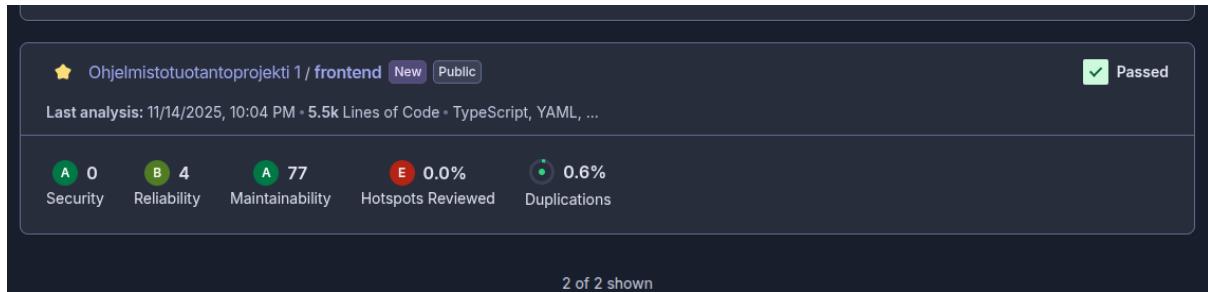
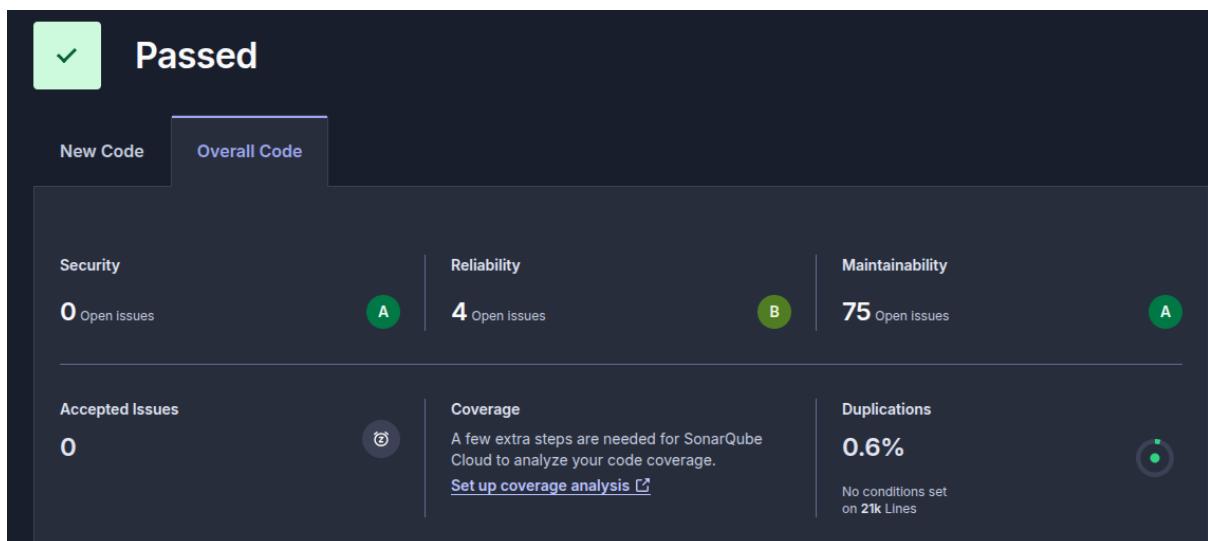


Frontend



The Stats for our application frontend were good from the start and only a handful of issues were found in the code by SonarQube.



One crucial finding that we made was that most of the security issues were the tests using a hardcoded password

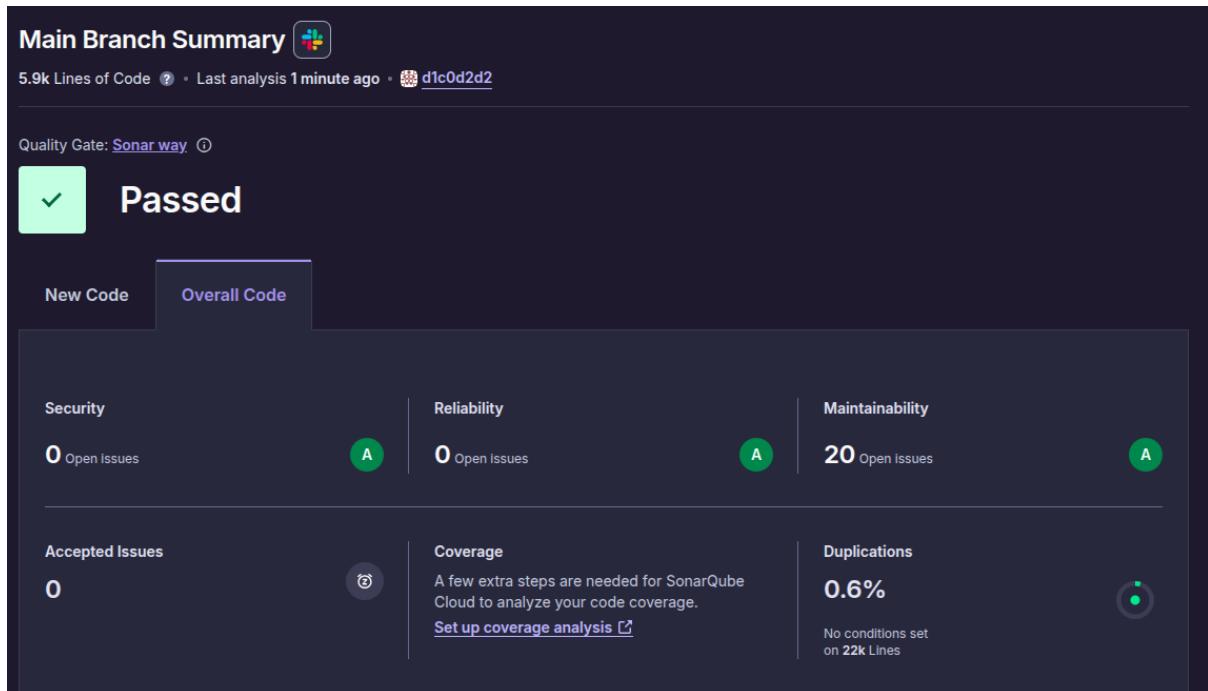
This screenshot shows a specific security hotspot identified in a test file. On the left, a sidebar lists '20 Security Hotspots to review' with a 'Review priority: High' dropdown set to 'Authentication'. The main panel shows a 'Status: To Review' message: 'This Security Hotspot needs to be reviewed to assess whether the code poses a risk.' Below this are tabs for 'Where is the risk?', 'What's the risk?', 'Assess the risk', 'How can I fix it?', and 'Activity'. The code snippet in the center shows a test file with several instances of hard-coded passwords. A callout box highlights one such instance with the text 'Review this potentially hard-coded password.' The code snippet includes lines 72 through 80:

```
const submitButton = screen.getByRole('button', { name: /login/i });
await userEvent.click(submitButton);

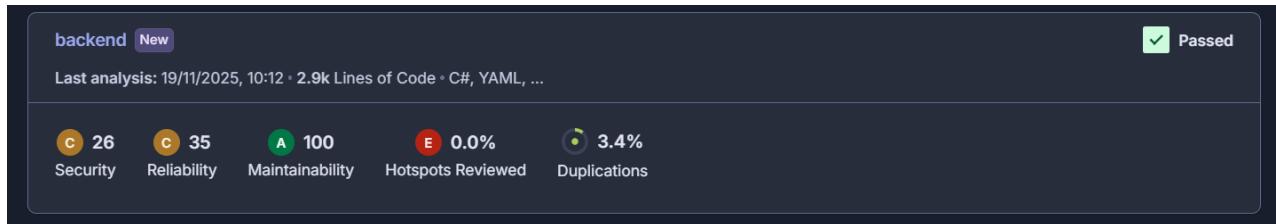
await waitFor(() => {
  expect(screen.queryByText(/login failed/i)).not.toBeInTheDocument();
  expect(mockLogin).toHaveBeenCalledWith({ email: 'test@example.com', password: 'password' });
});
```

Other things noted was the fact that the maintainability issues were of low severity.

And here is the frontend code analysis after cleanup. As we can see some issues remain but most of them are gone and the grade is A in all categories



Backend



The stats for the backend were initially okay, but there had to be some changes made so that it would be safer and more reliable.

Use model binding instead of accessing the raw request data

Use model binding instead of accessing the raw request data

Use model binding instead of accessing the raw request data

```
17
18
19
20
21
22
23
24
25
    [HttpGet("{id}/followers")]
    public async Task<IActionResult> GetFollowers(string id)
    {
        try
        {
            var authHeader = Request.Headers["Authorization"].ToString();
            if (string.IsNullOrWhiteSpace(authHeader) || !authHeader.StartsWith("Bearer "))
```

The security issues consisted mostly of this error on multiple different files. It wasn't a big thing to fix.

The screenshot shows a code review interface with two main sections. The first section is titled "src/application/Controllers/ProfileController.cs" and contains the message "Move 'ProfileController' into a named namespace." It includes a "Reliability" button (Medium), a "No tags" button, and status information (Open, Not assigned, L9, 5min effort, 2 months ago, Bug, Major). The second section is titled "Use model binding instead of accessing the raw request data" and includes buttons for Security (Medium), Reliability (Medium), Maintainability (Medium), and a "asp.net" tag. Both sections show the same status: Open, Not assigned, L23, 5min effort, 2 months ago, Code Smell, Major.

The reliability issues showed the same errors as the security ones and some controllers had to be moved into a named space.

The screenshot shows a static code analysis report for "backend" with a "New" status. It indicates a "Passed" result with a green checkmark. The report summary shows 0 issues across Security, Reliability, Maintainability, Hotspots Reviewed, and Duplications.

After making those necessary changes we eliminated all of the security and reliability issues and also at the same time the maintainability issues were reduced by over 50%.

The screenshot shows a Dockerfile analysis tool. The review priority is set to Medium. A warning is displayed: "Copying recursively might inadvertently add sensitive data to the container. Make sure it is safe here." Another note states: "This image might run with "root" as the default user. Make sure it is safe here." The Dockerfile content is shown with lines 2 through 11. Line 2: WORKDIR /app. Line 3: EXPOSE 5195. Line 5: FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build. Line 6: WORKDIR /src. Line 7: COPY . . Line 8: RUN dotnet restore. Line 9: RUN dotnet publish -c Release -o /app/publish. Line 11: FROM base AS final.

The remaining issues had to do with the Dockerfile and YML file. As a team we decided to take some time and consider if we are going to make the suggested changes.