed accurately	Product displayed accurately	passed	no issue
4	TC003	test product detail page	click the product > verify details	Product details displayed accurately	Product details displayed accurately	passed	work as expected
5	TC004	validate cart functionality	Add product to carts	cart update with added product	cart update as expected	passed	cart works correctly
6	TC005	validate sign up functionality	click signup > fill the details > enter	user account created sucessfully	Account created sucessfully	passed	sign up worked
7	TC006	validate log in functionality	enter email > click login	user login sucessfully	user login sucessfully	passed	log in worked
8	TC007	validate checkout process	added products of cart > proceed to checkout	checkout completed	order placed	passed	checkout smoothly working
9	TC008	remove product of cart	click remove the product	product removed	product removed sucessfully	passed	remove function is worked

Checklist forToday's Tasks:

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