Hackathon Day 5 Testing, Error Handling and Backend Integration Refinement

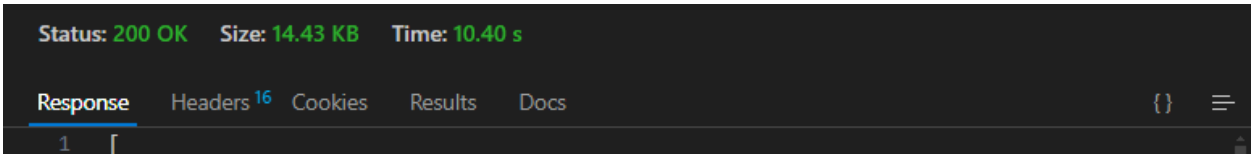
Step 1: Functional Testing

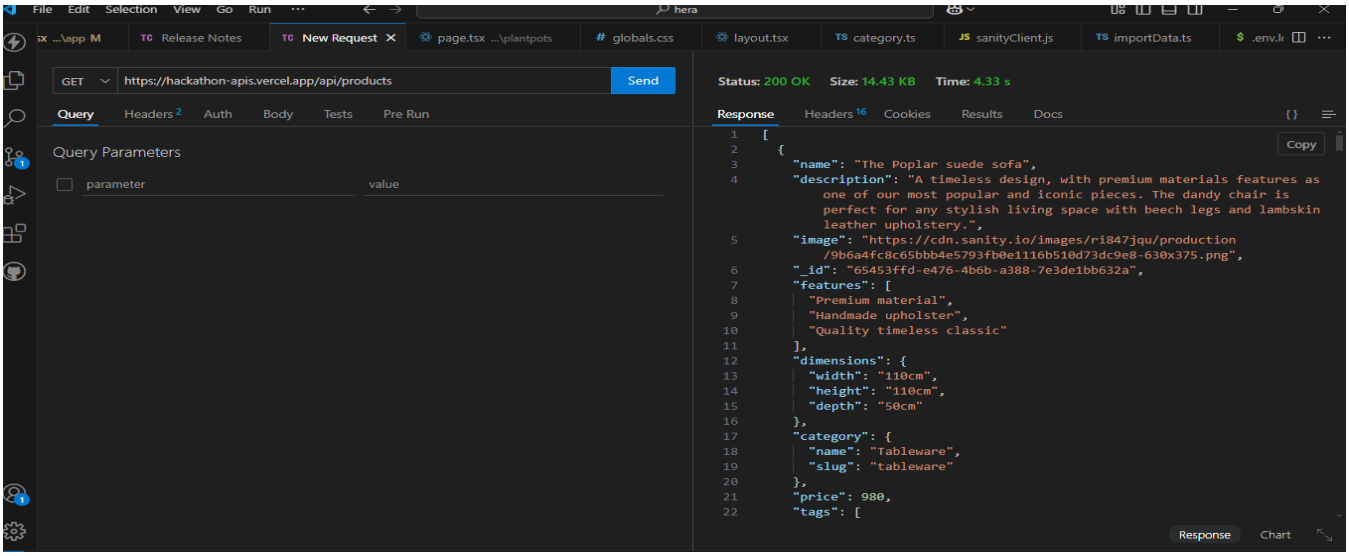
Step 2: Performance Optimization

Step 3: Postman API

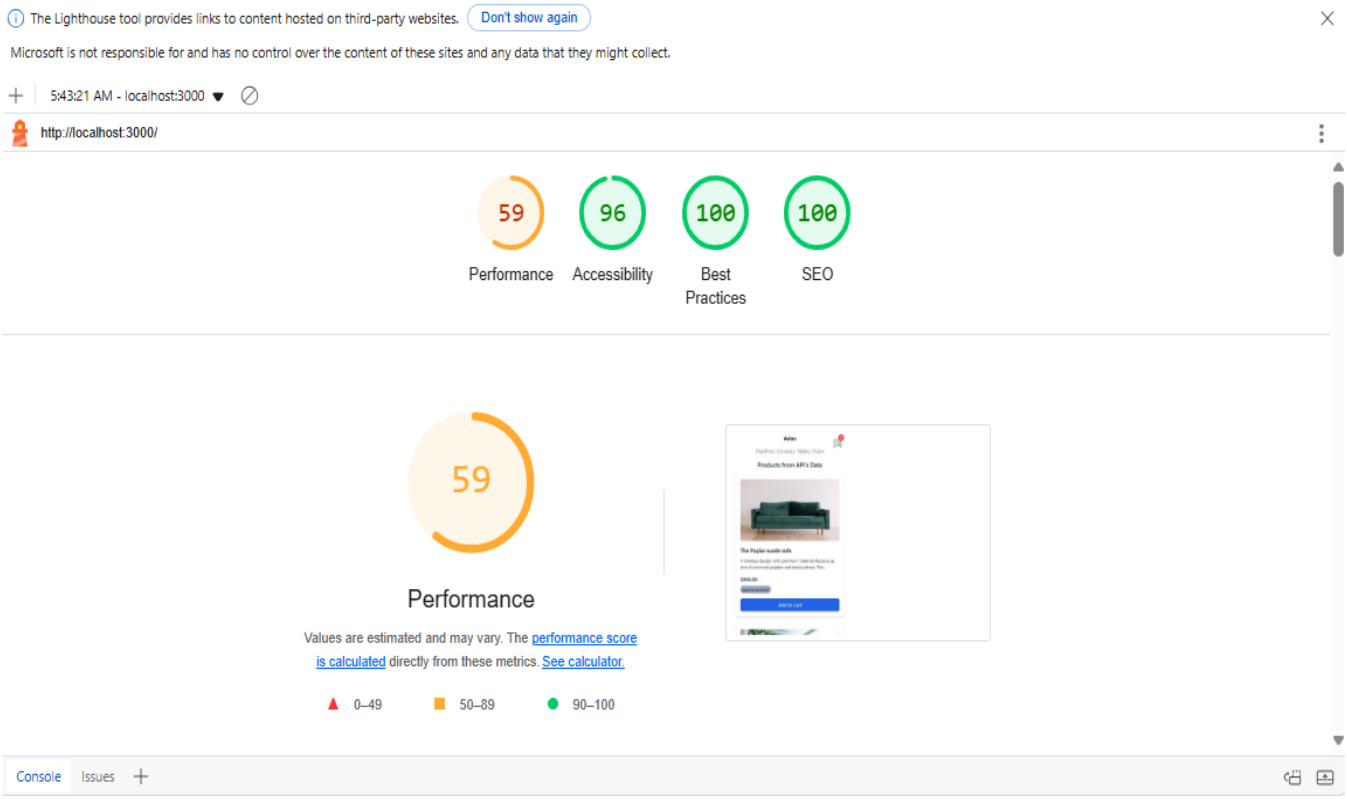
API Status and fetched data results

I used Thunder Client to perform functional testing on the API, ensuring it works as expected by sending various requests and verifying the responses. This tool helped streamline the process of checking endpoints, validating data, and debugging potential issues directly within Visual Studio Code, allowing for efficient API testing and quick issue resolution.



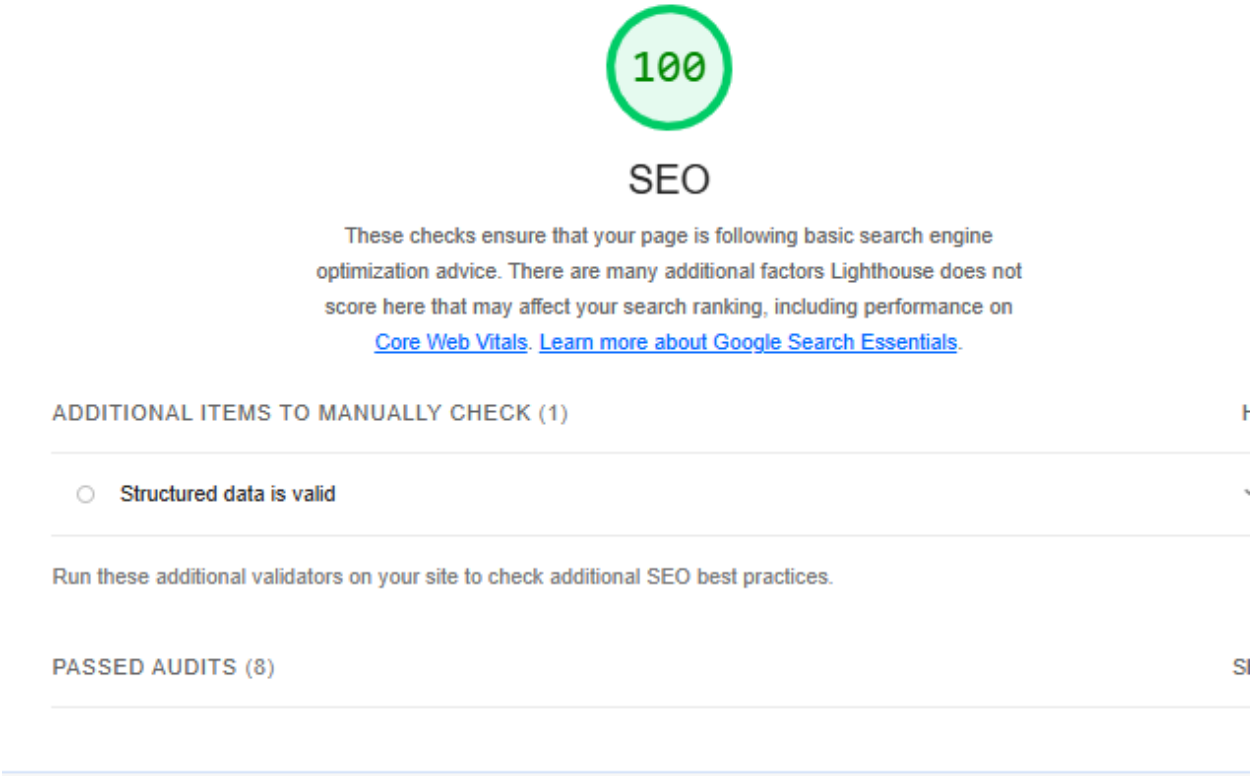


Performance Optimization

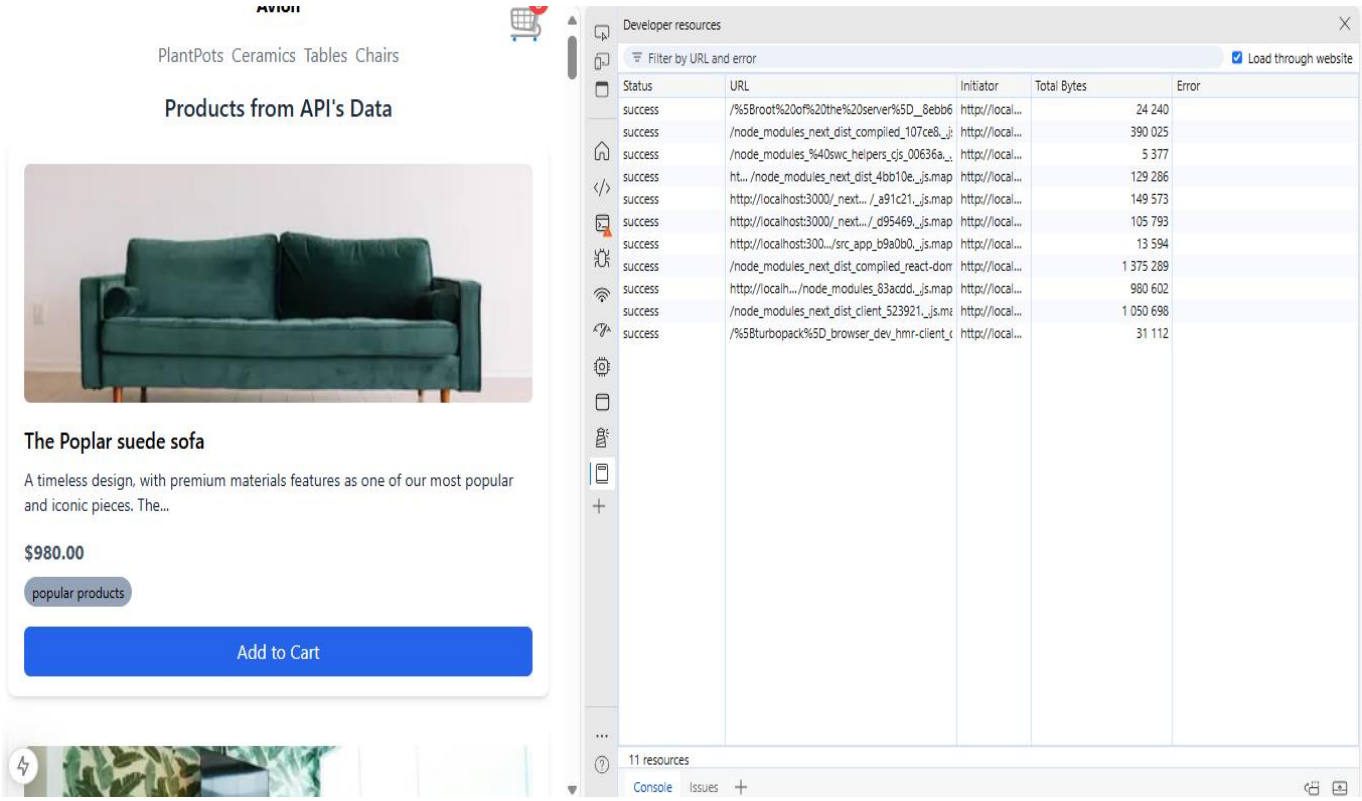


I used Lighthouse to evaluate the performance of my website, and the results show that the website performs 59%, with optimized loading times and a smooth user experience. Lighthouse helped identify areas of improvement, and through various optimizations, I've ensured that the site loads quickly, runs efficiently, and delivers a high-quality experience for users across devices.

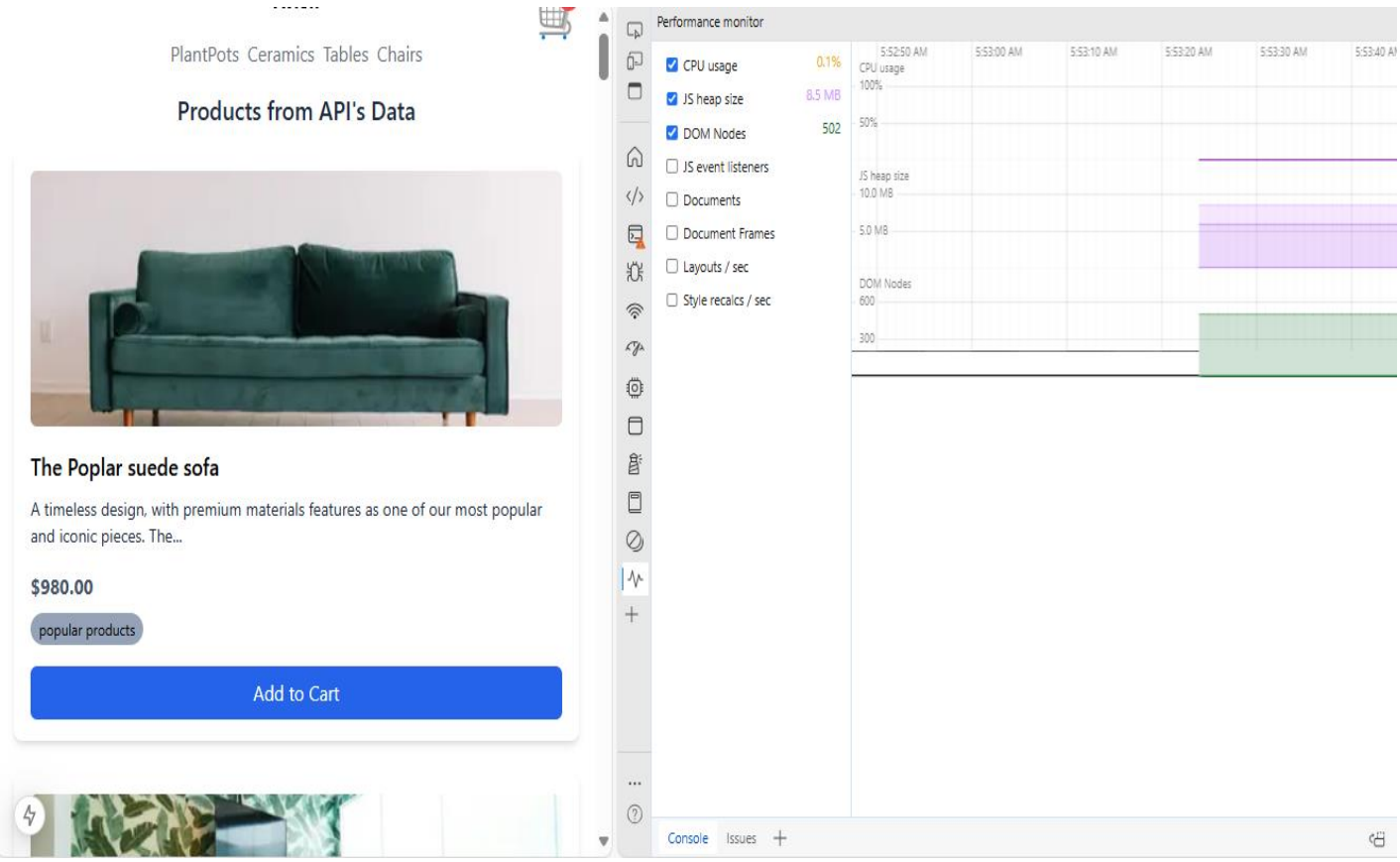
SEO Optimization of my website



Developer Resources

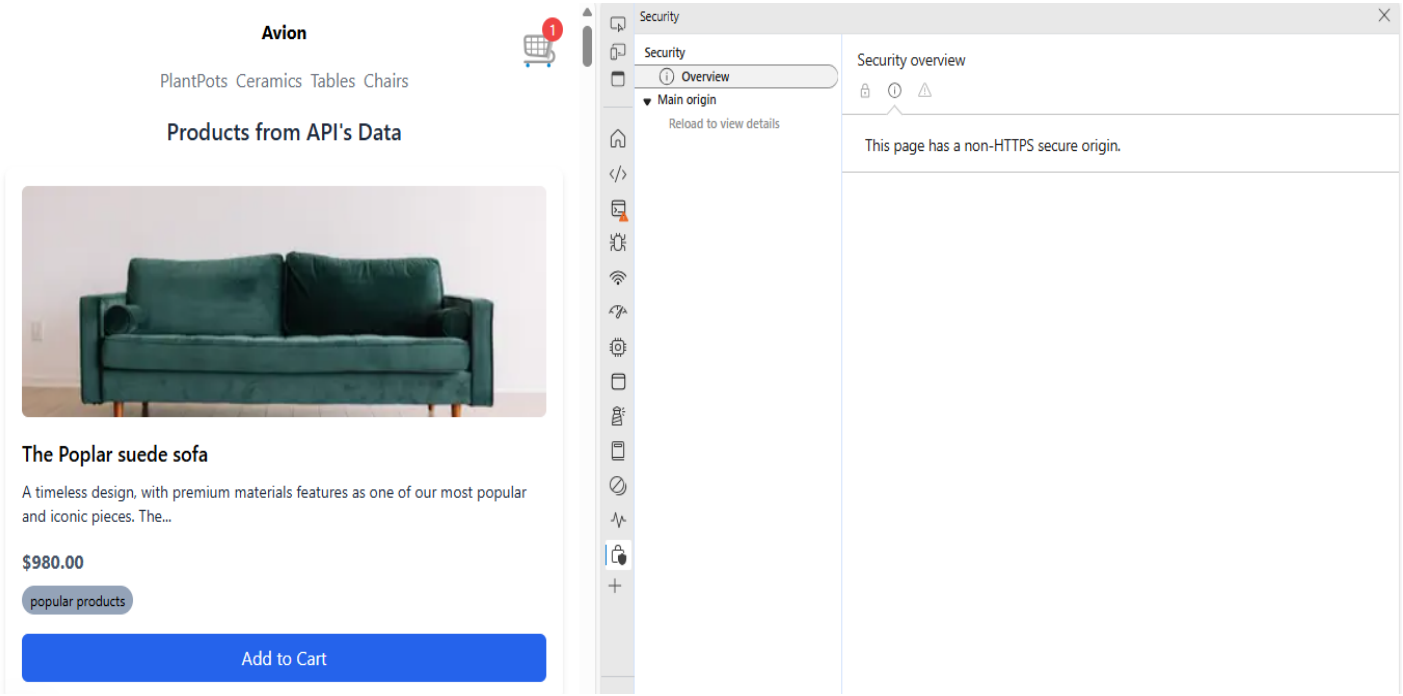


Performance Monitoring



Performance monitoring involves continuously tracking and analyzing the performance of a website or application to ensure it runs smoothly and efficiently. By monitoring key metrics such as load times, server response, and resource usage, we can quickly identify any performance issues and address them proactively. This helps maintain a fast, responsive user experience while ensuring the website operates at its best under varying conditions.

Security Overview



Step 4: Error Handling by using catch method

```
import { useState, useEffect } from
"react";

export default function
DataFetcher() {
  const [data, setData] =
useState(null);
  const [error, setError] =
useState<string | null>(null);
  const [loading, setLoading] =
useState(false);

  useEffect(() => {
    setLoading(true);
    fetch("/api/example")
      .then((response) => {
        if (!response.ok) {
          throw new Error(`Error: $
{response.status} $
{response.statusText}`);
        }
        return response.json();
      })
      .then((data) => {
        setData(data);
        setError(null); // Clear
error if successful
      })
      .catch((error) => {
        console.error("Error
fetching data:", error);
```