Test ID	Test Case Description	Expected Result	Actual Result	Status	Remarks
TST-001	Homepage Load Test	Homepage should load within 2 seconds	Homepage loaded within 2 seconds	Passed	Displayed Properly
TST-002	Car Display Test	All cars should be displayed correctly on the homepage	All cars displayed correctly	Passed	No Issues found
TST-003	Car Details Test	Car details should display upon selection	Car details displayed correctly	Passed	Showing Correctly
TST-004	Checkout Page Test	Checkout process should complete without errors	Checkout completed successfully	Passed	Successfully
TST-005	API Integration Test	API should fetch and display data correctly	API fetched and displayed data as expected	Passed	Working Properly
TST-006	Navigation Test	All navigation links should be functional	All navigation links work correctly	Passed	Working Properly
TST-007	Security Test	All API calls should be secure and use HTTPS	All API calls are secure and use HTTPS	Passed	Working Properly
TST-008	Image Loading Test	All car images should load correctly	All car images loaded without issues	Passed	Working Properly
TST-009	Accessibility Test	Site should meet accessibility standards	Site meets all accessibility standards	Passed	Working Properly
TST-010	Responsiveness Test	Site should adjust properly on various devices	Site adjusted properly on all tested devices	Passed	Displayed Properly

- Viewed the response, including the status code, response body, and headers.

5. **View the Response**
   - Used different viewing formats (Pretty, Raw, Preview) to examine the response.
6. **Save the Request**
   - Saved the request for future use in a collection.
7. **Use Environment Variables**
   - Created environment variables (e.g., `api_url`) to dynamically store values.
   - Used variables in request URLs (e.g., `{{api_url}}`).
8. **Test and Automate API Requests**
   - Added tests in the **Tests** tab to validate responses (e.g., check if status code is 200).
   - Used **Collection Runner** to execute multiple requests sequentially.

Secure API Communication and Storing Sensitive Data

To ensure secure API communication and manage sensitive data like API keys, I followed these steps:

1. Ensuring API Calls Over HTTPS

- I made sure that all API calls in my project are made over **HTTPS** (Hypertext Transfer Protocol Secure) to ensure secure transmission of data over the internet.
- I verified the base URL of the API to ensure it starts with `https://`.

  Example:

  ```javascript
  const apiUrl = 'https://api.example.com/data'; // Secure API endpoint
  ```

- For all API calls using **fetch** or **Axios**, the URLs are configured with **HTTPS** by default.

  Example:

  ```javascript
  // Example using fetch
  const response = await fetch('https://api.example.com/data');
  ```

2. Storing Sensitive Data in Environment Variables

For securely managing sensitive data such as API keys, secrets, or authentication tokens, I used **environment variables**. Here's how I implemented this in my **Next.js** project:

Step 1: Created `.env.local` File

- I created a `.env.local` file in the root directory of the Next.js project. This file is not committed to version control because it's added to `.gitignore`.
- Inside the `.env.local` file, I stored sensitive data, including API URLs and keys, like so:

  plaintext Copy Edit

  ```
  NEXT_PUBLIC_API_URL=https://api.example.com
  API_KEY=your_api_key_here
  ```

  - The `NEXT_PUBLIC_` prefix is used for variables that need to be accessible on the client-side, like the API URL.
  - For sensitive variables like **API_KEY**, I avoided the `NEXT_PUBLIC_` prefix to keep it only accessible server-side.

Step 2: Accessing Environment Variables in Code

- In my code, I accessed these environment variables securely using `process.env`.

Step 2: Accessing Environment Variables in Code

- In my code, I accessed these environment variables securely using `process.env`.

  Example:

  ```javascript
  const apiUrl = process.env.NEXT_PUBLIC_API_URL;
  const apiKey = process.env.API_KEY;

  const response = await fetch(`${apiUrl}/data?api_key=${apiKey}`);
  ```

  - The `NEXT_PUBLIC_API_URL` variable is safe to expose to the client-side since it's prefixed with `NEXT_PUBLIC_`.
  - The `API_KEY` variable is only used on the server-side to avoid exposing it to the browser.

Step 3: Adding `.env.local` to `.gitignore`

- To ensure the sensitive data doesn't get committed to version control, I added the `.env.local` file to my `.gitignore` file:

  ```plaintext
  .env.local
  ```

3. Using Environment Variables on Deployment

For deployment on **Vercel** (or similar platforms like **Netlify** or **AWS**), I securely configured environment variables:

- I navigated to the **Vercel** dashboard and accessed the **Settings** > **Environment Variables** section.
- Here, I added the necessary environment variables, including **API_KEY**, ensuring they are securely injected during production.

By following these steps, I ensured that:

- API keys and sensitive data are securely stored in environment variables and not exposed on the client-side.
- All API communication is encrypted via **HTTPS** for secure data transmission.

Postman API Testing Summary

I have followed these steps to test APIs using Postman:

1. **Download and Install Postman**
   - Downloaded and installed Postman from the official website.
2. **Create a New Request**
   - Opened Postman and clicked the "**New**" button to create a new request.
   - Chose a collection to save the request.
3. **Configure the Request**
   - Entered the API endpoint URL (e.g., `https://api.example.com/data`).
   - Selected the appropriate HTTP method (GET, POST, PUT, DELETE).
   - Added necessary headers, such as **Authorization** (`Bearer <your_token>`).
   - Added request body data in the **raw** format for POST or PUT requests.
4. **Send the Request**
   - Clicked **Send** to trigger the API call.
   - Viewed the response, including the status code, response body, and headers.
5. **View the Response**

Home Workspaces API Network Search Postman Invite Upgrade

My Workspace New Import

HEAD https://sanity-nextjs-a

No environment

Collections

Environments

History

My first collection

First folder inside collection

Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

https://sanity-nextjs-application.vercel.app/api/hackathon/template7

Save Share

HEAD https://sanity-nextjs-application.vercel.app/api/hackathon/template7 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

Body Cookies Headers (11) Test Results

200 OK 4.55 s 468 B

JSON Preview Visualize

1

200 OK

Request successful. The server has responded as required.

Online Console Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

PageSpeed Insights

Copy Link Docs

Mobile Desktop

Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

- Largest Contentful Paint image was lazily loaded
- Reduce initial server response time — Root document took 640 ms
- Reduce unused JavaScript — Potential savings of 25 KiB
- Eliminate render-blocking resources — Potential savings of 110 ms
- JavaScript execution time — 0.3 s
- Minimizes main-thread work — 0.6 s
- Avoid long main-thread tasks — 1 long task found
- Avoid large layout shifts — 1 layout shift found
- Avoids enormous network payloads — Total size was 244 KiB
- Avoids an excessive DOM size — 510 elements
- Avoid chaining critical requests — 1 chain found
- Largest Contentful Paint element — 2,060 ms

More information about the performance of your application. These numbers don't directly affect the Performance score.

PageSpeed Insights

Copy Link Docs

Mobile Desktop

Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

- Largest Contentful Paint image was lazily loaded
- Reduce unused JavaScript — Potential savings of 25 KiB
- JavaScript execution time — 0.3 s
- Initial server response time was short — Root document took 130 ms
- Avoid large layout shifts — 1 layout shift found
- Avoids enormous network payloads — Total size was 255 KiB
- Avoids an excessive DOM size — 510 elements
- Avoid chaining critical requests — 1 chain found
- Minimizes main-thread work — 0.5 s
- Largest Contentful Paint element — 490 ms
- Avoid long main-thread tasks — 1 long task found

More information about the performance of your application. These numbers don't directly affect the Performance score.

PASSED AUDITS (25) Show

web.postman.co/workspace/dd3053fa-35a3-4a8b-83e3-491f9d27722e/request/create?requestId=7dd3d88b-54de-4107-9563-d9f6510d1300

Home Workspaces API Network Search Postman Invite Upgrade

You seem to be offline. Data in this workspace might not be up to date.

My Workspace New Import

GET https://sanity-nextjs-ap

No environment

Collections

Environments

History

My first collection

First folder inside collection

Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

https://sanity-nextjs-application.vercel.app/api/hackathon/template7

Save Share

GET https://sanity-nextjs-application.vercel.app/api/hackathon/template7 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

Body Cookies Headers (12) Test Results 200 OK 2.58 s 5.38 KB

JSON Preview Visualize


```
[
    {
        "id": 1,
        "name": "Koenigsegg",
        "type": "Sport",
        "fuel_capacity": "90L",
```


Connecting... Console Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

Home Workspaces API Network Search Postman Invite Upgrade

My Workspace New Import

HEAD https://sanity-nextjs-a

No environment

Collections

Environments

History

My first collection

First folder inside collection

Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

https://sanity-nextjs-application.vercel.app/api/hackathon/template7

Save Share

HEAD https://sanity-nextjs-application.vercel.app/api/hackathon/template7 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

Body Cookies Headers (11) Test Results

200 OK 4.55 s 468 B

JSON Preview Visualize

1

200 OK

Request successful. The server has responded as required.

Online Console Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

PageSpeed Insights

Copy Link Docs

Mobile Desktop

Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

- Largest Contentful Paint image was lazily loaded
- Reduce initial server response time — Root document took 640 ms
- Reduce unused JavaScript — Potential savings of 25 KiB
- Eliminate render-blocking resources — Potential savings of 110 ms
- JavaScript execution time — 0.3 s
- Minimizes main-thread work — 0.6 s
- Avoid long main-thread tasks — 1 long task found
- Avoid large layout shifts — 1 layout shift found
- Avoids enormous network payloads — Total size was 244 KiB
- Avoids an excessive DOM size — 510 elements
- Avoid chaining critical requests — 1 chain found
- Largest Contentful Paint element — 2,060 ms

More information about the performance of your application. These numbers don't directly affect the Performance score.

PageSpeed Insights

Copy Link Docs

Mobile Desktop

Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

- Largest Contentful Paint image was lazily loaded
- Reduce unused JavaScript — Potential savings of 25 KiB
- JavaScript execution time — 0.3 s
- Initial server response time was short — Root document took 130 ms
- Avoid large layout shifts — 1 layout shift found
- Avoids enormous network payloads — Total size was 255 KiB
- Avoids an excessive DOM size — 510 elements
- Avoid chaining critical requests — 1 chain found
- Minimizes main-thread work — 0.5 s
- Largest Contentful Paint element — 490 ms
- Avoid long main-thread tasks — 1 long task found

More information about the performance of your application. These numbers don't directly affect the Performance score.

PASSED AUDITS (25) Show

Test ID	Test Case Description	Expected Result	Actual Result	Status	Remarks
TST-001	Homepage Load Test	Homepage should load within 2 seconds	Homepage loaded within 2 seconds	Passed	Displayed Properly
TST-002	Car Display Test	All cars should be displayed correctly on the homepage	All cars displayed correctly	Passed	No Issues found
TST-003	Car Details Test	Car details should display upon selection	Car details displayed correctly	Passed	Showing Correctly
TST-004	Checkout Page Test	Checkout process should complete without errors	Checkout completed successfully	Passed	Successfully
TST-005	API Integration Test	API should fetch and display data correctly	API fetched and displayed data as expected	Passed	Working Properly
TST-006	Navigation Test	All navigation links should be functional	All navigation links work correctly	Passed	Working Properly
TST-007	Security Test	All API calls should be secure and use HTTPS	All API calls are secure and use HTTPS	Passed	Working Properly
TST-008	Image Loading Test	All car images should load correctly	All car images loaded without issues	Passed	Working Properly
TST-009	Accessibility Test	Site should meet accessibility standards	Site meets all accessibility standards	Passed	Working Properly
TST-010	Responsiveness Test	Site should adjust properly on various devices	Site adjusted properly on all tested devices	Passed	Displayed Properly

Test ID	Test Case Description	Expected Result	Actual Result	Status	Remarks
TST-001	Homepage Load Test	Homepage should load within 2 seconds	Homepage loaded within 2 seconds	Passed	Displayed Properly
TST-002	Car Display Test	All cars should be displayed correctly on the homepage	All cars displayed correctly	Passed	No Issues found
TST-003	Car Details Test	Car details should display upon selection	Car details displayed correctly	Passed	Showing Correctly
TST-004	Checkout Page Test	Checkout process should complete without errors	Checkout completed successfully	Passed	Successfully
TST-005	API Integration Test	API should fetch and display data correctly	API fetched and displayed data as expected	Passed	Working Properly
TST-006	Navigation Test	All navigation links should be functional	All navigation links work correctly	Passed	Working Properly
TST-007	Security Test	All API calls should be secure and use HTTPS	All API calls are secure and use HTTPS	Passed	Working Properly
TST-008	Image Loading Test	All car images should load correctly	All car images loaded without issues	Passed	Working Properly
TST-009	Accessibility Test	Site should meet accessibility standards	Site meets all accessibility standards	Passed	Working Properly
TST-010	Responsiveness Test	Site should adjust properly on various devices	Site adjusted properly on all tested devices	Passed	Displayed Properly