GLOBAL MART ONLINE SHOP

DEV: MAHNOOR GHAFFAR

Hackathon Day 05

TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINRMENT

- Functional Testing
- Error Handling

1.Functional Testing:

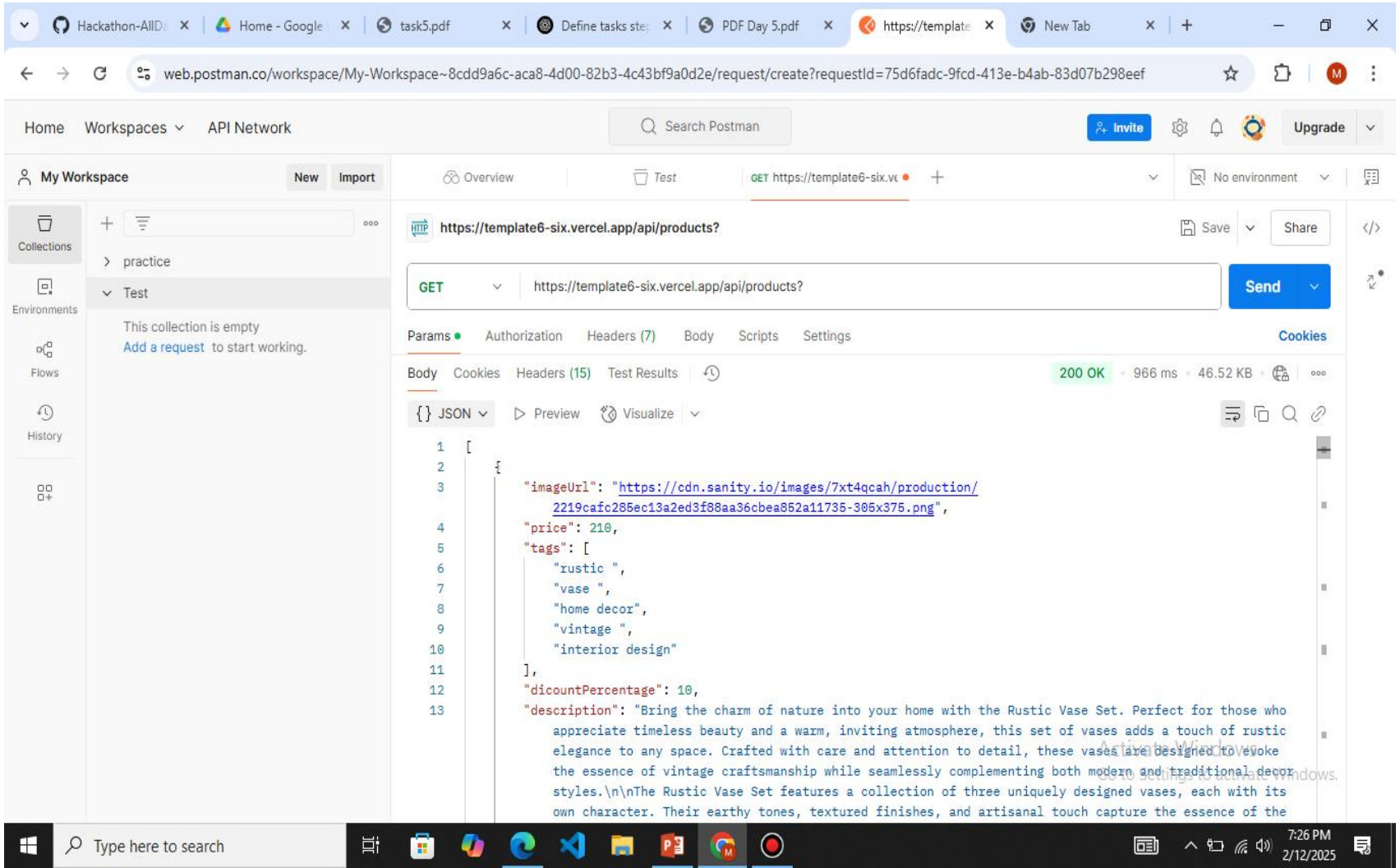
Test Core Features:

Test Case ID	Description	Test Steps	Expected Results	Actual Result	Status	Severity Level	Remarks
FT-001	**navigation links** (Ensure correct redirections).	Navigate to any page check all links are working correctly	All pages are visible without any error	As expected all links are correctly working	passed	low	Links are working correctly
FT-002	Ensure that all products are display correctly	Navigate to the shop page Check if all product images, names,price are displayed correctly.	All products are visible without any distortion.	As expected all products are correctly displayed	passed	low	Product displayed correct.
FT-003	Add a product to the cart.	1. Click on the "Add to Cart" button for a product. Add, remove and update quantity.	Product is added removed updated successfully.	Cart update as expected.	Pass	medium	Functionality Is working good.

Test Case ID	Description	Test Steps	Expected Results	Actual Result	Status	Severity Level	Remarks
FT-004	**Verify that dynamic product details page load correctly**	Click on a product card. Check if the product detail page loads with the correct information.	The page should show correct details.	Product detail page loads correctly	passed	low	Dynamic routing work as expected

Postman

Used for Testing Api Response



Error Handling

- **Product Listing**

- The `ProductCard` component handles API errors by catching them in `getdata`, returning an empty array if fetching fails. This ensures a fallback UI with a "No products available" message is displayed. Fallback images are used if `imageUrl` is missing, maintaining functionality. Once the API is restored, it fetches and displays data from Sanity.

```
async function getdata(): Promise<Product[]> {
  try {
    const fetchData = await client.fetch<Product[]>(
      `*[_type=="product"]{
        _id,
        title,
        "imageUrl": productImage.asset->url,
        price,
        tags,
        dicountPercentage,
        description,
        isNew
      }`
    );
    return fetchData;
  } catch (error) {
    console.error("Failed to fetch data:", error);
    return []; // Return an empty array in case of an error
  }
}
```

Product Details page

1.Using a try-catch block in the fetchProduct function to catch API errors.

2.Setting an error state if the API fails or the product is not found.

3.Displaying fallback images if imageUrl is missing.

4.Showing loading states or error messages (Loading..., Error: {error}, or No product data found) based on the state (loading, error, or product).

5.Ensuring the UI remains functional even if the API fails, with fallback mechanisms in place.

```
  (data.length > 0) {
        setProduct(data[0]);
        setSelectedSize(data[0].availableSizes ?
data[0].availableSizes[0] : null);
      } else {
        setError('Product not found');
      }
    } catch (err) {
      setError('Failed to fetch product data');
    } finally {
      setLoading(false);
    }
  };
```

Cart Page

The checkout page validates form fields before placing an order, highlighting errors in red if fields are empty.
If validation fails,Swal.fire alert the user. Other wise it confirms the order, clears the cart, and redirects to home.

```
else {
      Swal.fire({
        title: "Error!",
        text: "Please fill in all the
required fields.",
        icon: "error",
        confirmButtonText: "OK",
      });
    }
  };
```

```
const validateForm = () => {
    const errors = {
      firstName: !formValues.firstName,
      lastName: !formValues.lastName,
      address: !formValues.address,
      city: !formValues.city,
      zipCode: !formValues.zipCode,
      phone: !formValues.phone,
      email: !formValues.email,
 };
    setFormErrors(errors);
    return Object.values(errors).every((error) => !error);
  };
 const handlePlaceOrder = () => {
    if (validateForm()) {
      Swal.fire({
        title: "Order Placed!",
        text: "Your order has been successfully placed.",
        icon: "success",
        confirmButtonText: "OK",
      }).then(() => {
        dispatch(clearCart()); // Clear the cart after placing the order
        router.push("/"); // Navigate to the home page or
confirmation page}); }
```

THANK YOU

mahnoorghaffar9@gmail.com

Global Mart