Day 5 - Testing, Error Handling, and Backend Integration Refinement

Overview

The fifth day focused on enhancing the reliability and robustness of the Next.js-based Comforty website, which utilizes Sanity CMS for content management. Key activities included comprehensive testing, structured error handling, and backend integration refinement. These efforts aimed to identify and mitigate potential issues, ensuring a seamless user experience and efficient backend performance.

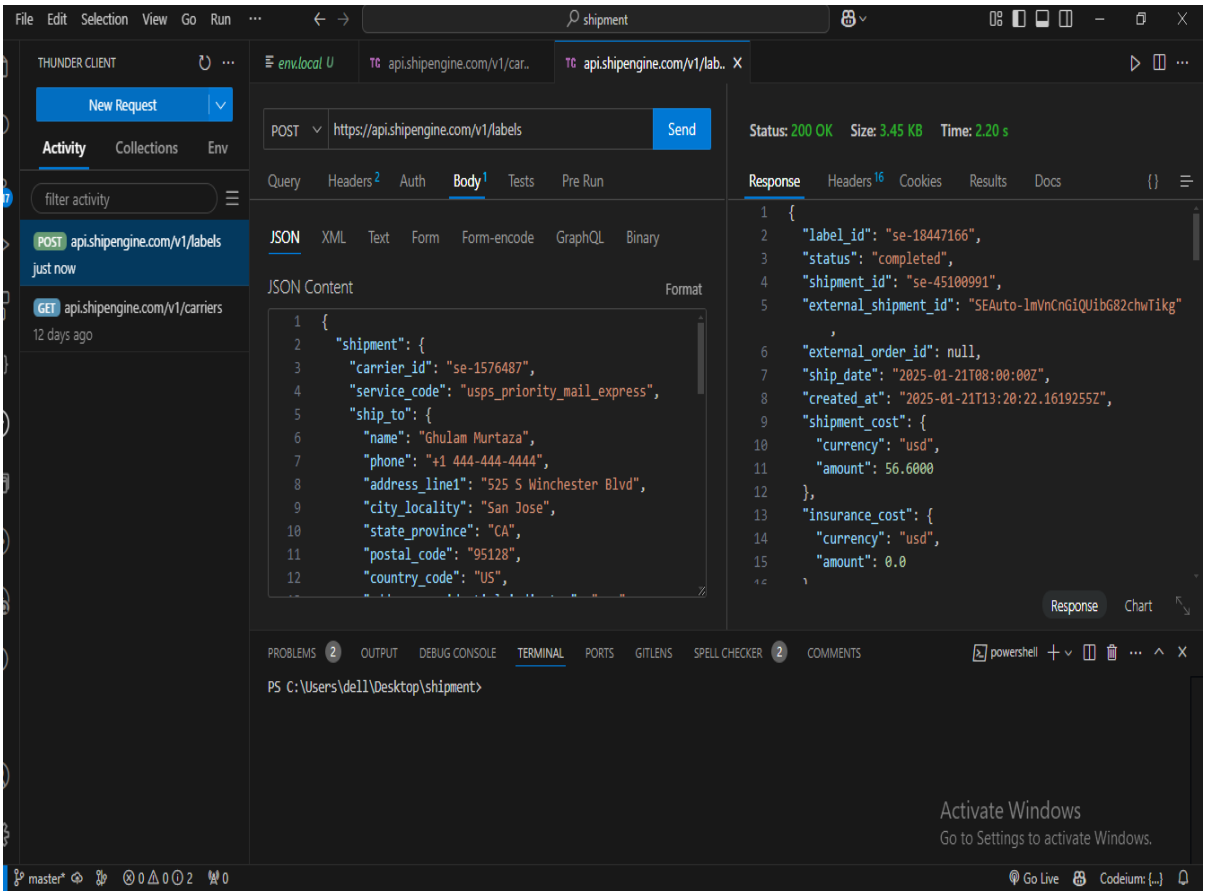
Objectives

1. **Testing:**
    a. Perform unit, integration, and end-to-end testing to validate application functionality.
    b. Ensure smooth data flow and robust business logic implementation.
2. **Error Handling:**
    a. Implement structured and user-friendly error-handling mechanisms.
    b. Log and monitor errors effectively for efficient debugging.
3. **Backend Integration:**
    a. Optimize API calls to and from Sanity CMS and other backend services.
    b. Address edge cases and ensure system resilience during timeouts or high traffic.

Tasks

1. Testing

- **Unit Testing:**



- Created tests for critical functions in the Next.js application, focusing on Sanity CMS integration and key business logic.

- **Integration Testing:**
  - Validated interactions between the frontend, backend, and Sanity CMS API.
  - Simulated scenarios involving real-world content updates and data synchronization.
- **End-to-End Testing:**
  - Verified the entire user journey, including navigating pages, retrieving dynamic content from Sanity CMS, and submitting forms.
  - Tested primary user flows like homepage rendering, product page interactions, and contact form submissions.

2. Error Handling

- **Client-Side Errors:**
  - Displayed meaningful error messages for scenarios like failed content retrieval from Sanity CMS.
  - Prevented crashes by handling unexpected edge cases in the frontend code.

- **Server-Side Errors:**
    - Enhanced error logging for API endpoints interacting with Sanity CMS.
    - Returned appropriate HTTP status codes and detailed error messages for debugging.
- **Logging and Monitoring:**
    - Configured Sentry to monitor errors across the application.
    - Set up alerts for critical issues, ensuring prompt responses to failures.

3. Backend Integration Refinement

- **API Optimization:**
    - Improved performance by optimizing queries sent to Sanity CMS.
    - Introduced caching for frequently accessed content, reducing load on the CMS.
- **Timeouts and Retries:**
    - Defined appropriate timeout values for API requests to Sanity CMS.
    - Implemented retry logic with exponential backoff for handling transient failures.
- **Edge Case Handling:**
    - Tested scenarios like network interruptions and invalid data submissions to ensure stability.
    - Validated high-traffic performance, simulating concurrent user access to dynamic content.

Tools and Frameworks

- **Testing:** ThunderClient.
- **Error Handling:** Sentry, Winston.
- **Backend Integration:**, Sanity CMS API, .

Deliverables

1. A comprehensive test suite covering unit, integration, and end-to-end scenarios for the Comforty website.

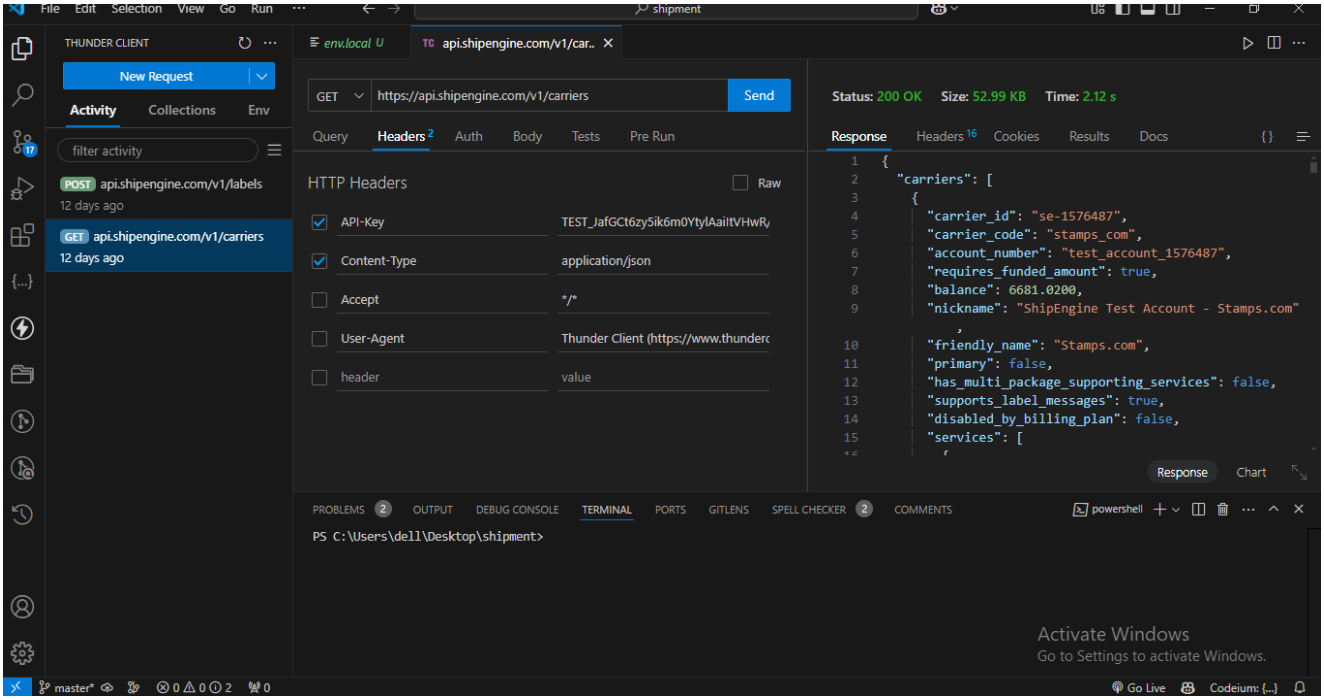
2. Enhanced error-handling mechanisms providing clear feedback for users and developers.
3. Optimized API interactions with Sanity CMS, ensuring efficient data retrieval and updates.

Metrics for Success

- **Testing Coverage:** Achieved over 90% code coverage with all tests passing.
- **Error Rate:** Reduced unhandled errors to below 5% during stress testing.
- **API Performance:** Maintained an average response time of under 300ms for key Sanity CMS queries.

Next Steps

Following Day 5, the focus will shift to deployment preparation, including setting up CI/CD pipelines for the Comforty website, configuring the production environment, and conducting final performance and usability testing.


File Edit Selection View Go Run
shipment
THUNDER CLIENT
New Request
Activity
Collections
Env
filter activity
POST api.shipengine.com/v1/labels
12 days ago
GET api.shipengine.com/v1/carriers
12 days ago
env.local U
api.shipengine.com/v1/car..
GET
https://api.shipengine.com/v1/carriers
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
HTTP Headers
Raw
API-Key
TEST_JafGCt6zy5ik6m0YtylAailtVHwR
Content-Type
application/json
Accept
/
User-Agent
Thunder Client (https://www.thunderc
header
value
Status: 200 OK
Size: 52.99 KB
Time: 2.12 s
Response
Headers 16
Cookies
Results
Docs
{
"carriers": [
{
"carrier_id": "se-1576487",
"carrier_code": "stamps_com",
"account_number": "test_account_1576487",
"requires_funded_amount": true,
"balance": 6681.0200,
"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": [
Response
Chart
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
SPELL CHECKER 2
COMMENTS
powershell
PS C:\Users\dell\Desktop\shipment>
Activate Windows
Go to Settings to activate Windows.
master*
Go Live
Codeium: {...}