

Testing Report: Marketplace Ecommerce Website Development

Project Overview:

On Day 4 of the development of the Marketplace Ecommerce Website, the focus was on integrating key functionalities to enhance user experience and smooth operations. The primary objective was to test and validate the implemented features, ensuring proper interaction between frontend and backend, smooth data flow, and a seamless user experience.

Testing Tools Used:

1. Postman:

- **Purpose:** Used for API testing to verify the correctness and functionality of the backend endpoints.
- **Usage:** Ensured API calls were returning the correct data, and validated responses for all actions including fetching products, adding to the cart, submitting checkout, and handling voucher codes.

2. Manual Testing:

- **Purpose:** Test the functionality of features like dynamic routes, cart actions, pagination, and search bar through user interactions on the frontend.
- **Usage:** Performed actions like adding items to the cart, navigating between product pages, applying voucher codes, etc.

Key Features Tested:

1. Data Fetching & Dynamic Routes:

- **Objective:** Ensure product data is fetched correctly from the backend and displayed dynamically on the frontend.
- **Test Steps:**
 - Tested the GET request for fetching product data using Postman to ensure the backend responds with the correct list of products (with names, descriptions, prices, etc.).
 - Tested dynamic routing by clicking on a product on the product listing page and checking if the correct product details page is displayed.
- **Result:**
 - API returned the correct product data and displayed dynamically on the product listing page.

- Dynamic routing was successful, allowing users to view individual product details.

2. Add to Cart Functionality:

- **Objective:** Verify that users can add products to their cart and receive appropriate feedback.
- **Test Steps:**
 - Clicked "Add to Cart" on both the product listing and product detail pages.
 - Checked if SweetAlert2 pop-up confirmation appears.
 - Tested persistence of cart data using local storage after refreshing the page or navigating to different pages.
- **Result:**
 - The add-to-cart functionality worked smoothly, confirming the addition with SweetAlert2 pop-ups.
 - Cart data persisted successfully in local storage.

3. Gift Mechanism with Voucher System:

- **Objective:** Ensure the voucher code system is correctly applied and provides the expected outcomes.
- **Test Steps:**
 - Tested the entry of valid and invalid voucher codes during checkout.
 - Valid code should apply the gift mechanism; invalid code should show an error message.
- **Result:**
 - Valid voucher codes applied the gift mechanism successfully.
 - Invalid voucher codes displayed the correct error message.

4. CRUD Operations on Products:

- **Objective:** Ensure the admin can perform CRUD operations on products through API.
- **Test Steps:**
 - Performed POST, PUT, and DELETE requests using Postman to add, update, and delete products.
 - Verified that data changes were reflected on the frontend (product catalog).
- **Result:**
 - All CRUD operations worked as expected. Admin users can create, update, and delete products efficiently.

5. Pagination:

- **Objective:** Ensure pagination is functional and users can navigate through pages easily.
- **Test Steps:**
 - Navigated through product pages and tested pagination controls.
 - Verified that each page displays a subset of products, and clicking next/previous navigates correctly.
- **Result:**
 - Pagination worked as expected, allowing users to browse products in manageable chunks.

6. Search Bar Functionality:

- **Objective:** Ensure the search bar correctly filters products based on user input.
- **Test Steps:**
 - Entered various product names, categories, and keywords into the search bar.
 - Verified that search results were filtered dynamically.
 - Tested cases where no products matched the search query to verify proper error handling.
- **Result:**
 - The search bar functioned correctly, displaying filtered results in real-time.
 - Error handling for no matching products worked as expected.

7. News and Features Sections:

- **Objective:** Ensure the news and features sections display the correct promotional and product-related content.
- **Test Steps:**
 - Checked that the news and features sections displayed the correct offers, sales, and product arrivals.
- **Result:**
 - The sections were successfully updated with relevant news and product information.

8. User Authentication & Authorization:

- **Objective:** Ensure secure user login and proper access to personal data.
- **Test Steps:**
 - Logged in using valid credentials to verify secure access.

- Tested restricted access to protected routes for unauthenticated users.
- **Result:**
 - Authentication worked as expected, ensuring that users can securely access their profiles and order history.

9. Checkout & Cart Flow Enhancements:

- **Objective:** Verify the checkout process, ensuring smooth payment gateway integration and a user-friendly experience.
- **Test Steps:**
 - Added items to the cart, proceeded to checkout, and tested the flow from cart review to payment.
 - Tested the UI/UX of the checkout process to ensure smooth navigation.
- **Result:**
 - The checkout process was smooth, with UI/UX enhancements making the flow more intuitive.
 - Payment gateway integration was functional.

Test Results Summary:

Feature	Test Outcome	Comments
Data Fetching & Dynamic Routes	Passed	Data is fetched correctly and dynamic routes are functional.
Add to Cart Functionality	Passed	SweetAlert2 notifications and local storage persistence work well.
Gift Mechanism & Voucher System	Passed	Voucher codes applied gifts correctly; error handling worked for invalid codes.
CRUD Operations	Passed	Admin users can successfully manage products.
Pagination	Passed	Pagination worked seamlessly, with correct navigation.
Search Bar	Passed	Search function filters products in real-time with error handling.
News & Features Sections	Passed	Sections display the correct promotions and updates.
User Authentication & Authorization	Passed	Secure login and access control are working correctly.

Checkout & Cart Flow

Passed

Checkout process was smooth with enhanced UI/UX and functional payment gateway integration.

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:

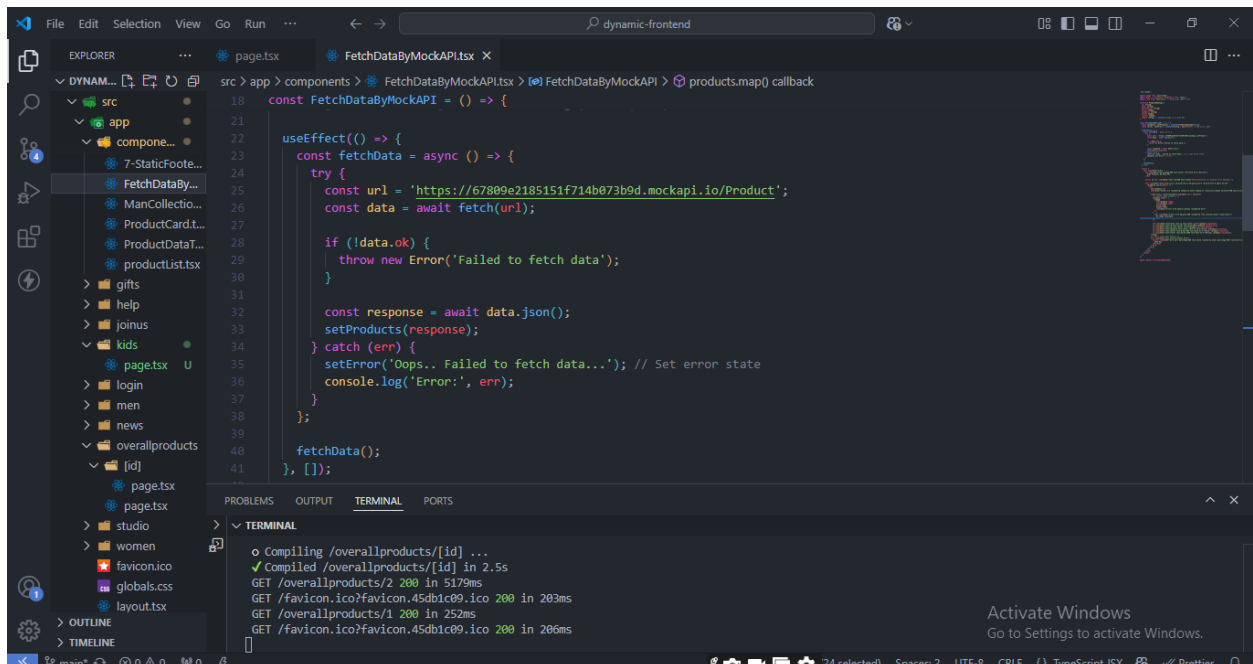
- Validate user input with specific error messages (e.g., invalid email).
- Notify users if APIs return incomplete data.

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.



```
File Edit Selection View Go Run ... dynamic-frontend
EXPLORER
  DYNAM...
  src
    app
      7-StaticFoote...
      FetchDataBy...
      ManCollectio...
      ProductCard.t...
      ProductDataT...
      productList.tsx
    gifts
    help
    joinus
    kids
    login
    men
    news
    overallproducts
    [id]
    page.tsx
    page.tsx
    studio
    women
    favicon.ico
    globals.css
    layout.tsx
  OUTLINE
  TIMELINE
  main

FetchDataByMockAPI.tsx
src > app > components > FetchDataByMockAPI.tsx > FetchDataByMockAPI > products.map() callback
18 const FetchDataByMockAPI = () => {
21
22   useEffect(() => {
23     const fetchData = async () => {
24       try {
25         const url = 'https://67809e2185151f714b073b9d.mockapi.io/Product';
26         const data = await fetch(url);
27
28         if (!data.ok) {
29           throw new Error('Failed to fetch data');
30         }
31
32         const response = await data.json();
33         setProducts(response);
34       } catch (err) {
35         setError('Oops.. Failed to fetch data...'); // Set error state
36         console.log('Error:', err);
37       }
38     };
39     fetchData();
40   }, []);
41 }
```

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

Activate Windows
Go to Settings to activate Windows.

```
File Edit Selection View Go Run ... dynamic-frontend
EXPLORER
src > app > overallproducts > [id] > page.tsx
  src
  app
  compone...
    7-StaticFoote...
    FetchDataBy...
    ManCollectio...
    ProductCard.t...
    ProductDataT...
    productList.tsx
  gifts
  help
  joinus
  kids
  pagetx U
  login
  men
  news
  overallproducts
    [id]
      page.tsx
      page.tsx
  studio
  women
  favicon.ico
  globals.css
  layout.tsx
  OUTLINE
  TIMELINE

page.tsx
src > app > overallproducts > [id] > page.tsx
export default function ProductPage({ params }: { params: { product: string } }) {
  useEffect(() => {
    const fetchProduct = async () => {
      try {
        const url = `https://67809e218515f714b073b9d.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.');
```

```
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
```

Activate Windows
Go to Settings to activate Windows.

```
File Edit Selection View Go Run ... dynamic-frontend
EXPLORER
src > app > overallproducts > [id] > page.tsx
  src
  app
  compone...
    7-StaticFoote...
    FetchDataBy...
    ManCollectio...
    ProductCard.t...
    ProductDataT...
    productList.tsx
  gifts
  help
  joinus
  kids
  pagetx U
  login
  men
  news
  overallproducts
    [id]
      page.tsx
      page.tsx
  studio
  women
  favicon.ico
  globals.css
  layout.tsx
  OUTLINE
  TIMELINE

page.tsx
src > app > overallproducts > [id] > page.tsx
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);
    }
  });

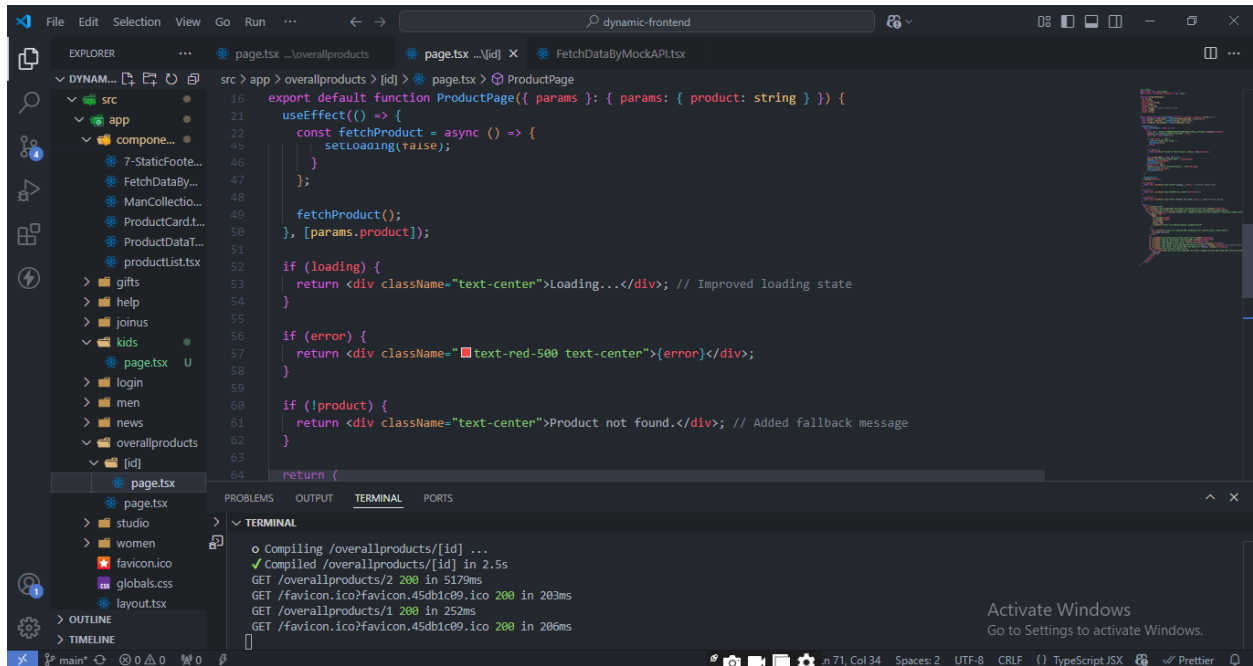
  fetchProduct();
}, [params.product]);

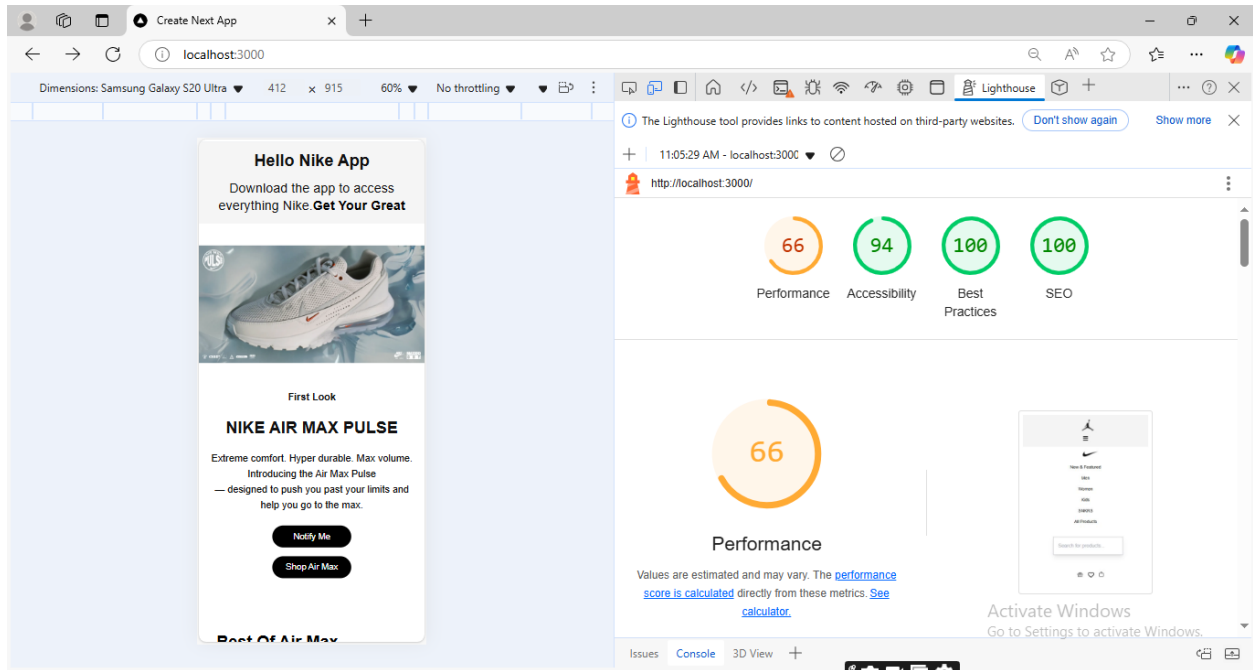
if (loading) {
  return <div className="text-center">Loading...</div>; // Improved loading state
}

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
```

Activate Windows
Go to Settings to activate Windows.





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

1	Test Case ID	Test Case ID	Test Case ID	Test Case ID	Test Case ID	Test Case ID	Test Case ID
2	TC001	Verify login functionality	1. Open login page. 2. Enter valid username and password. 3. Click on login button.	User is logged in successfully and redirected to the homepage	User is logged in successfully and redirected to the homepage	Passed	High
3	TC002	Verify login with invalid credentials	1. Open login page. 2. Enter invalid username and password. 3. Click on login button.	User sees an error message "Invalid username or password."	User sees an error message "Invalid username or password."	Passed	High
4	TC003	Check product details page	1. Go to product listing page. 2. Click on a product.	Product details page is displayed with correct name, price, and description	Product details page is displayed with correct name, price, and description	Passed	High
5	TC004	Add product to cart	1. Go to product details page. 2. Click 'Add to Cart' button.	Product is added to the cart and cart icon updates with item count	Product added to the cart and cart icon update well.	Passed	High
6	TC005	Check cart page	1. Go to cart page.	Cart page displays all added products with correct details.	Cart page displays all added products with correct details	Passed	High
7	TC006	Verify checkout process	1. Go to cart page. 2. Click on checkout button. 3. Enter shipping and payment details.	User is directed to confirmation page with correct details and total price.	User is directed to confirmation page with correct price	Passed	High
8	TC007	Verify responsiveness on mobile	1. Open website on mobile device.	Website layout adjusts correctly for mobile view.	Website layout adjusts correctly for mobile view.	Passed	High
9	TC008	Verify sign-up process	1. Open sign-up page. 2. Enter valid details. 3. Click sign-up button.	User is signed up successfully and redirected to login page.	User is signed up successfully and redirected to login page.	Passed	High
10	TC009	Check product search	1. Go to homepage. 2. Type a product name in the search bar. 3. Press enter.	Relevant products are shown in search results.	Relevant products are shown in search results.	Passed	High
11	TC010	Verify email notifications	1. Complete a purchase. 2. Check email inbox for confirmation.	User receives a confirmation email with order details.	User does not receive a confirmation email	Failed	Low