Step 1: Test Functionality

1. **Define Objectives**:
   Start by understanding which feature or functionality you are going to test. For example, if you are working on a Contact Management System, you need to test functionalities like adding, updating, deleting, finding, and listing contacts.
2. **Write Test Cases**:
   Create test cases for each functionality, for example:
   - When adding a contact, the system should successfully add the contact if valid data is provided.
   - When invalid data is entered, an error message should be shown.
   - When updating a contact, changes should be reflected correctly in the system.

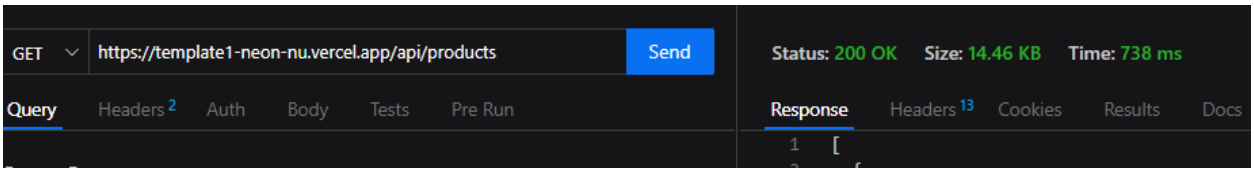
Test planning is all about defining the scope and creating a strategy for testing the functionalities thoroughly.

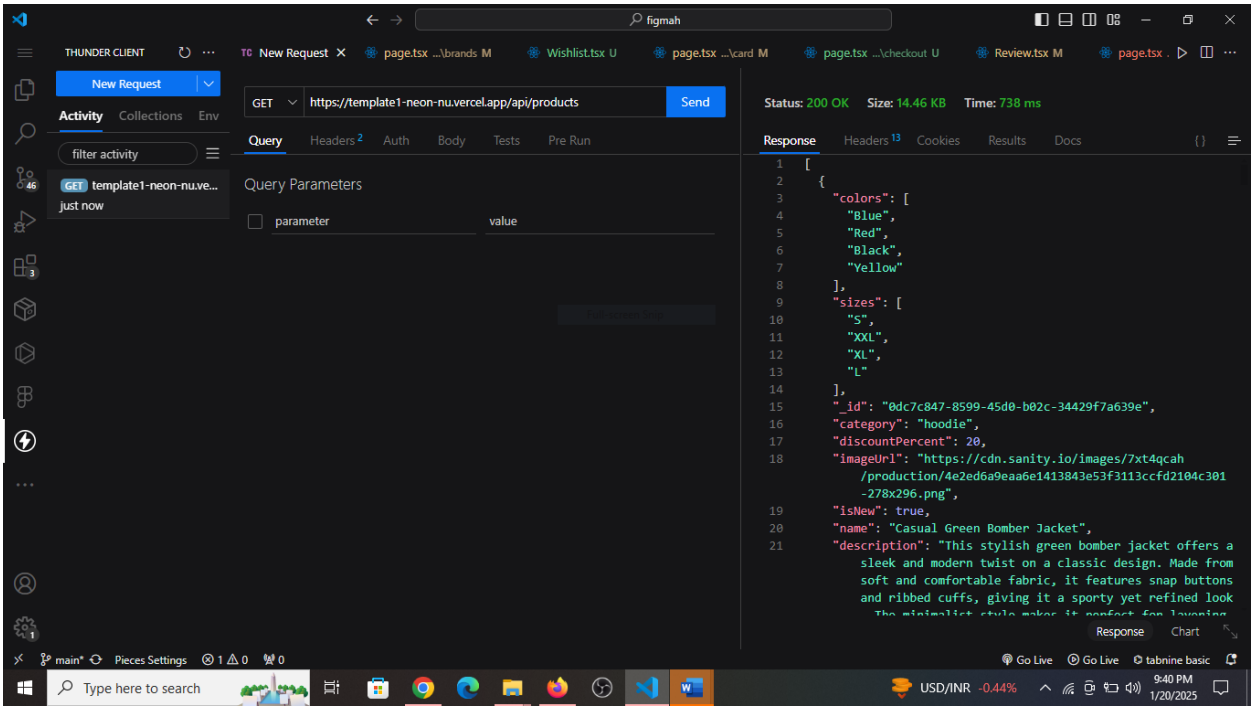
Step 2: Test Execution in Postman

1. **Run the API Requests**:

Execute the defined test cases for each API endpoint in Postman:

- **POST**: Send data to add a contact, and check for a success response.
- **GET**: Fetch data for an existing contact and verify the correct details are returned.
- **PUT**: Update a contact's information and verify that the changes are reflected.
- **DELETE**: Send a delete request and confirm that the contact is removed.

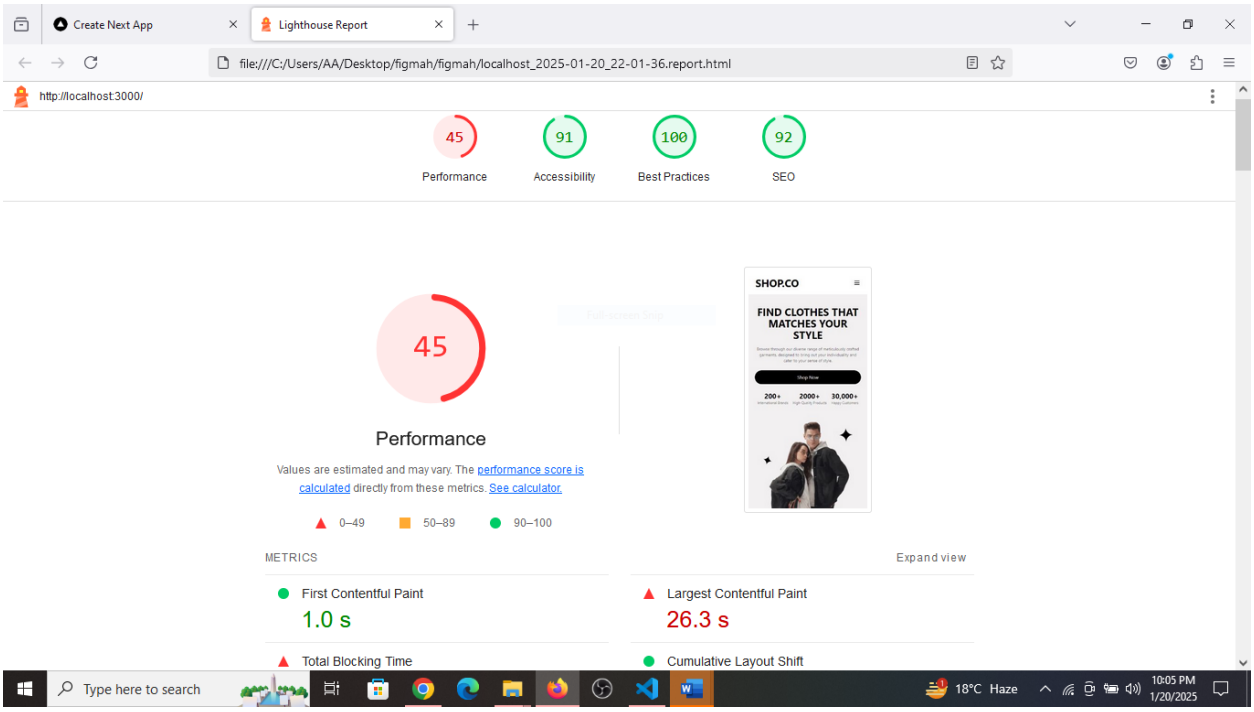


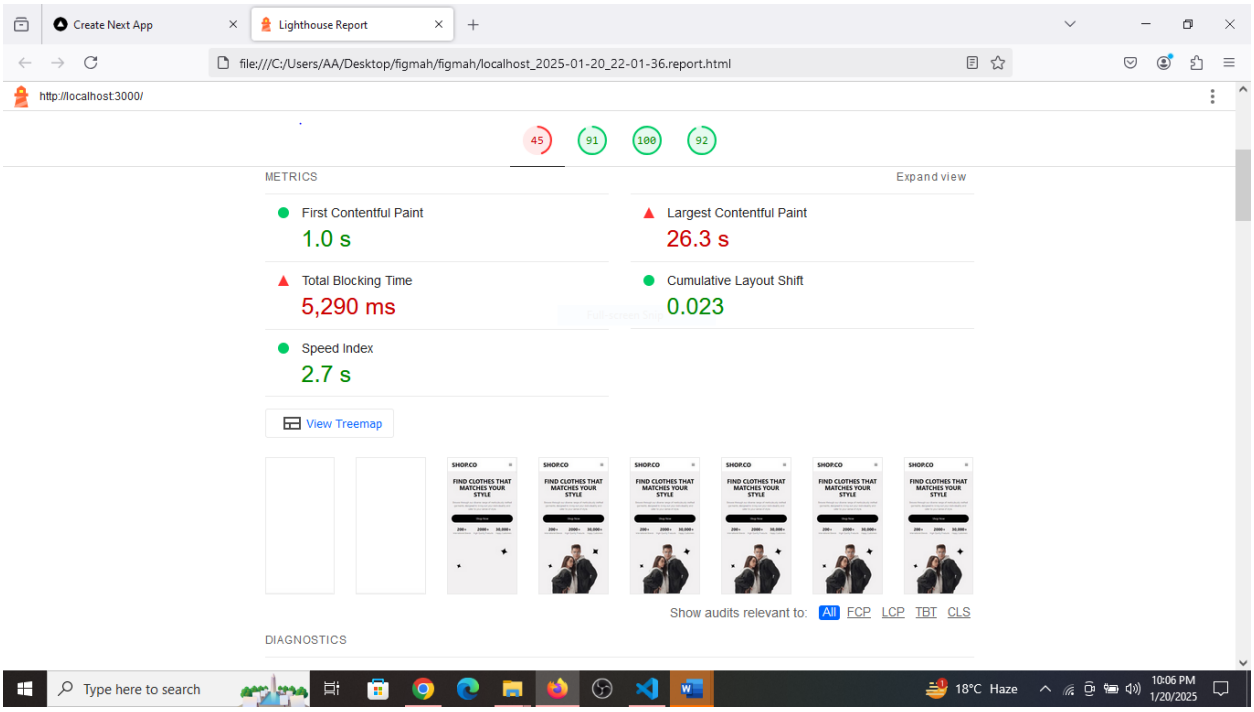


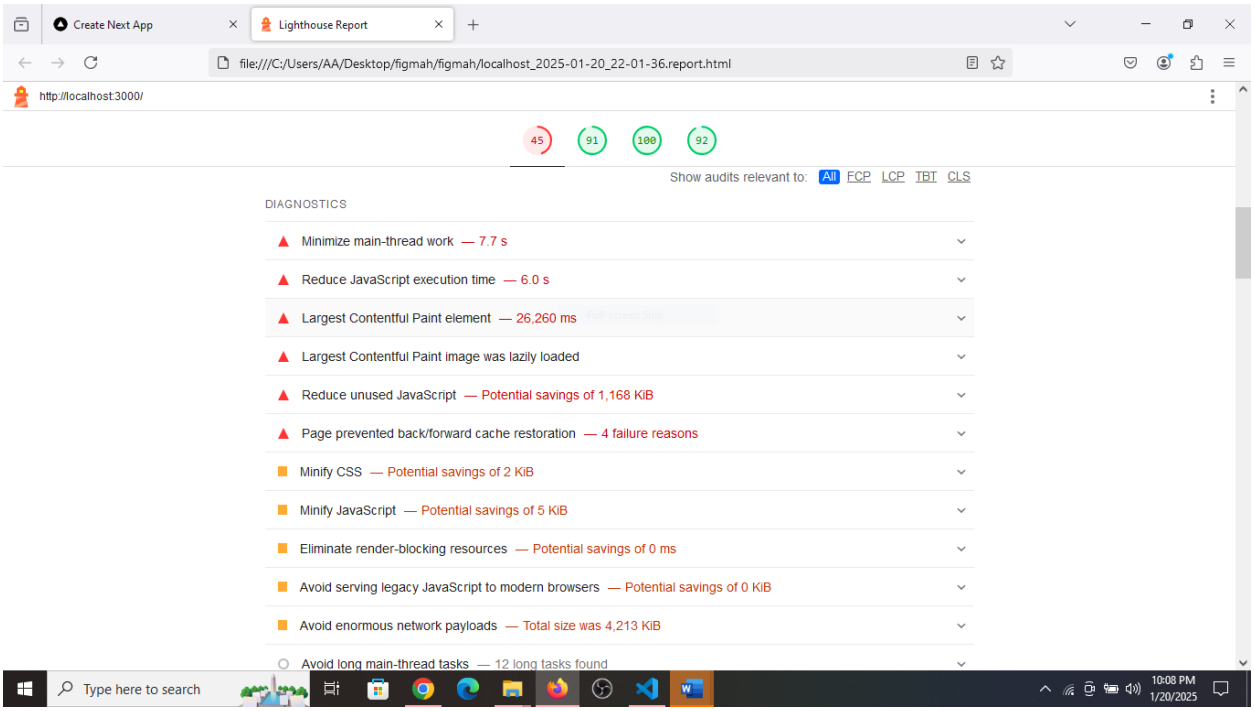
Step 3: Error Handling in Postman API Testing

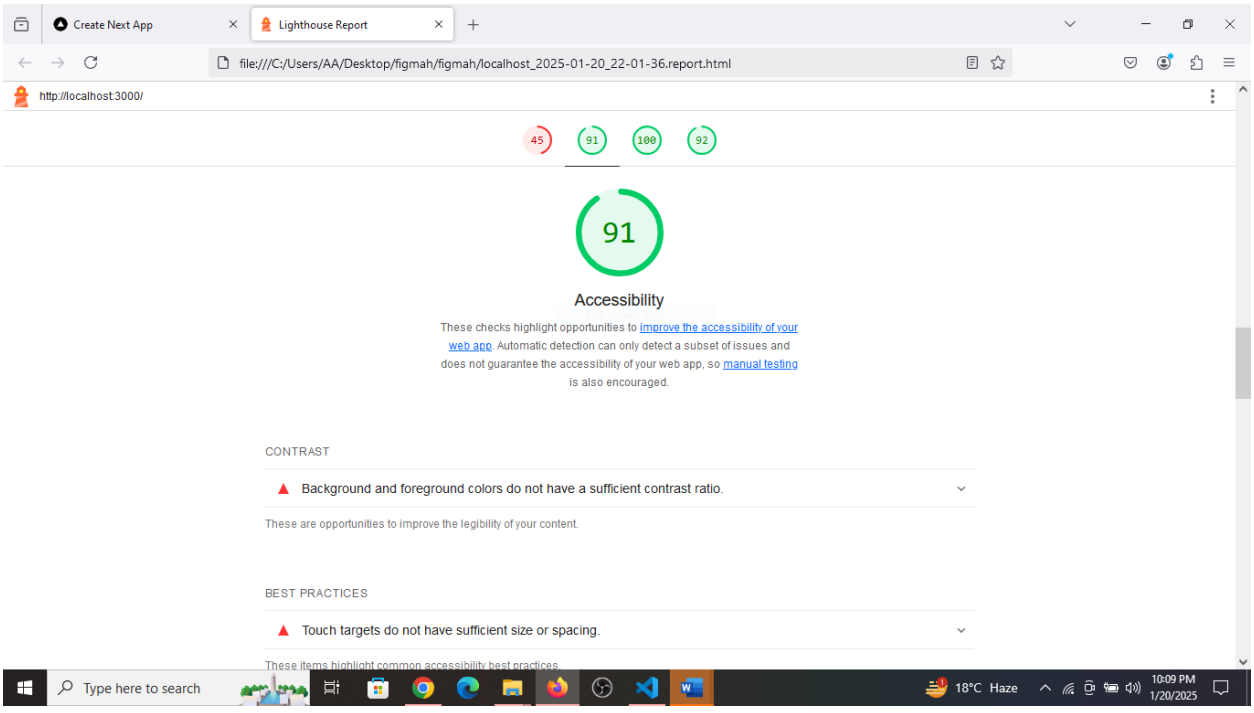
1. **Identify Common Errors**:
   When testing APIs, anticipate potential errors that can occur, such as:
   - **400 Bad Request**: Invalid input data (e.g., missing required fields).
   - **404 Not Found**: When the requested resource doesn't exist.
   - **500 Internal Server Error**: A server-side error indicating issues on the backend.
   - **401 Unauthorized**: When the user doesn't have proper authentication or permissions.
2. **Test with Invalid Data**:
   To handle errors, test the API with invalid or missing data:
   - **Missing Parameters**: For a POST request, try submitting a request without required fields (e.g., missing contact name or email).
   - **Invalid Data Types**: Test APIs with incorrect data formats, like sending a string where a number is expected.
   - **Unauthorized Requests**: For GET, PUT, or DELETE requests, test with missing or incorrect authentication tokens to check for 401 Unauthorized errors.
3. **Error Response Validation**:
   Ensure that the error messages and status codes returned by the API are correct. For instance:
   - **400 Bad Request** should return a meaningful error message, like "Missing required field: name".
   - **404 Not Found** should indicate that the resource could not be found.
   - **500 Internal Server Error** should ideally log the error message or issue for developers.

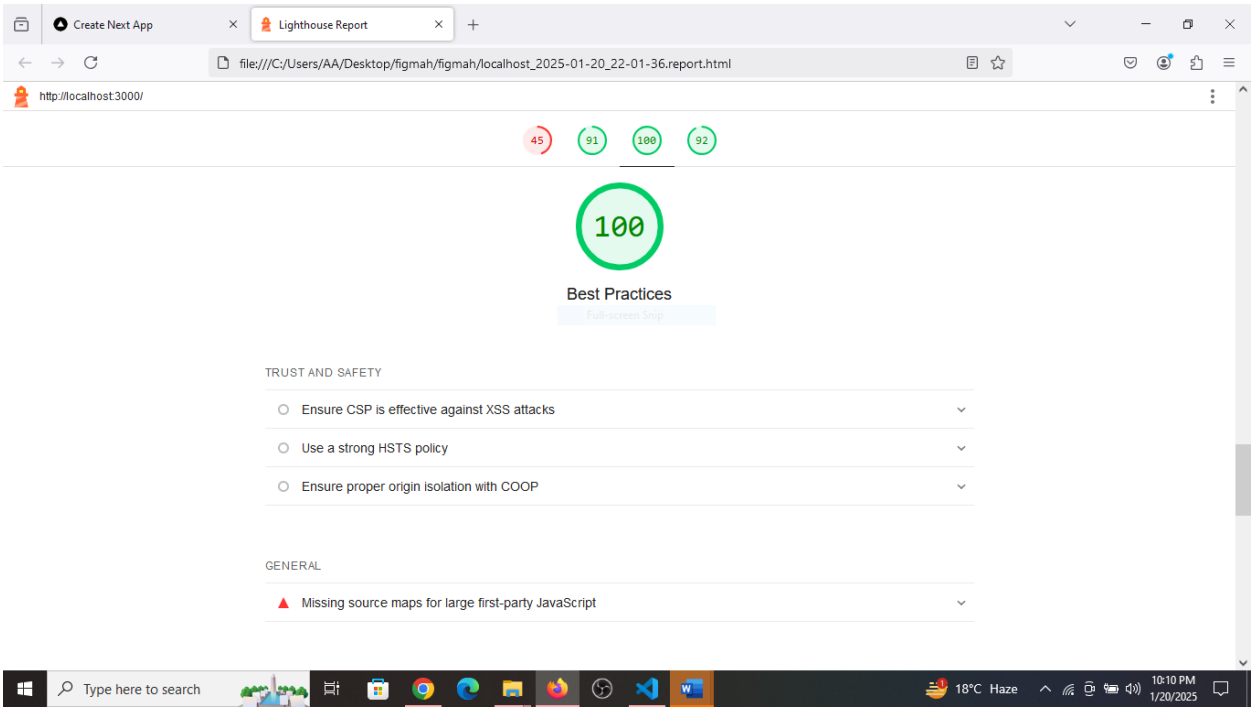
Step 4: **Performance**



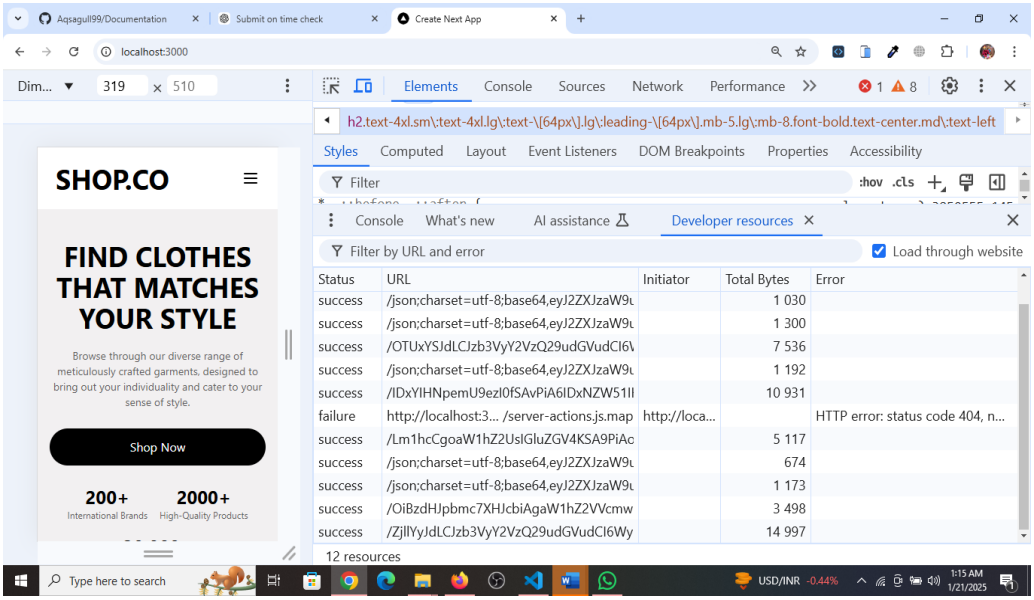



Create Next App
Lighthouse Report
file:///C:/Users/AA/Desktop/figmah/figmah/localhost_2025-01-20_22-01-36.report.html
http://localhost:3000/
45
91
100
92
Show audits relevant to: All FCP LCP TBT CLS
DIAGNOSTICS
Minimize main-thread work — 7.7 s
Reduce JavaScript execution time — 6.0 s
Largest Contentful Paint element — 26,260 ms
Largest Contentful Paint image was lazily loaded
Reduce unused JavaScript — Potential savings of 1,168 KiB
Page prevented back/forward cache restoration — 4 failure reasons
Minify CSS — Potential savings of 2 KiB
Minify JavaScript — Potential savings of 5 KiB
Eliminate render-blocking resources — Potential savings of 0 ms
Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB
Avoid enormous network payloads — Total size was 4,213 KiB
Avoid long main-thread tasks — 12 long tasks found
Type here to search
10:08 PM
1/20/2025

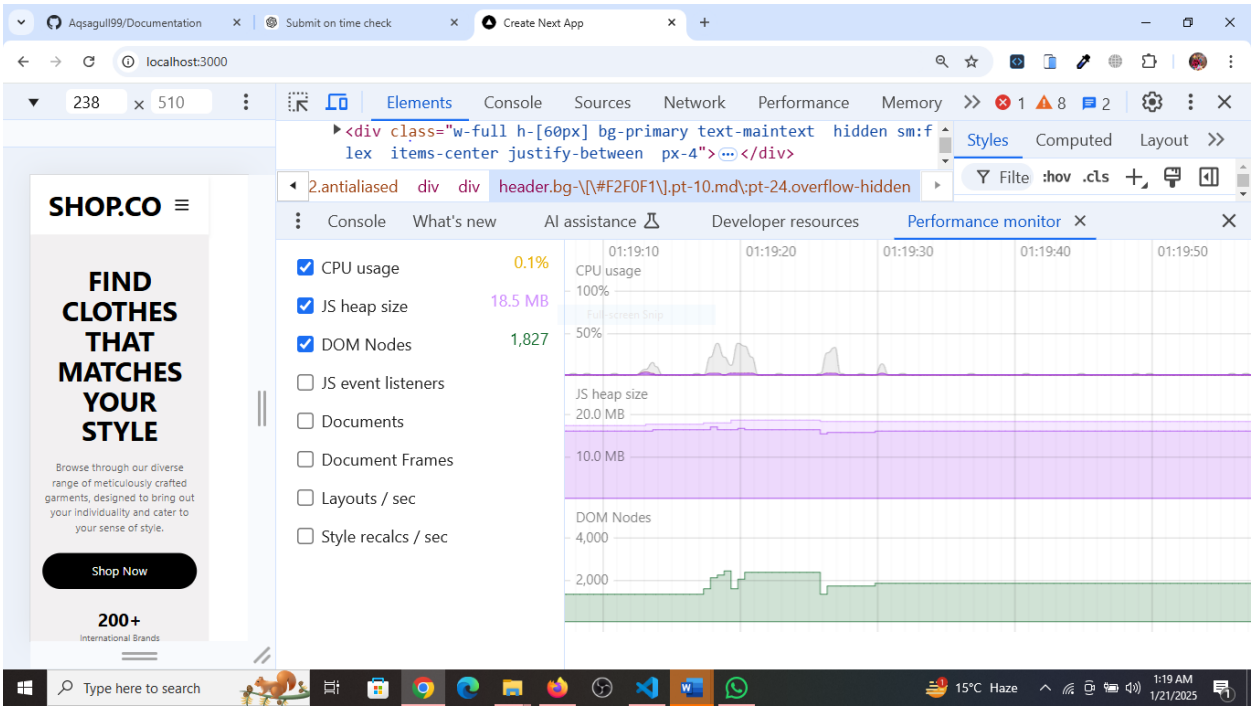

Create Next App
Lighthouse Report
file:///C:/Users/AA/Desktop/figmah/figmah/localhost_2025-01-20_22-01-36.report.html
http://localhost:3000/
45
91
100
92
91
Accessibility
These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.
CONTRAST
Background and foreground colors do not have a sufficient contrast ratio.
These are opportunities to improve the legibility of your content.
BEST PRACTICES
Touch targets do not have sufficient size or spacing.
Type here to search
18°C Haze
10:09 PM
1/20/2025


Create Next App
Lighthouse Report
file:///C:/Users/AA/Desktop/figmah/figmah/localhost_2025-01-20_22-01-36.report.html
http://localhost:3000/
45
91
100
92
100
Best Practices
TRUST AND SAFETY
Ensure CSP is effective against XSS attacks
Use a strong HSTS policy
Ensure proper origin isolation with COOP
GENERAL
Missing source maps for large first-party JavaScript
Type here to search
18°C Haze
10:10 PM
1/20/2025

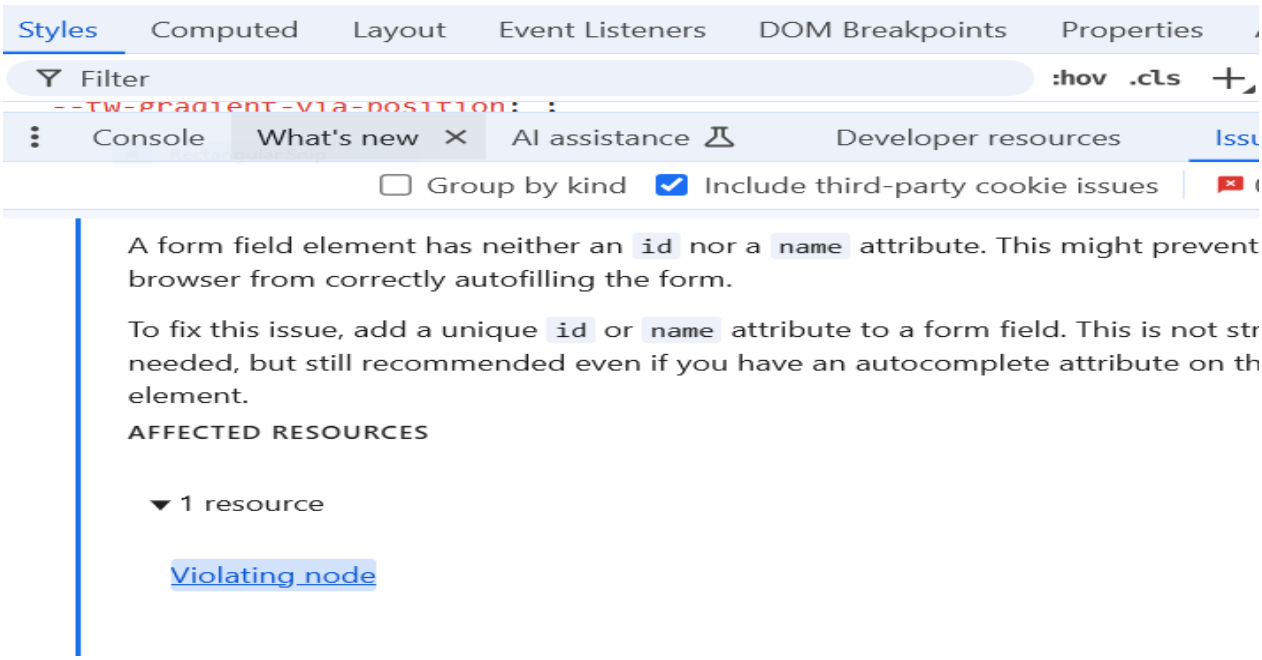
Developer Resources:



Performance monitor:



Issue:



Security: