LLM Message Dispatcher

| ⚙ Filter ⌄ | 1W | **1M** | 1Y |

Lines of code

595 (/1/analyses/1/scope/analysis-data/file-content)

Programming languages

`TS` `JS`

Code Health ⓘ

Healthy

--- Month reference

	10 –
	5 –
	0 –
17/02 24/02 3/03 10/03 17/03	

Knowledge Distribution ⓘ

Healthy

Team-Code Alignment ⓘ

Not available

Additional configuration steps are required before this factor can be used.

Define teams (/developer-configurations/for-project/1/developers)

Deliver...

Addition... before t...

Configu... integrat...

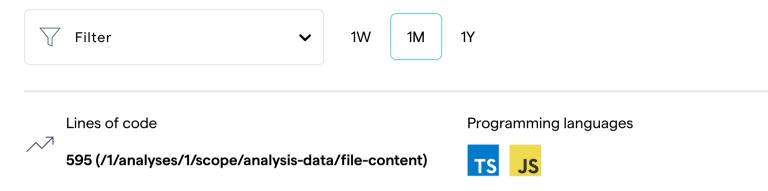ⓘ **What is Code Health?** ⌄

Hotspot Code Health ⓘ

Healthy

Hotspots are the files with most development activity. Even a minor amount of technical debt in a hotspot will become expensive due to the high development activity.

View hotspots (/1/analyses/1/code/biomarkers?)

Average Code Health ⓘ

Healthy

The weighted average health of all the files in the codebase. This metric indicates how deep any potential code health issues go.

Explore codebase (/1/analyses/1/code/hotspots/system-map?max-code-health=10&min-change-freq=default#codehealth)

Worst Performer ⓘ

Healthy

The lowest code health score measured in the codebase. Points out long-term risks that you need to be aware of if the low performer is worked on.

View worst performers (/1/analyses/1/code/hotspots/system-map?max-code-health=9.5363865&min-change-freq=0#codehealth)

↗ Latest improvements

No improvements available

⚠ Active risks & improvement opportunities

⚡ Code health focus areas

Interactive hotspot map

Visualize hotspots, code health, defects, etc. and filter by team and area.

Red code	**0%**
Yellow code	**0%**
Green code	**100%**

View hotspots (/1/analyses/1/code/hotspots/system-map?max-code-health=10&min-change-freq=default#codehealth)

File-level hotspots

Prioritizing improvements to these files will result in the greatest immediate benefit in terms of technical debt.

Hotspots	**100%**

Development effort in hotspots   **100%**

View hotspots (/1/analyses/1/code/biomarkers?)

↑ 8.19

Architectural code health

Monitor the code health of architectural components.

You need to **configure (/projects/1/configuration#architecture)** architectural components and rerun the analysis to see this data.

Configure components (/projects/1/configuration#architecture)

Quality gates

Track the impact of CodeScene's Pull Request integration. Numbers are from current month.

PR checks performed   0

Negative findings   0

Findings fixed	0
Findings ignored	0
Findings suppressed	0

View PR statistics (/1/analyses/1/project-management/delta/statistics)

Planned vs. unplanned work

This is time spent on work that was not part of the original scope.

Couldn't discover completed tasks. You need to configure a Work Done Transition name in your PM configuration.

View unplanned work (/1/analyses/1/project-management/hotspots/system-level)

Change coupling

Identify implicit dependencies in your architecture.

File-level

By commits   **0 entities**

Across commits   **0 entities**

Between repositories   **0 entities**

View coupled entities (/1/analyses/1/code/temporal-coupling/by-commits)