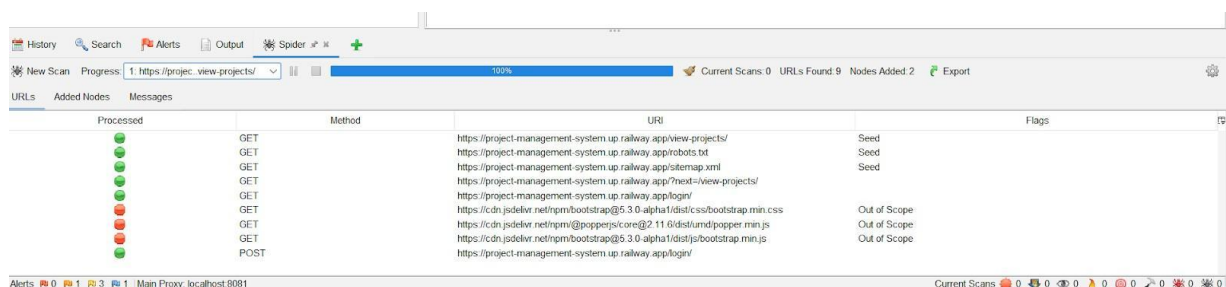


Non Functional Testing

Non-functional testing in Django web apps involves testing the application's behavior and performance under different conditions beyond its expected functional requirements. These tests focus on aspects such as scalability, usability, security, and performance.

Security Testing



Processed	Method	URI	Flags
●	GET	https://project-management-system.up.railway.app/view-projects/	Seed
●	GET	https://project-management-system.up.railway.app/robots.txt	Seed
●	GET	https://project-management-system.up.railway.app/sitemap.xml	Seed
●	GET	https://project-management-system.up.railway.app/?next=/view-projects/	
●	GET	https://project-management-system.up.railway.app/login/	
●	GET	https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha.1/dist/css/bootstrap.min.css	Out of Scope
●	GET	https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.0/dist/umd/popper.min.js	Out of Scope
●	GET	https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha.1/dist/js/bootstrap.min.js	Out of Scope
●	POST	https://project-management-system.up.railway.app/login/	

This output indicates that the spidering process performed by OWASP ZAP on the Django web application successfully retrieved several pages, including the login page at <https://project-management-system.up.railway.app/login/>. The robots.txt and sitemap.xml URLs were also retrieved, as well as the /view-projects/ URL, which was the page that the application redirected to after a successful login (as indicated by the /?next=/view-projects/ URL).

The Out of Scope URLs are resources hosted on a Content Delivery Network (CDN) and are not part of the Django web application itself. These resources are typically used by the application to style the pages (CSS files) or provide interactivity (JavaScript files).

The POST request to the /login/ URL indicates that the spidering process was also able to submit a login form. This is an important step in security testing, as it allows for testing the authentication and authorization mechanisms of the application. The response to the POST request should be further analyzed to ensure that the login process is secure and is not vulnerable to attacks such as SQL injection, cross-site scripting (XSS), or cross-site request forgery (CSRF).

XSS is a vulnerability that occurs when an attacker is able to inject malicious code into a web page, which then executes in the browser of other users who

visit the page. This can allow the attacker to steal sensitive information or perform actions on behalf of the user without their consent. XSS attacks typically involve the manipulation of client-side scripts and are often caused by inadequate input validation and output encoding.

CSRF is a vulnerability that occurs when an attacker is able to trick a user into performing an unintended action on a web application that they are currently authenticated to. This is achieved by injecting a malicious request (e.g., a hidden form or a crafted URL) that performs the action without the user's knowledge. CSRF attacks typically involve the manipulation of server-side requests and are often caused by inadequate authentication and session management.

Host:	https://project-management-system.up.railway.app					
Plugin	Strength	Progress	Elapsed	Reqs	Alerts	Status
Path Traversal	Medium		00:18.899	75	0	✓
Remote File Inclusion	Medium		00:12.440	50	0	✓
Heartbleed OpenSSL Vulnerability	Medium		00:01.910	4	0	✓
Source Code Disclosure - /WEB-INF folder	Medium		00:02.602	9	0	✓
Source Code Disclosure - CVE-2012-1823	Medium		00:02.945	2	0	✓
Remote Code Execution - CVE-2012-1823	Medium		00:02.511	6	0	✓
External Redirect	Medium		00:12.781	27	0	✓
Server Side Include	Medium		00:04.864	20	0	✓
Cross Site Scripting (Reflected)	Medium		00:07.525	26	0	✓
Cross Site Scripting (Persistent) - Prime	Medium		00:01.220	5	0	✓
Cross Site Scripting (Persistent) - Spider	Medium		00:00.884	4	0	✓
Cross Site Scripting (Persistent)	Medium		00:00.239	0	0	✓
SQL Injection	Medium		00:37.371	144	0	✓
SQL Injection - MySQL	Medium		00:10.252	42	0	✓
SQL Injection - Hypersonic SQL	Medium		00:07.609	31	0	✓
SQL Injection - Oracle	Medium		00:08.561	35	0	✓
SQL Injection - PostgreSQL	Medium		00:07.781	32	0	✓
SQL Injection - SQLite	Medium		00:11.190	43	0	✓
Cross Site Scripting (DOM Based)	Medium		00:01.748	0	0	✗
SQL Injection - MsSQL	Medium		00:05.196	16	0	✓
Server Side Code Injection	Medium		00:18.098	40	0	✓
Remote OS Command Injection	Medium		00:59.088	207	0	✓
XML External Entity Attack	Medium		00:00.253	0	0	✓
Generic Padding Oracle	Medium		00:00.644	2	0	✓
Cloud Metadata Potentially Exposed	Medium		00:03.073	8	0	✓
Directory Browsing	Medium		00:03.073	4	0	✓
Buffer Overflow	Medium		00:01.987	5	0	✓
Format String Error	Medium		00:03.808	15	0	✓
CRLF Injection	Medium		00:08.908	35	0	✓
Parameter Tampering	Medium		00:01.475	6	0	✓
ELMAH Information Leak	Medium		00:00.369	1	0	✓
Trace.axd Information Leak	Medium		00:00.853	2	0	✓
.htaccess Information Leak	Medium		00:00.523	2	0	✓
.env Information Leak	Medium		00:00.508	2	0	✓

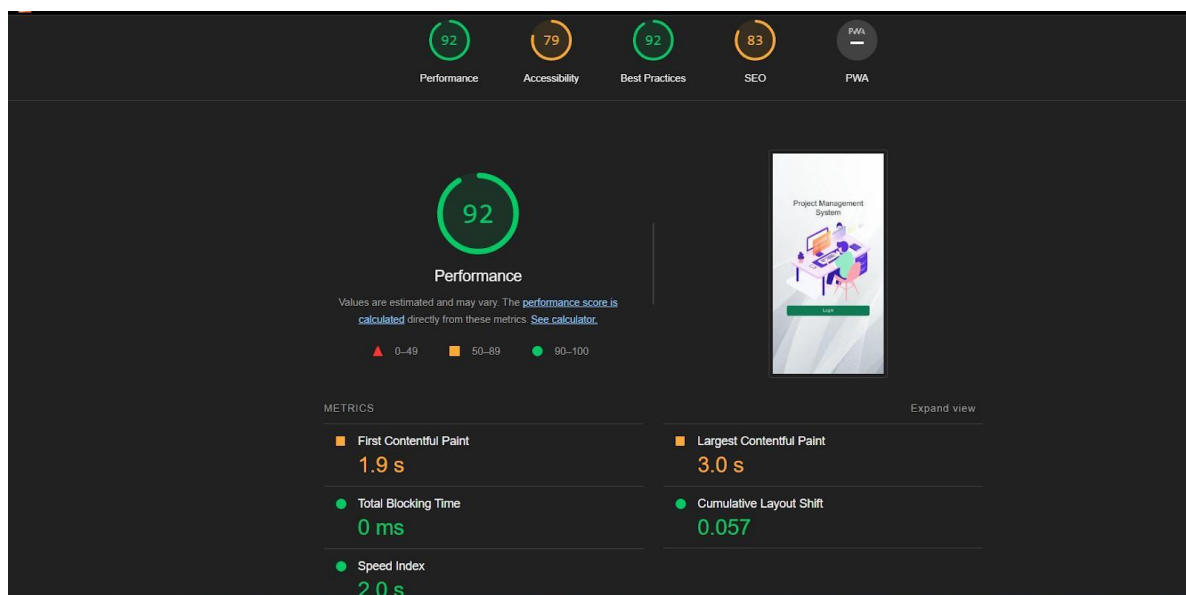
As we can see here, we haven't got any alerts for Cross-Site Scripting, or Cross-Site Request Forgery, or any other vulnerability.

We haven't got any Cross-Site Request Forgery vulnerability because we have used CSRF middleware and the template tag which provides easy-to-use protection against Cross-Site Request Forgeries.

Performance / Accessibility Testing

Testing a web application in order to make sure that each and every user can easily access the website is known as Accessibility Testing. The specialized and dedicated branch of testing that helps ensure that websites are indeed effective in this area is called “Web Accessibility Testing”.

Landing page testing



Login page testing

