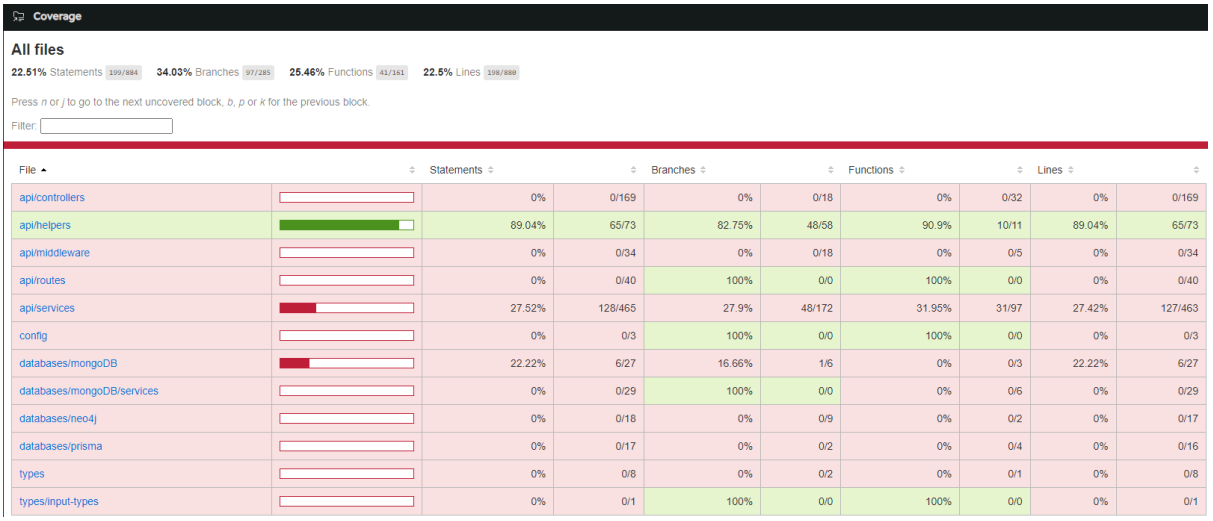
We used Vitest for its fast testing capabilities. It allowed us to observe our code coverage which enabled us to find where to focus our efforts.

Overall coverage report


Coverage

All files

22.51% Statements 199/884 34.03% Branches 97/285 25.46% Functions 41/161 22.5% Lines 198/880

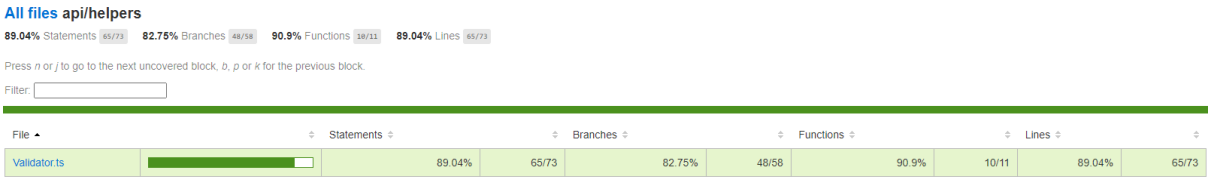
Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
api/controllers	0%	0/169	0%	0/18	0%	0/32	0%	0/169
api/helpers	89.04%	65/73	82.75%	48/58	90.9%	10/11	89.04%	65/73
api/middleware	0%	0/34	0%	0/18	0%	0/5	0%	0/34
api/routes	0%	0/40	100%	0/0	100%	0/0	0%	0/40
api/services	27.52%	128/465	27.9%	48/172	31.95%	31/97	27.42%	127/463
config	0%	0/3	100%	0/0	100%	0/0	0%	0/3
databases/mongoDB	22.22%	6/27	16.66%	1/6	0%	0/3	22.22%	6/27
databases/mongoDB/services	0%	0/29	100%	0/0	0%	0/6	0%	0/29
databases/neo4j	0%	0/18	0%	0/9	0%	0/2	0%	0/17
databases/prisma	0%	0/17	0%	0/2	0%	0/4	0%	0/16
types	0%	0/8	0%	0/2	0%	0/1	0%	0/8
types/input-types	0%	0/1	100%	0/0	100%	0/0	0%	0/1

This shows how much coverage of our statements, branches and functions we've covered. In this example, we can see that most of our helpers function is tested and passing - This is the code where in our validation logic and helper functions lies.

Specific coverage of api/helpers


All files api/helpers

89.04% Statements 65/73 82.75% Branches 48/58 90.9% Functions 10/11 89.04% Lines 65/73

Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

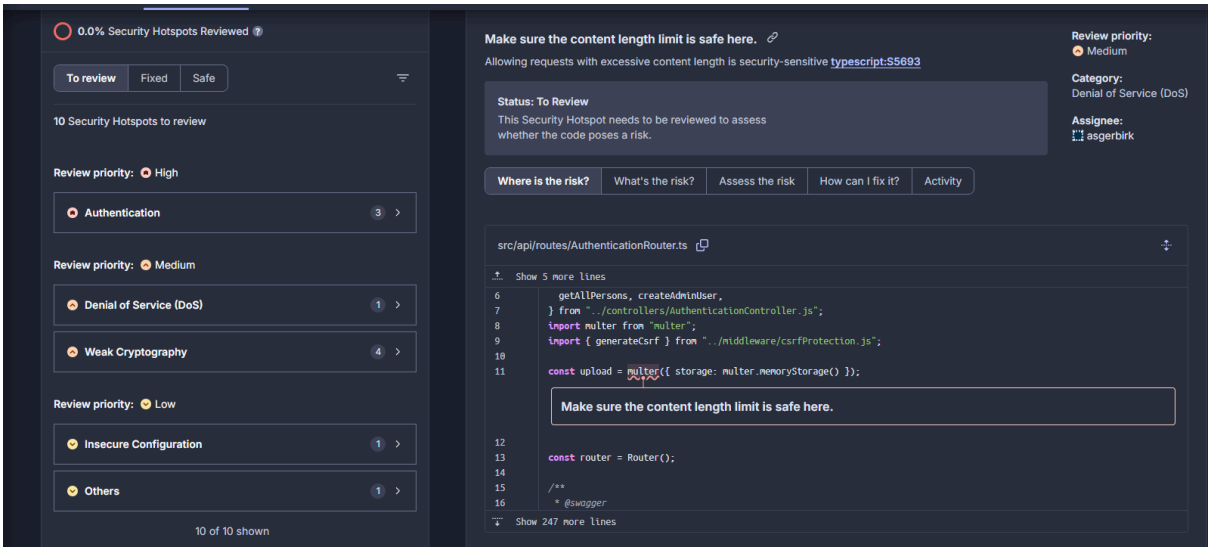
File	Statements		Branches		Functions		Lines	
Validator.ts	89.04%	65/73	82.75%	48/58	90.9%	10/11	89.04%	65/73

For this we have 89% coverage of statements and lines, which is 65 out of 73.
For the branches (paths) we have 82.75% coverage which is 48 out of 58.
And for functions we have 90% coverage, which is 10 out of 11.

By having a coverage report, it gives us a clear overview of the code that hasn't been tested. Therefore we focus on writing the uni tests for areas with low coverage. Seeing the branches that were covered helped us a lot with adjusting our tests.

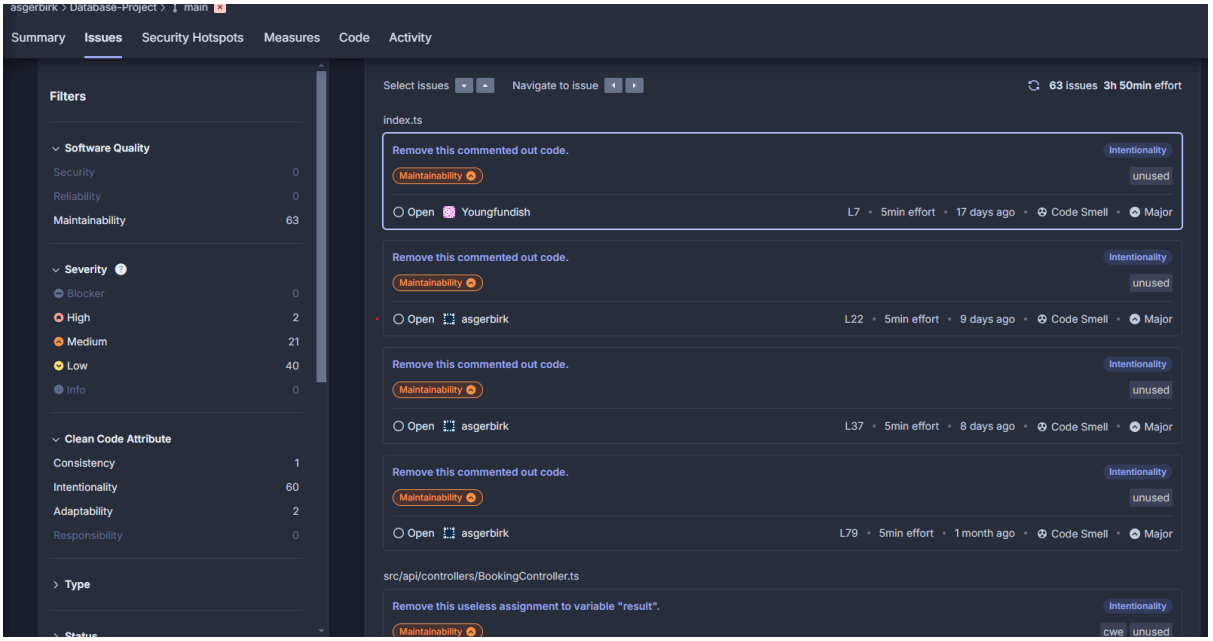
For static testing tools, we used SonarQube to analyze our code for bugs, vulnerabilities and code smells.
It'll help us catch issues early and make us optimize our code.

Security hotspots

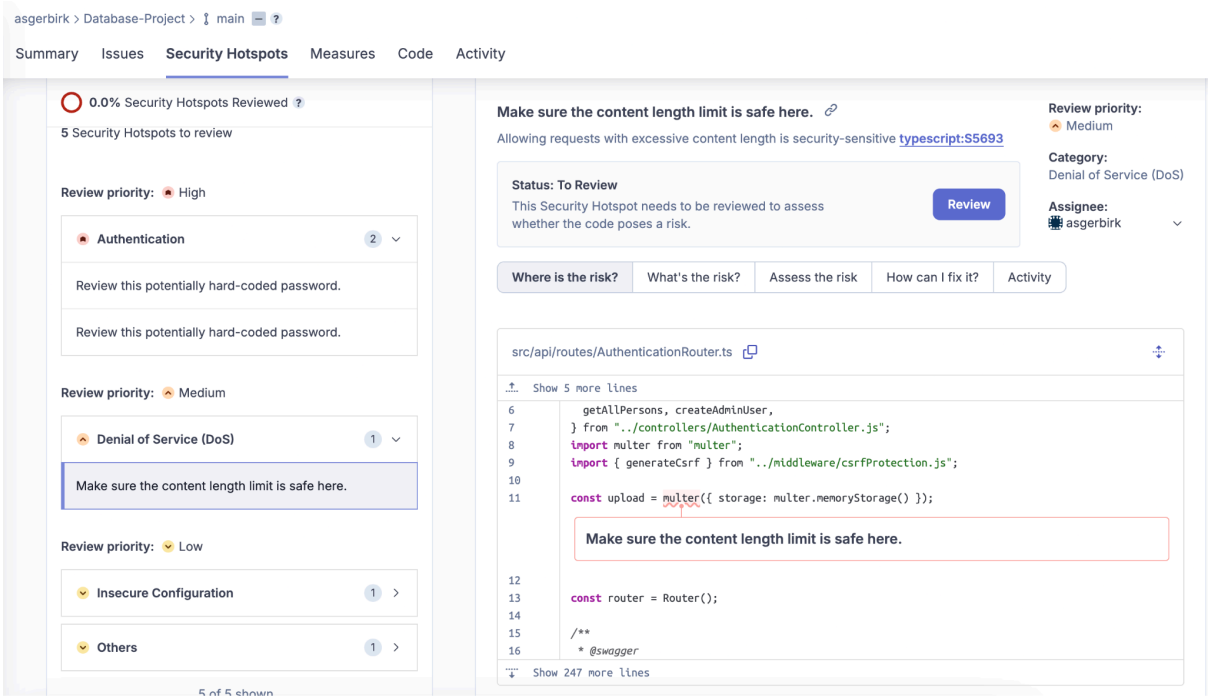


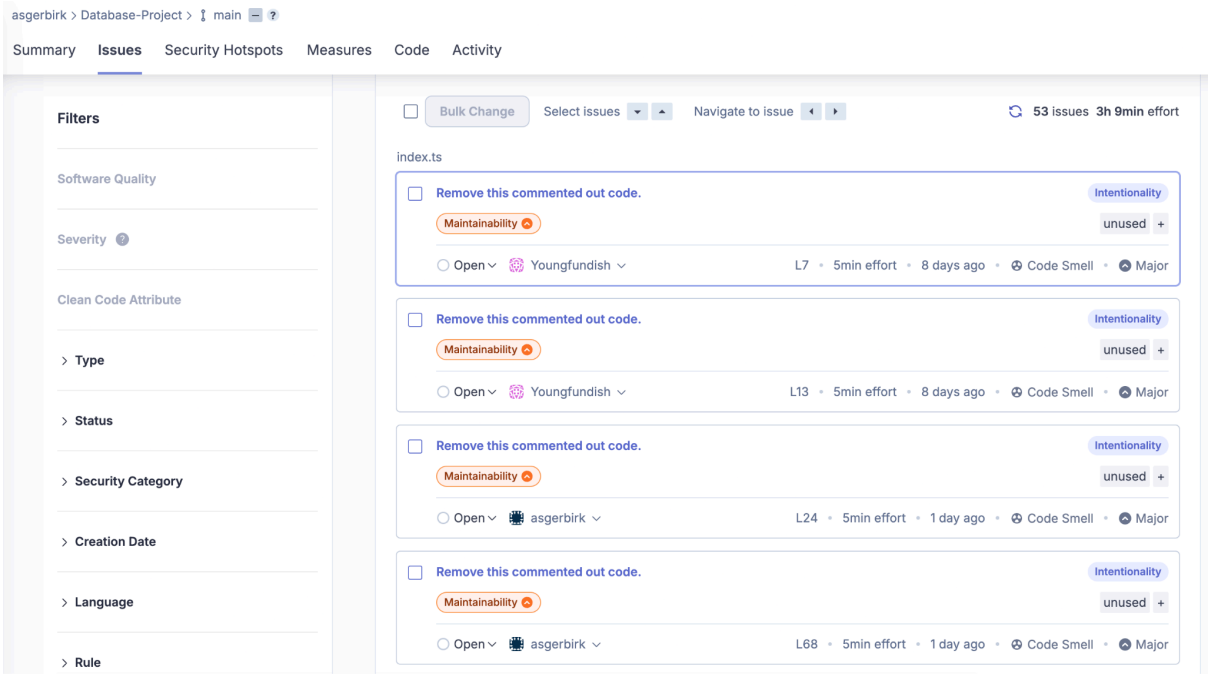
Security hotspots of different priorities → high, medium and low.
Helps identify the risks and lets us fix potential vulnerabilities.

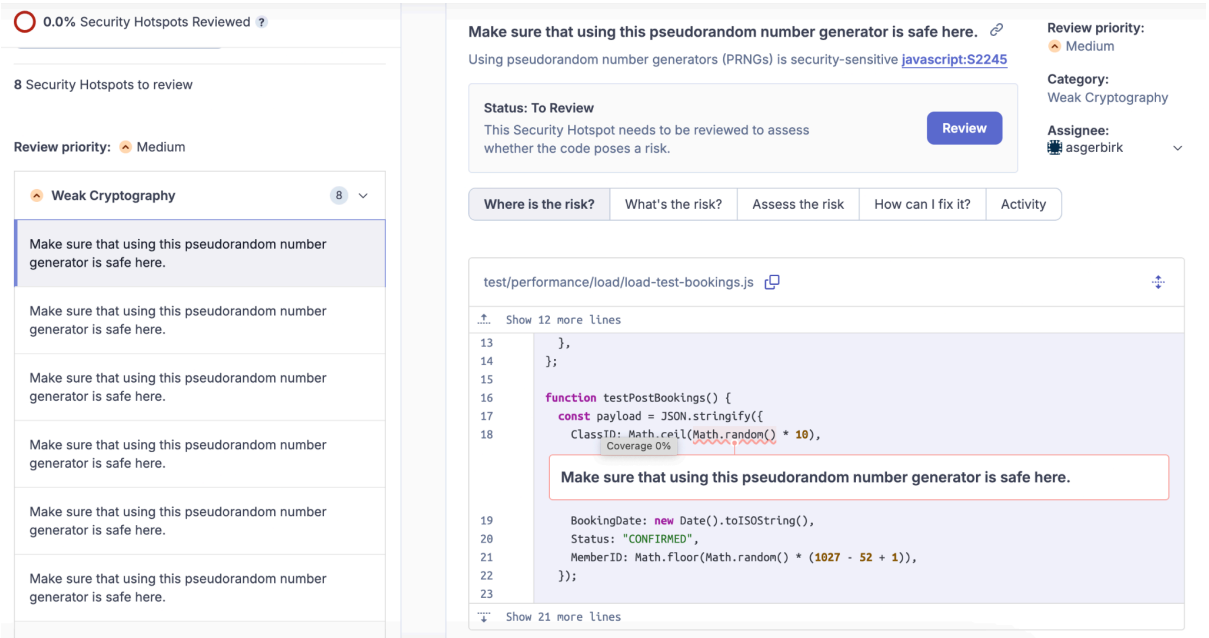
Overall issues



Defined by severities → high, medium and low.
Also shows which code attributes are connected to the issues.
These are consistency, intentionality and adaptability.
There are 60 intentionality issues, which is mainly about unused imports and commented code. This helps us clean up our codebase.


asgerbirk > Database-Project > main
Summary
Issues
Security Hotspots
Measures
Code
Activity
0.0% Security Hotspots Reviewed
5 Security Hotspots to review
Review priority: High
Authentication 2
Review this potentially hard-coded password.
Review this potentially hard-coded password.
Review priority: Medium
Denial of Service (DoS) 1
Make sure the content length limit is safe here.
Review priority: Low
Insecure Configuration 1
Others 1
5 of 5 shown
Make sure the content length limit is safe here.
Allowing requests with excessive content length is security-sensitive typescript:S5693
Status: To Review
This Security Hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Review priority:
Medium
Category:
Denial of Service (DoS)
Assignee:
asgerbirk
Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity
src/api/routes/AuthenticationRouter.ts
Show 5 more lines
getAllPersons, createAdminUser,
} from "../controllers/AuthenticationController.js";
import multer from "multer";
import { generateCsrf } from "../middleware/csrfProtection.js";
const upload = multer({ storage: multer.memoryStorage() });
Make sure the content length limit is safe here.
const router = Router();
/**
* @swagger
Show 247 more lines


asgerbirk > Database-Project > main
Summary
Issues
Security Hotspots
Measures
Code
Activity
Filters
Software Quality
Severity
Clean Code Attribute
Type
Status
Security Category
Creation Date
Language
Rule
Bulk Change
Select issues
Navigate to issue
53 issues 3h 9min effort
index.ts
Remove this commented out code.
Intentionality
Maintainability
unused
Open
Youngfundish
L7 • 5min effort • 8 days ago • Code Smell • Major
Remove this commented out code.
Intentionality
Maintainability
unused
Open
Youngfundish
L13 • 5min effort • 8 days ago • Code Smell • Major
Remove this commented out code.
Intentionality
Maintainability
unused
Open
asgerbirk
L24 • 5min effort • 1 day ago • Code Smell • Major
Remove this commented out code.
Intentionality
Maintainability
unused
Open
asgerbirk
L68 • 5min effort • 1 day ago • Code Smell • Major


0.0% Security Hotspots Reviewed
8 Security Hotspots to review
Review priority: Medium
Weak Cryptography 8
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Using pseudorandom number generators (PRNGs) is security-sensitive javascript:S2245
Status: To Review
This Security Hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Review priority: Medium
Category: Weak Cryptography
Assignee: asgerbirk
Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity
test/performance/load/load-test-bookings.js
Show 12 more lines
13 },
14 };
15
16 function testPostBookings() {
17 const payload = JSON.stringify({
18 ClassID: Math.ceil(Math.random() * 10),
Coverage 0%
Make sure that using this pseudorandom number generator is safe here.
19 BookingDate: new Date().toISOString(),
20 Status: "CONFIRMED",
21 MemberID: Math.floor(Math.random() * (1027 - 52 + 1)),
22 });
23
Show 21 more lines