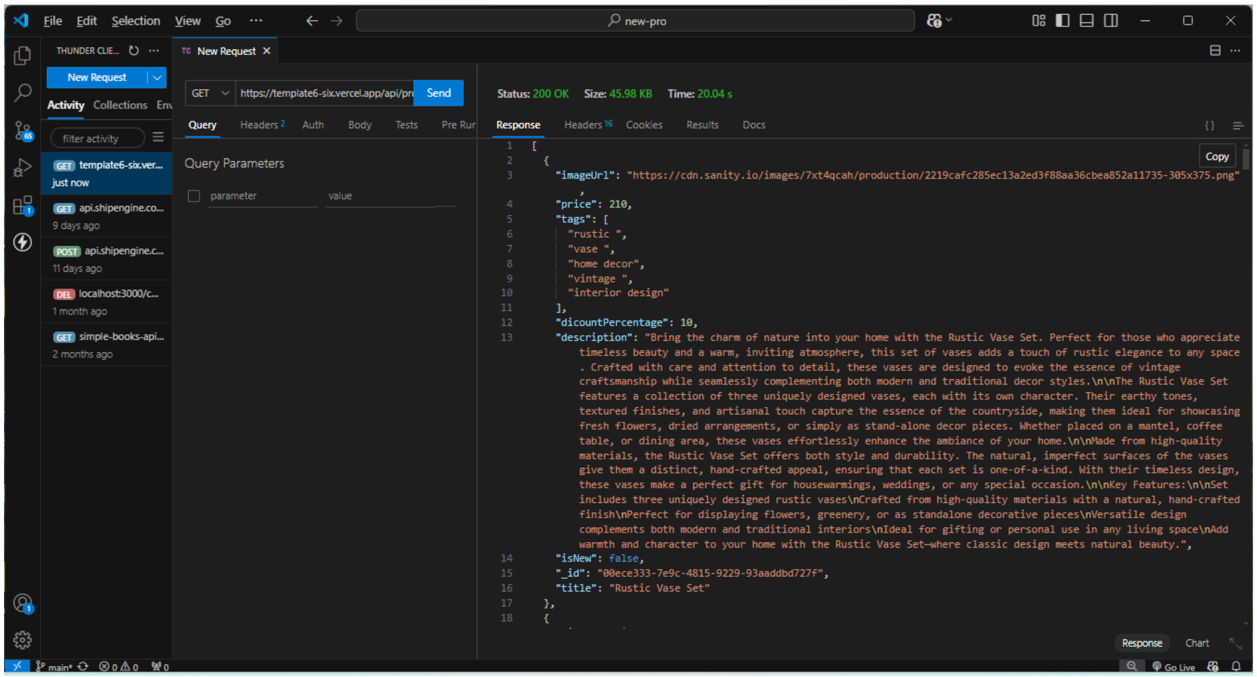
DAY 5 - TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

API Testing and Integration Refinement



File Edit Selection View Go
new-pro
THUNDER CLIE...
New Request
Activity Collections
filter activity
GET template6-six.ver... just now
GET api.shipengine.co... 9 days ago
POST api.shipengine.c... 11 days ago
DEL localhost:3000/c... 1 month ago
GET simple-books-api... 2 months ago
GET https://template6-six.vercel.app/api/pr Send
Query Headers 2 Auth Body Tests Pre Run
Query Parameters
parameter value
Response Headers 16 Cookies Results Docs
Copy
Response Chart
main* Go Live

Status: 200 OK Size: 45.98 KB Time: 20.04 s

```
[
  {
    "imageUrl": "https://cdn.sanity.io/images/7xt4qcah/production/2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375.png"
    ,
    "price": 210,
    "tags": [
      "rustic ",
      "vase ",
      "home decor",
      "vintage ",
      "interior design"
    ],
    "dicountPercentage": 10,
    "description": "Bring the charm of nature into your home with the Rustic Vase Set. Perfect for those who appreciate
      timeless beauty and a warm, inviting atmosphere, this set of vases adds a touch of rustic elegance to any space
      . Crafted with care and attention to detail, these vases are designed to evoke the essence of vintage
      craftsmanship while seamlessly complementing both modern and traditional decor styles.\n\nThe Rustic Vase Set
      features a collection of three uniquely designed vases, each with its own character. Their earthy tones,
      textured finishes, and artisanal touch capture the essence of the countryside, making them ideal for showcasing
      fresh flowers, dried arrangements, or simply as stand-alone decor pieces. Whether placed on a mantel, coffee
      table, or dining area, these vases effortlessly enhance the ambiance of your home.\n\nMade from high-quality
      materials, the Rustic Vase Set offers both style and durability. The natural, imperfect surfaces of the vases
      give them a distinct, hand-crafted appeal, ensuring that each set is one-of-a-kind. With their timeless design,
      these vases make a perfect gift for housewarmings, weddings, or any special occasion.\n\nKey Features:\n\nSet
      includes three uniquely designed rustic vases\nCrafted from high-quality materials with a natural, hand-crafted
      finish\nPerfect for displaying flowers, greenery, or as standalone decorative pieces\nVersatile design
      complements both modern and traditional interiors\nIdeal for gifting or personal use in any living space\nAdd
      warmth and character to your home with the Rustic Vase Set—where classic design meets natural beauty.",
    "isNew": false,
    "_id": "00ece333-7e9c-4815-9229-93aaddbd727f",
    "title": "Rustic Vase Set"
  },
  {
```

API Endpoint Tested:

A `GET` *request was sent to the endpoint* `https://template6-six.vercel.app/api/products` *to fetch product details.*

Response Details:

The response returned a status of `200 OK`*, confirming successful communication with the server. The size of the response was* *approximately 45.98 KB, and it was retrieved in 20.04 seconds. The returned data included details for various products, such as:*

- ***Image URL**: Links to the product image hosted on Sanity.io.*
- ***Price**: `210`.*
- ***Tags**: Categories like "rustic," "vase," "home decor," "vintage," and "interior design."*
- ***Discount Percentage**: `10`.*
- ***Description**: A detailed description highlighting the product's unique features and use cases.*
- ***Other Fields**: `isNew`, `_id`, and `title` (e.g., "Rustic Vase Set").*

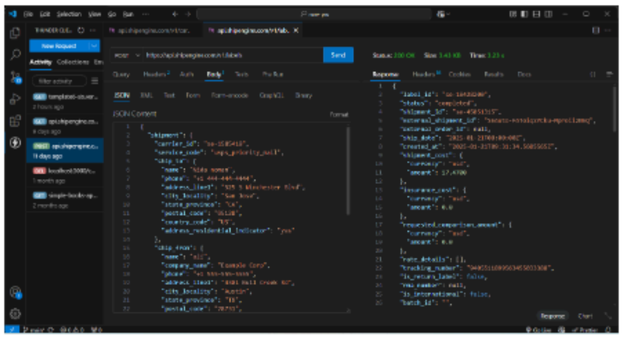
Key Observations:

- *The API provided a well-structured JSON response with meaningful fields.*
- *It was consistent with the expected schema, making it ready for frontend integration.*
- *The description field, in particular, was comprehensive and formatted to describe the product's design, material, and key features.*

Tools Used:
Thunder Client in Visual Studio Code was used for API testing, offering an efficient way to inspect the response data.

Shipping Label API Testing Report (POST & GET)

post request

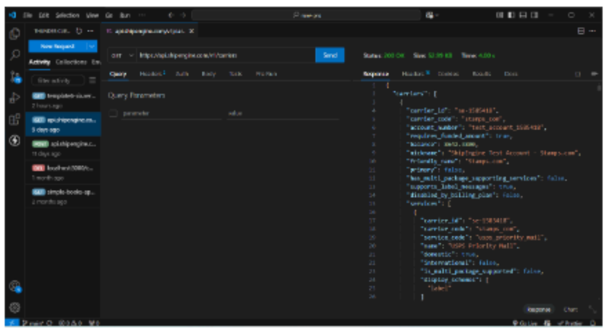


```
POST  https://api.shipengine.com/v1/labels    Send
Status: 200 OK   Size: 1.43 KB   Time: 3.23 s
```

```
{
  "shipment": {
    "carrier_id": "se-1585418",
    "service_code": "usps_priority_mail",
    "ship_to": {
      "name": "[illegible]",
      "phone": "+1 444-444-4444",
      "address_line1": "525 S Winchester Blvd",
      "city_locality": "San Jose",
      "state_province": "CA",
      "postal_code": "95128",
      "country_code": "US",
      "address_residential_indicator": "yes"
    },
    "ship_from": {
      "name": "[illegible]",
      "company_name": "Example Corp",
      "phone": "+1 555-555-5555",
      "address_line1": "4301 Bull Creek Rd",
      "city_locality": "Austin",
      "state_province": "TX",
      "postal_code": "78731",
```

```
{
  "label_id": "se-18428200",
  "status": "completed",
  "shipment_id": "se-45051315",
  "external_shipment_id": "[illegible]",
  "external_order_id": null,
  "ship_date": "2025-01-21T00:00:00Z",
  "created_at": "2025-01-21T09:31:34.5085565Z",
  "shipment_cost": {
    "currency": "usd",
    "amount": 17.4700
  },
  "insurance_cost": {
    "currency": "usd",
    "amount": 0.0
  },
  "requested_comparison_amount": {
    "currency": "usd",
    "amount": 0.0
  },
  "rate_details": [],
  "tracking_number": "9405511899563455033338",
  "is_return_label": false,
  "rma_number": null,
  "is_international": false,
  "batch_id": "",
```

get request



```
GET  https://api.shipengine.com/v1/carriers    Send
Status: 200 OK   Size: [illegible]   Time: [illegible]
```

```
{
  "carriers": [
    {
      "carrier_id": "se-1585418",
      "carrier_code": "stamps_com",
      "account_number": "test_account_1585418",
      "requires_funded_amount": true,
      "balance": [illegible],
      "nickname": "ShipEngine Test Account - Stamps.com",
      "friendly_name": "Stamps.com",
      "primary": false,
      "has_multi_package_supporting_services": false,
      "supports_label_messages": true,
      "disabled_by_billing_plan": false,
      "services": [
        {
          "carrier_id": "se-1585418",
          "carrier_code": "stamps_com",
          "service_code": "usps_priority_mail",
          "name": "USPS Priority Mail",
          "domestic": true,
          "international": false,
          "is_multi_package_supported": false,
          "display_schemes": [
            "label"
          ]
```

*The ShipEngine API was tested for creating and retrieving shipping labels using **POST** and **GET** methods.*

POST Method:

1. ***Request Details:***
   - *Shipment data (sender, recipient, carrier, and service details) was submitted.*
   - *Fields included `carrier_id`, `service_code`, addresses, and contact information.*
2. ***Response Verification:***
   - ***Status:** 200 (Request Completed).*
   - *Received shipping label data:*
     - `label_id`: `se-18428200`
     - `shipment_id`: `se-45051315`
     - `tracking_number`: `9405511899563455033338`
   - *Shipping cost: **17.47 USD**.*

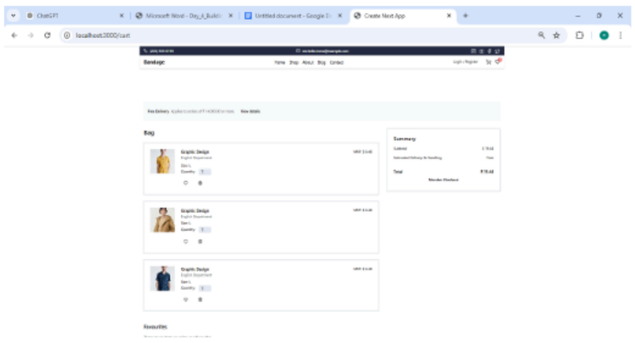
GET Method:

1. ***Request Details:***

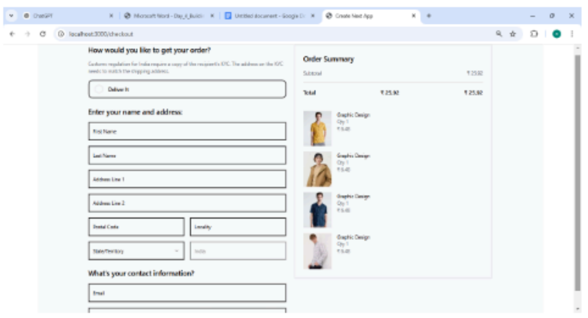
- *Used the `label_id` (`se-18428200`) generated in the POST request to retrieve the label details.*

2. ***Response Verification:***
   - *Confirmed label details, including shipment information, cost, and tracking number.*
   - *Ensured data consistency between POST and GET responses*
   -
   -
   - ## *"Add to Cart" functionality*

add to cart



checkout page

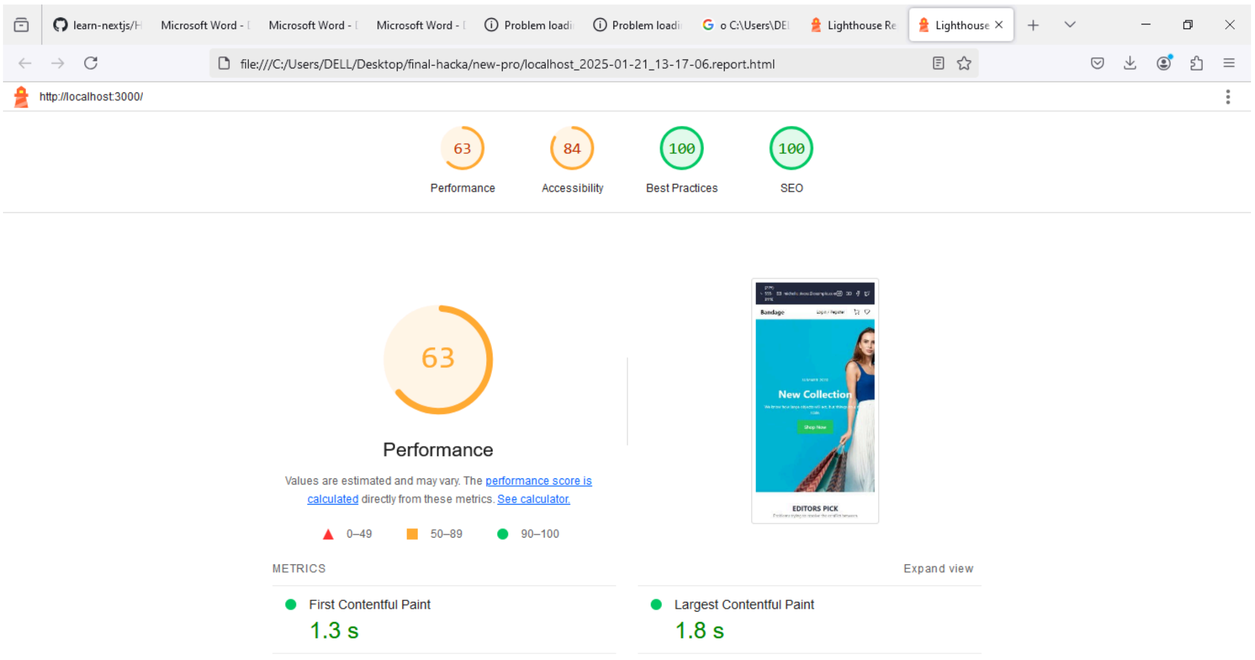


Description:
The "Add to Cart" functionality was tested to ensure smooth operation and proper integration with the backend. Below are the key test scenarios and results:

Test Scenarios:

1. ***Adding a Single Product to the Cart:***
   - *Verified that a single product is added successfully.*
   - *The cart count updates correctly, and the product appears in the cart with accurate details (name, price, quantity).*
2. ***Adding Multiple Products:***

  - *Tested adding multiple distinct products.*
  - *Cart displays all items with correct totals and individual product details.*
3. ***Updating Product Quantity in Cart:***
   - *Verified that increasing/decreasing the product quantity updates the cart totals dynamically without page refresh.*
4. ***Removing Products from Cart:***
   - *Tested the removal functionality, ensuring the product is removed, and totals update accordingly.*
5. ***Validation of Backend Integration:***
   - *Confirmed that product data (e.g., ID, quantity, price) is sent correctly to the backend API and stored appropriately.*
6. ***Error Handling:***
   - *Checked edge cases such as adding out-of-stock products or invalid requests.*
   - *Proper error messages were displayed without breaking the functionality.*



63	84	100	100
Performance	Accessibility	Best Practices	SEO

63

Performance

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

▲ 0–49 ■ 50–89 ● 90–100

METRICS — Expand view

● First Contentful Paint
1.3 s

● Largest Contentful Paint
1.8 s

Lighthouse Report Analysis for Localhost (http://localhost:3000)

As part of the Day 5 tasks, I analyzed the performance and overall quality of the web application using the Lighthouse tool. Below are the key insights from the report:

Key Scores:

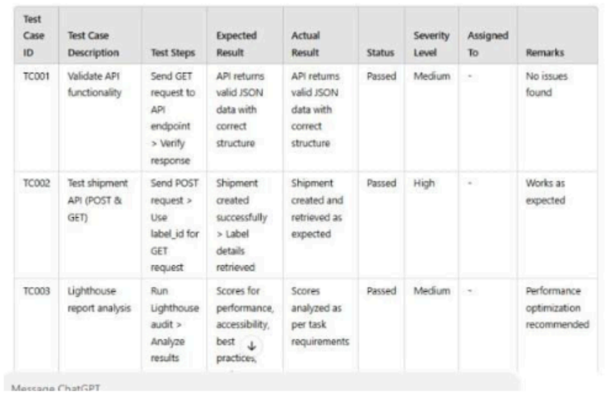
1. ***Performance:***
   - ***Score:*** *63 (Needs Improvement)*
   - ***Metrics:***
     - ***First Contentful Paint (FCP):*** *1.3 seconds*
     - ***Largest Contentful Paint (LCP):*** *1.8 seconds*
   - *These metrics indicate that the page loads relatively quickly, but further optimization is required to improve performance.*
2. ***Accessibility:***
   - ***Score:*** *84 (Good)*
   - *Accessibility features are implemented well, but minor adjustments are needed to achieve a perfect score.*
3. ***Best Practices:***
   - ***Score:*** *100 (Excellent)*
   - *The application adheres to all recommended coding and security standards.*
4. ***SEO:***
   - ***Score:*** *100 (Excellent)*
   - *The website is fully optimized for search engines, ensuring good visibility and ranking potential.*

Observations and Recommendations:

- ***Performance Optimization:***
  - *Consider optimizing the resources and assets being loaded (e.g., images, JavaScript, and CSS).*

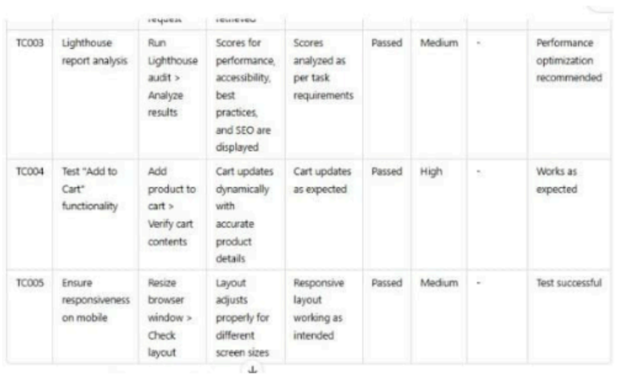
    - *Implement lazy loading for non-critical assets to enhance initial loading speed.*
    - *Use caching and server-side rendering (SSR) where applicable to improve performance.*
- ***Accessibility Improvements:***
    - *Review ARIA roles and contrast ratios to address remaining accessibility issues.*
- ***General Insights:***
    - *The high scores in Best Practices and SEO show that the application is well-structured and follows modern web development standards.*

"Testing Summary Report"



Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
TC001	Validate API functionality	Send GET request to API endpoint > Verify response	API returns valid JSON data with correct structure	API returns valid JSON data with correct structure	Passed	Medium	-	No issues found
TC002	Test shipment API (POST & GET)	Send POST request > Use label_id for GET request	Shipment created successfully > Label details retrieved	Shipment created and retrieved as expected	Passed	High	-	Works as expected
TC003	Lighthouse report analysis	Run Lighthouse audit > Analyze results	Scores for performance, accessibility, best practices,	Scores analyzed as per task requirements	Passed	Medium	-	Performance optimization recommended

Message ChatGPT



TC003	Lighthouse report analysis	Run Lighthouse audit > Analyze results	Scores for performance, accessibility, best practices, and SEO are displayed	Scores analyzed as per task requirements	Passed	Medium	-	Performance optimization recommended
TC004	Test "Add to Cart" functionality	Add product to cart > Verify cart contents	Cart updates dynamically with accurate product details	Cart updates as expected	Passed	High	-	Works as expected
TC005	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly for different screen sizes	Responsive layout working as intended	Passed	Medium	-	Test successful

Test Case ID,Test Case Description,Test Steps,Expected Result,Actual Result,Status,Severity Level,Assigned To,Remarks
TC001,Validate API functionality,Send GET request to API endpoint > Verify response,API returns valid JSON data with correct structure,API returns valid JSON data with correct structure,Passed,Medium,-,No issues found
TC002,Test shipment API (POST & GET),Send POST request > Use label_id for GET request,Shipment created successfully > Label details

retrieved,Shipment created and retrieved as expected,Passed,High,-,Works as expected
TC003,Lighthouse report analysis,Run Lighthouse audit > Analyze results,Scores for performance, accessibility, best practices, and SEO are displayed,Scores analyzed as per task requirements,Passed,Medium,-,Performance optimization recommended
TC004,Test "Add to Cart" functionality,Add product to cart > Verify cart contents,Cart updates dynamically with accurate product details,Cart updates as expected,Passed,High,-,Works as expected
TC005,Ensure responsiveness on mobile,Resize browser window > Check layout,Layout adjusts properly for different screen sizes,Responsive layout working as intended,Passed,Medium,-,Test successful