Functional Testing Report for Nike E-commerce Website

Prepared by: Huma Mohsin

Date: January 21, 2025

Objective

To validate that all features of the Nike e-commerce website function as intended, providing a seamless user experience for browsing, purchasing, and interacting with the website.

Scope of Testing

The testing covered critical functionalities, including navigation, user interaction, and backend integration to ensure a smooth shopping experience.

Functional Testing Results

1. Features Tested

Navigation Links:

Verified all links for correct redirection.

Help Route:

Displays store information but does not respond to interactions with icons (e.g., thumbs-up, call).

User Registration/Login:

User data is captured correctly but not stored in the database.

Sign-In:

Navigates correctly but fails to process submitted user information.

Gift Cards/Vouchers:

Works as expected; users can open the voucher window and select options or redeem vouchers.

Add to Cart:

Functionalities of Add to cart is in progress.

Search Bar:

Successfully brings a search items upon searching.

News and Features:

Works properly and includes pagination functionality for efficient browsing of articles or features. Users can navigate between pages of news or features.

Buttons (Men/Women):

Correctly navigates to the relevant category routes.

Picture Cards:

Displays product information but lacks functionality for navigation to specific product detail.

Explore More Section:

Displays categories men, women, kids

Footer Links:

Defined but non-functional for navigation.

Menu Items:

Structured well but lacks click event functionality.

News and features:

It works properly, adds Pagination feature also.

2. Product-Related Issues

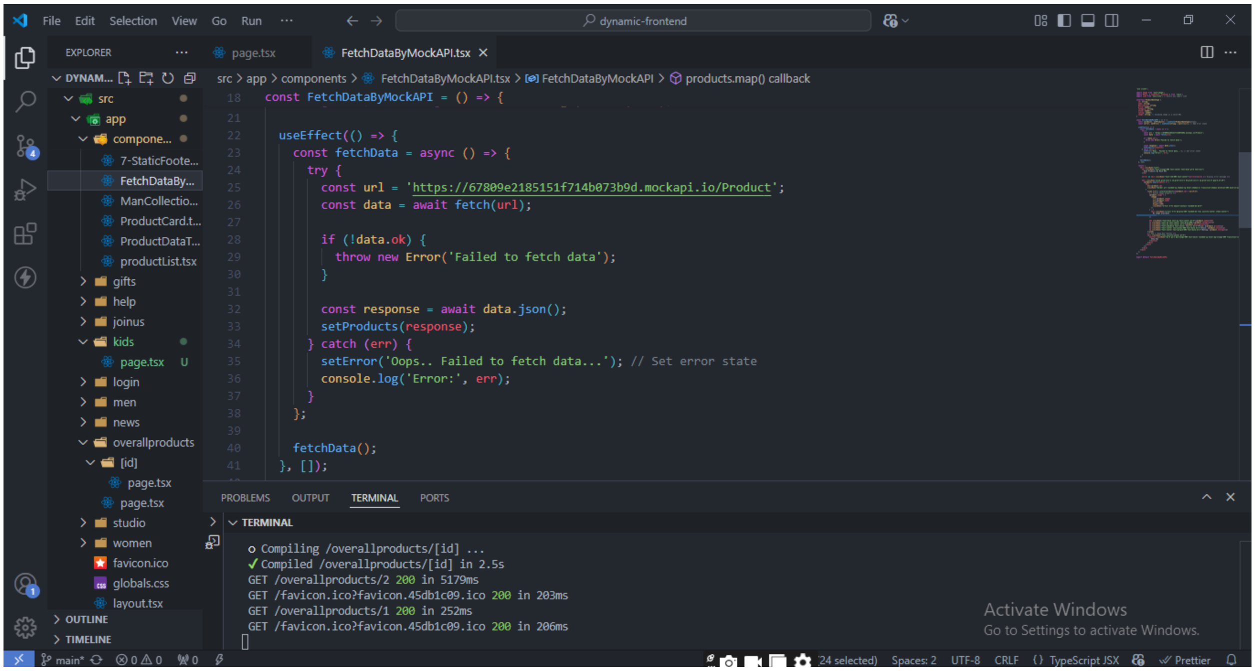
Products fetched from mock APIs instead of live databases.

Shop Now Button	Non-functional when a product list is displayed.
Dynamic Routes:	Fails to fetch and display product details of single product.
Filters and Hide Buttons:	Displayed but non-functional

.

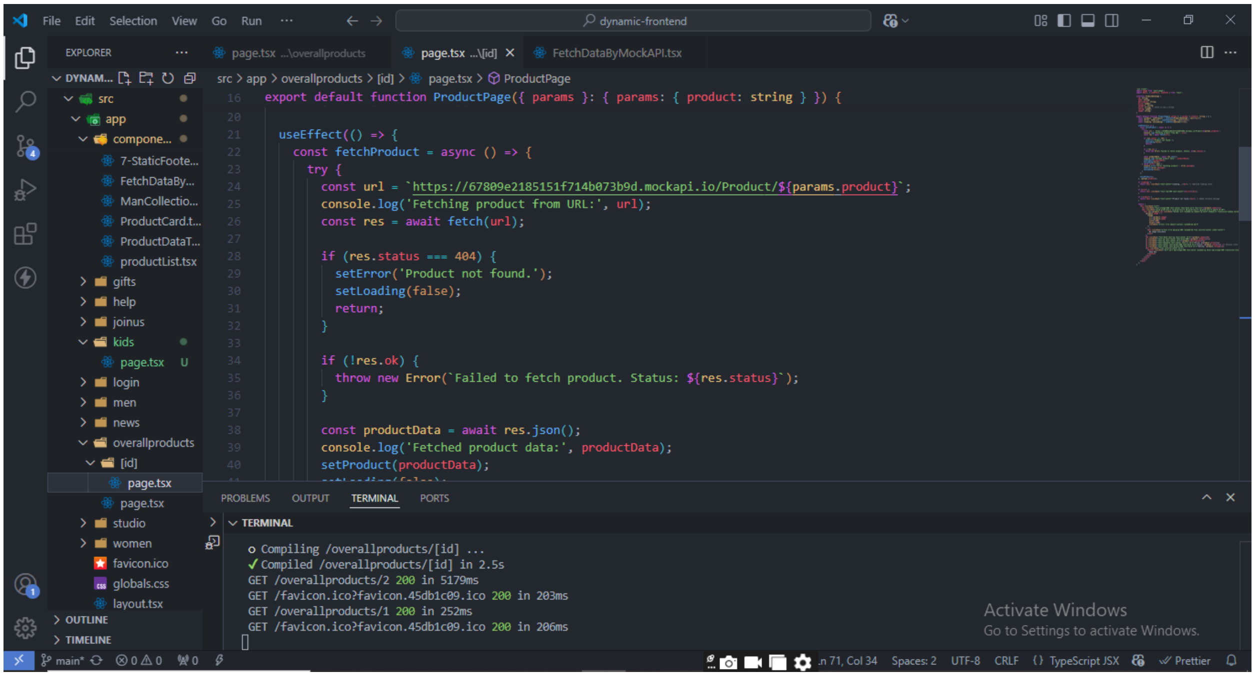
Error Handling Observations

1. **Network Failures**:
   - Display error messages like "Unable to connect to the server. Please check your internet connection."
   - Ensure the app offers retry options without crashing.
2. **Invalid or Missing Data**:
   - Validate user input with specific error messages (e.g., invalid email).
   - Notify users if APIs return incomplete data.
3. **Server Errors**:
   - Display user-friendly messages like "Something went wrong. Please try again later."
4. **Fallback UI**:
   - Use placeholders or retry buttons when API data is missing.



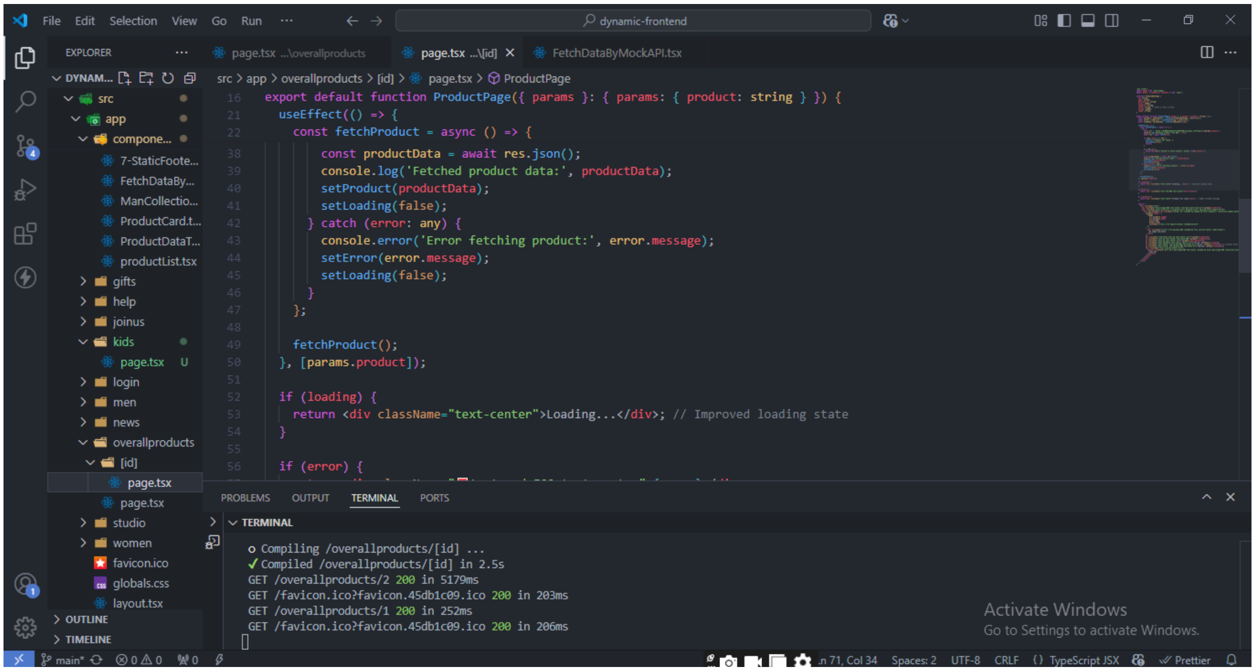
```tsx
const FetchDataByMockAPI = () => {

  useEffect(() => {
    const fetchData = async () => {
      try {
        const url = 'https://67809e2185151f714b073b9d.mockapi.io/Product';
        const data = await fetch(url);

        if (!data.ok) {
          throw new Error('Failed to fetch data');
        }

        const response = await data.json();
        setProducts(response);
      } catch (err) {
        setError('Oops.. Failed to fetch data...'); // Set error state
        console.log('Error:', err);
      }
    };

    fetchData();
  }, []);
```

```
 o Compiling /overallproducts/[id] ...
 ✓ Compiled /overallproducts/[id] in 2.5s
 GET /overallproducts/2 200 in 5179ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 203ms
 GET /overallproducts/1 200 in 252ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 206ms
```



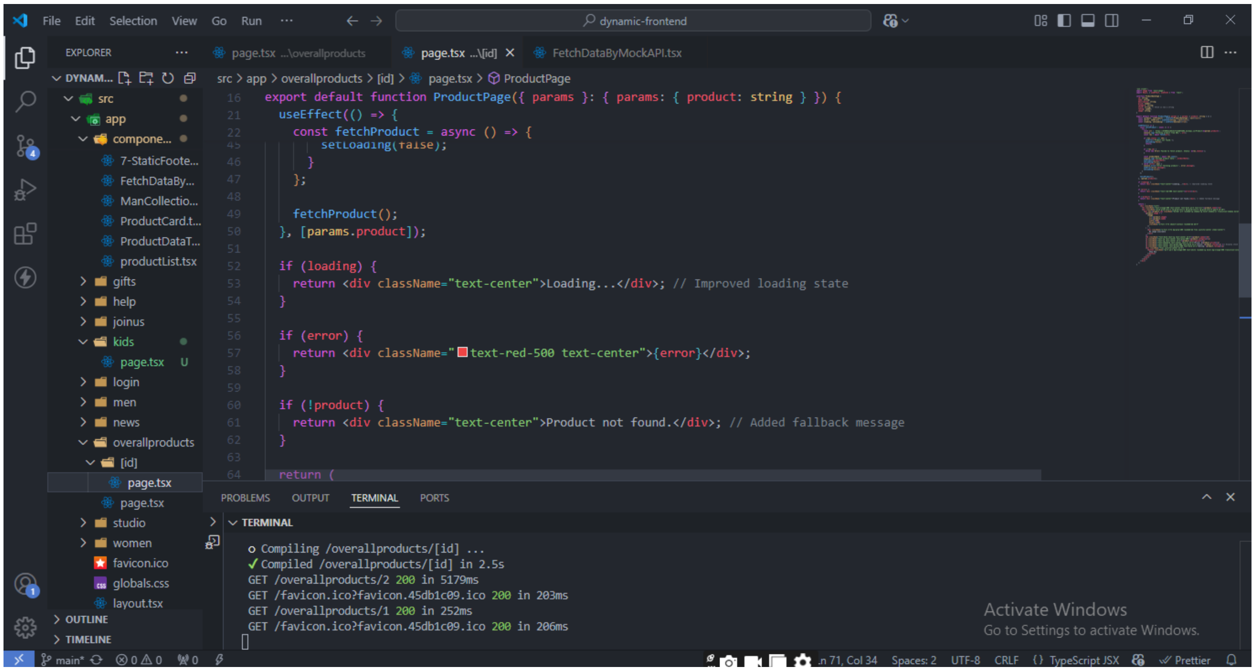
```tsx
export default function ProductPage({ params }: { params: { product: string } }) {

  useEffect(() => {
    const fetchProduct = async () => {
      try {
        const url = `https://67809e2185151f714b073b9d.mockapi.io/Product/${params.product}`;
        console.log('Fetching product from URL:', url);
        const res = await fetch(url);

        if (res.status === 404) {
          setError('Product not found.');
          setLoading(false);
          return;
        }

        if (!res.ok) {
          throw new Error(`Failed to fetch product. Status: ${res.status}`);
        }

        const productData = await res.json();
        console.log('Fetched product data:', productData);
        setProduct(productData);
```

```
 o Compiling /overallproducts/[id] ...
 ✓ Compiled /overallproducts/[id] in 2.5s
 GET /overallproducts/2 200 in 5179ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 203ms
 GET /overallproducts/1 200 in 252ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 206ms
```



```
export default function ProductPage({ params }: { params: { product: string } }) {
  useEffect(() => {
    const fetchProduct = async () => {
        const productData = await res.json();
        console.log('Fetched product data:', productData);
        setProduct(productData);
        setLoading(false);
      } catch (error: any) {
        console.error('Error fetching product:', error.message);
        setError(error.message);
        setLoading(false);
      }
    };

    fetchProduct();
  }, [params.product]);

  if (loading) {
    return <div className="text-center">Loading...</div>; // Improved loading state
  }

  if (error) {
```

```
o Compiling /overallproducts/[id] ...
✓ Compiled /overallproducts/[id] in 2.5s
GET /overallproducts/2 200 in 5179ms
GET /favicon.ico?favicon.45db1c09.ico 200 in 203ms
GET /overallproducts/1 200 in 252ms
GET /favicon.ico?favicon.45db1c09.ico 200 in 206ms
```



```
export default function ProductPage({ params }: { params: { product: string } }) {
  useEffect(() => {
    const fetchProduct = async () => {
        setLoading(false);
      }
    };

    fetchProduct();
  }, [params.product]);

  if (loading) {
    return <div className="text-center">Loading...</div>; // Improved loading state
  }

  if (error) {
    return <div className="text-red-500 text-center">{error}</div>;
  }

  if (!product) {
    return <div className="text-center">Product not found.</div>; // Added fallback message
  }

  return (
```

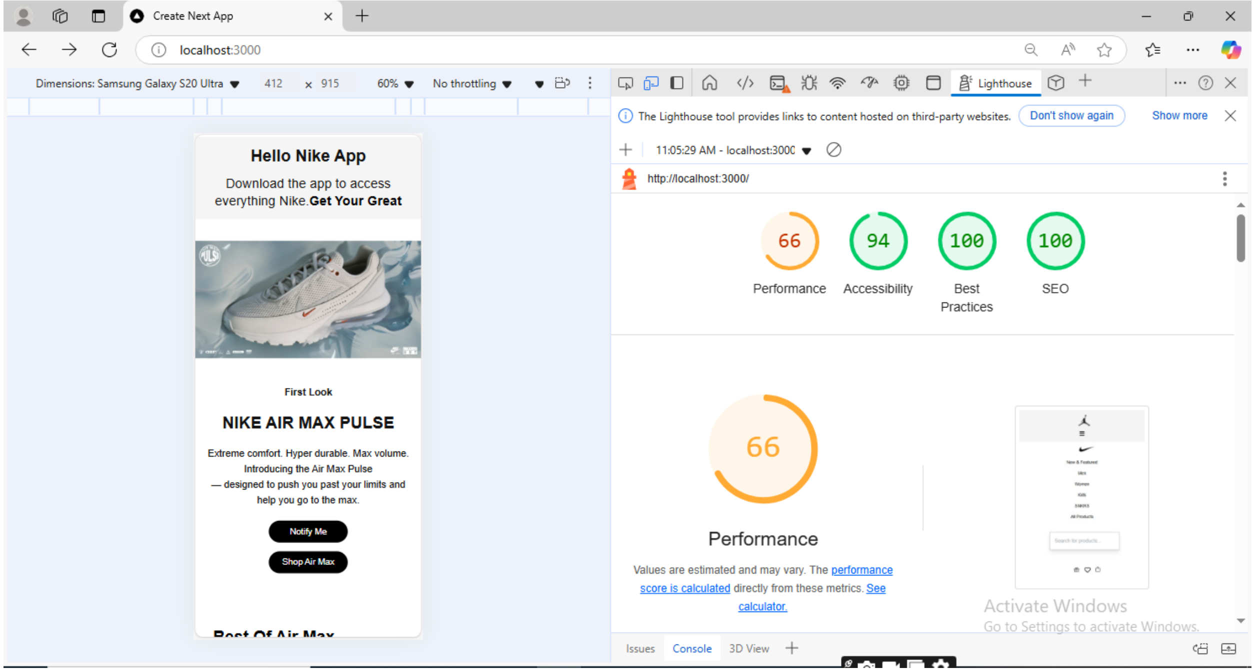
```
o Compiling /overallproducts/[id] ...
✓ Compiled /overallproducts/[id] in 2.5s
GET /overallproducts/2 200 in 5179ms
GET /favicon.ico?favicon.45db1c09.ico 200 in 203ms
GET /overallproducts/1 200 in 252ms
GET /favicon.ico?favicon.45db1c09.ico 200 in 206ms
```

Performance Optimization

Tools Used: Google Lighthouse

Performance:	**66**
Best Practices:	**100**

Accessibility:	94
SEO:	100



The Lighthouse tool provides links to content hosted on third-party websites.

http://localhost:3000/

Hello Nike App

Download the app to access everything Nike. **Get Your Great**

First Look

NIKE AIR MAX PULSE

Extreme comfort. Hyper durable. Max volume. Introducing the Air Max Pulse — designed to push you past your limits and help you go to the max.

Notify Me

Shop Air Max

Best Of Air Max

66

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

Cross-Browser and Device Testing

Browsers Tested: Chrome	Firefox, Safari, Edge
Devices Tested:	Tablet, Desktop,Mobile

Focus Areas:

- Responsive Design
- Consistent Navigation
- Verified accessibility features (keyboard navigation)

Security Measures

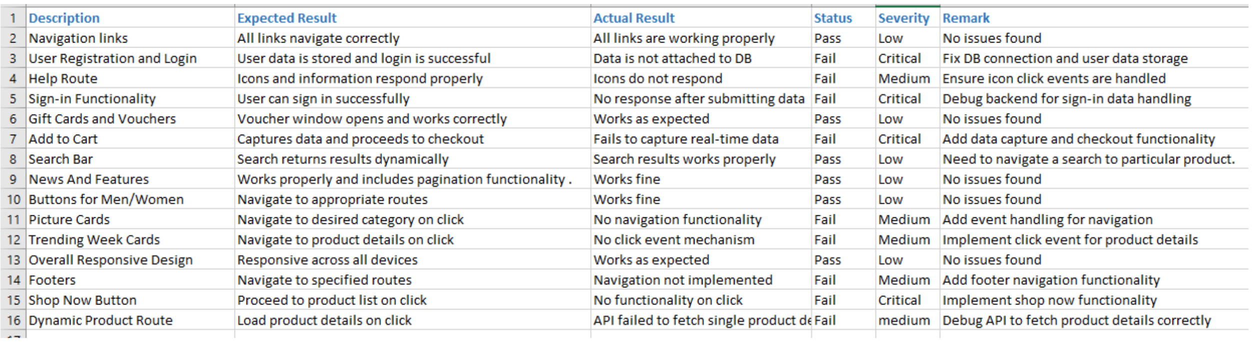
- Enforced HTTPS for all communications.
- Secured API keys using environment variables.
- Prevented injection attacks by validating user input and sanitizing data.

Documentation Updates

Compiled test results, resolutions, performance improvements, and security updates with supporting screenshots for future reference.

CSV for Functional Report

CSV for Functional Report is also attached.



1	Description	Expected Result	Actual Result	Status	Severity	Remark
2	Navigation links	All links navigate correctly	All links are working properly	Pass	Low	No issues found
3	User Registration and Login	User data is stored and login is successful	Data is not attached to DB	Fail	Critical	Fix DB connection and user data storage
4	Help Route	Icons and information respond properly	Icons do not respond	Fail	Medium	Ensure icon click events are handled
5	Sign-in Functionality	User can sign in successfully	No response after submitting data	Fail	Critical	Debug backend for sign-in data handling
6	Gift Cards and Vouchers	Voucher window opens and works correctly	Works as expected	Pass	Low	No issues found
7	Add to Cart	Captures data and proceeds to checkout	Fails to capture real-time data	Fail	Critical	Add data capture and checkout functionality
8	Search Bar	Search returns results dynamically	Search results works properly	Pass	Low	Need to navigate a search to particular product.
9	News And Features	Works properly and includes pagination functionality .	Works fine	Pass	Low	No issues found
10	Buttons for Men/Women	Navigate to appropriate routes	Works fine	Pass	Low	No issues found
11	Picture Cards	Navigate to desired category on click	No navigation functionality	Fail	Medium	Add event handling for navigation
12	Trending Week Cards	Navigate to product details on click	No click event mechanism	Fail	Medium	Implement click event for product details
13	Overall Responsive Design	Responsive across all devices	Works as expected	Pass	Low	No issues found
14	Footers	Navigate to specified routes	Navigation not implemented	Fail	Medium	Add footer navigation functionality
15	Shop Now Button	Proceed to product list on click	No functionality on click	Fail	Critical	Implement shop now functionality
16	Dynamic Product Route	Load product details on click	API failed to fetch single product de	Fail	medium	Debug API to fetch product details correctly