Analyze a Full-Stack Next.js and NestJS Source

September 10, 2026

Analyze a Full-Stack Next.js and NestJS Source

This page explains how to prepare and review a full-stack Next.js and NestJS source project in Atloria. Completing this review helps you confirm that Atloria has the context it needs before you create user-facing documentation or investigate product behavior.

Overview

Atloria analyzes the source provided for a project and organizes its findings into a project-focused workspace. For a full-stack Next.js and NestJS application, this helps you understand how the user interface, server-side behavior, and shared product concepts relate to one another.

Use source analysis when you have added or selected a project and need to verify the available source context before documenting the product. This is especially useful when the product includes both frontend screens and backend services, because user-visible behavior can depend on the relationship between the two.

During analysis, Atloria can identify application structure and relationships, such as components that render other components, frontend hooks and contexts, state stores, API interactions, services, and data models. Treat this information as context for understanding the product—not as a substitute for checking the user experience in the running application.

When a parse completes successfully, the project's source context is available for review. A successful parse may also complete with warnings. In that case, you can continue reviewing the available findings, while noting that some source details may require validation.

> 📷 *Screenshot pending: The Projects area showing a selected project and the Source action.*

Prerequisites

Before you begin, make sure that you have:

- Access to Atloria and permission to open the relevant project.
- A project containing the full-stack Next.js and NestJS source.
- Source that has been added or connected to the project.
- A clear product area or documentation goal to investigate.
- Enough time for the source parse to finish before relying on analysis results.
- A supported browser session in which you can access **Projects** and the project workspace.

Step-by-Step Instructions

1. Select **Projects** to open the list of projects available to you.
2. Use **Search projects...** to locate the full-stack Next.js and NestJS project if it is not immediately visible.
3. Open the project you want to analyze from the project list.
4. Select **Open Project Dashboard** to enter the focused workspace for the selected project.
5. Select **Source** to open the project’s source-analysis area.
6. Select **View Source** to review the source context that Atloria has prepared for the project.
7. Confirm that the source analysis has completed successfully before using its findings for documentation work.
8. Review any warnings shown with the completed parse before assuming that every part of the project was analyzed.
9. Identify the frontend area relevant to your task, such as a page, screen component, or user-facing feature.
10. Identify the corresponding backend area when the feature depends on server-side data, actions, or business rules.
11. Follow the displayed relationship context to understand how the user-facing area connects to supporting application behavior.

12. Record the user-visible screens, actions, and outcomes that you need to validate in the running product.
13. Select **User Docs** when you are ready to use the prepared source context while working on user documentation.
14. Select ← **Back to Projects** when you need to leave the focused project and choose another project.

> 📷 *Screenshot pending: The Source area after analysis, including parse completion status and source relationship context.*

Tips and Best Practices

- Start from the user experience. Begin with the screen or workflow that users interact with, then use source analysis to understand supporting behavior. This keeps your documentation focused on what users see and do.
- Review both sides of a full-stack feature. In a Next.js and NestJS project, a frontend screen may depend on data or actions provided by the backend. Confirm the connection before documenting outcomes such as saved changes, loaded profile data, or status updates.
- Treat warnings as a review signal. A parse can complete successfully with warnings. You can use the available results, but validate important user-facing details in the product before publishing documentation.
- Use relationship context to trace behavior carefully. Atloria can surface chains involving rendered components, hooks, contexts, stores, actions, API calls, services, and models. Use these relationships to understand why a screen behaves as it does, but describe the final documentation in user language.
- Keep terminology consistent with the product. When you write the resulting guide, use labels that users see, such as **Projects**, **Source**, **View Source**, and **Open Project Dashboard**.
- Analyze only the scope you need. If you are documenting one feature, focus first on the relevant frontend screen and its supporting behavior. This makes validation faster and reduces unrelated findings.

Related Pages

- Manage Projects
- Open a Project Dashboard
- Review Source Context
- Create User Documentation from Product Evidence
- Validate Documentation Against the Running Product

Troubleshooting

The project does not appear in the list

1. Select **Projects**.
2. Enter the project name in **Search projects...**.
3. Confirm that you have access to the project if it still does not appear.

Source analysis has not completed

1. Return to the project dashboard.
2. Select **Source**.
3. Wait until the parse completion status indicates that the analysis has finished before relying on the source findings.

The parse completed with warnings

1. Review the warning information shown with the completed parse.
2. Continue using the available source context for initial investigation.
3. Validate critical screens and user outcomes in the running application before finalizing documentation.