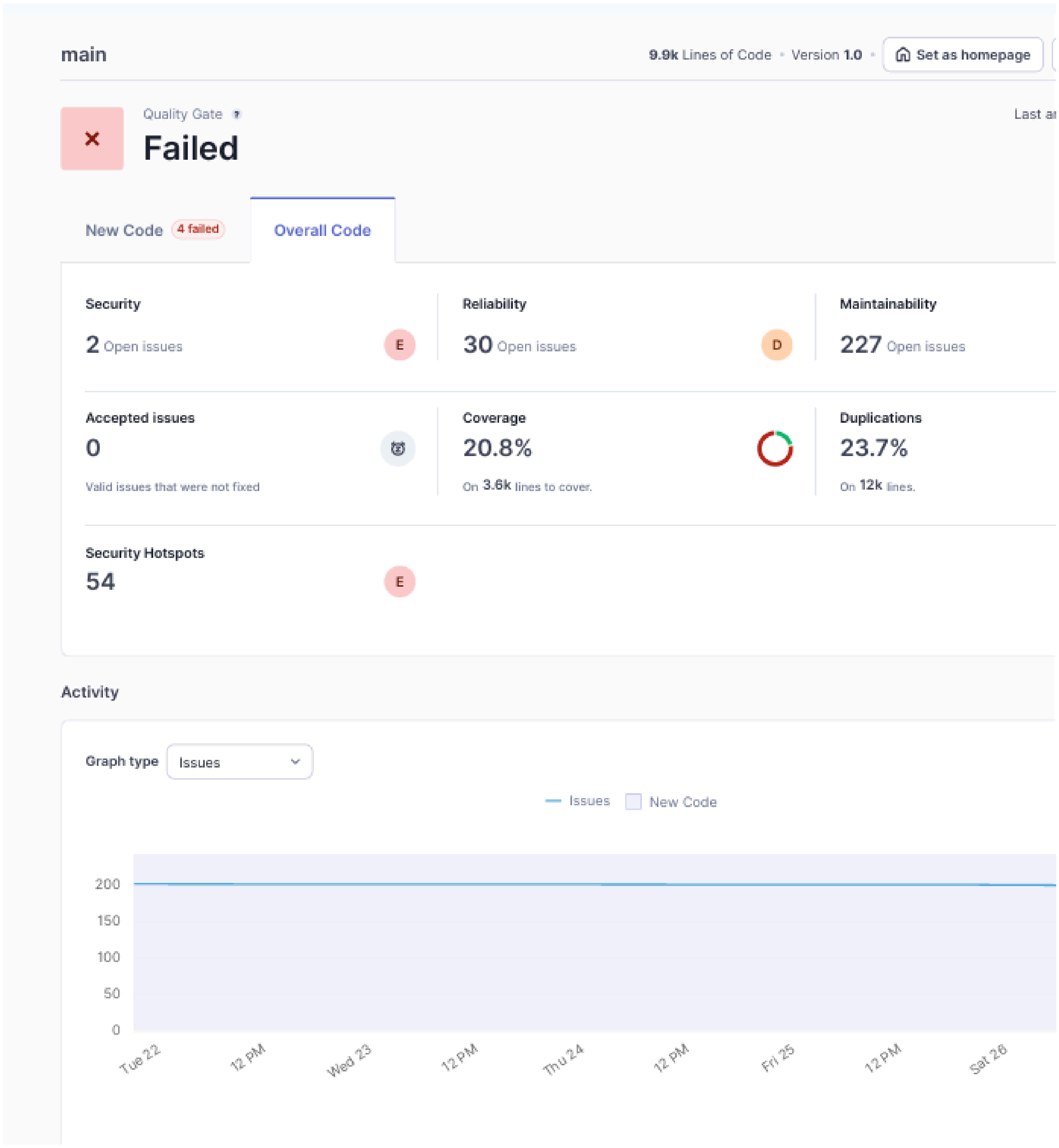
CS4218 Team 08 **Milestone 2**



main

9.9k Lines of Code • Version 1.0 • Set as homepage

Quality Gate

Failed

Last a

New Code 4 failed | Overall Code

Security
2 Open issues E

Reliability
30 Open issues D

Maintainability
227 Open issues

Accepted issues
0
Valid issues that were not fixed

Coverage
20.8%
On 3.6k lines to cover.

Duplications
23.7%
On 12k lines.

Security Hotspots
54 E

Activity

Graph type: Issues

The report above outlines a profound strategy for targeting code coverage improvement, testing expansion, and enhancing maintainability. To improve our code coverage, we have crafted a unique plan below:

1. Increase Unit Testing

The SonarQube analysis indicates a coverage level of 20.8%, significantly below industry standards for applications. We can start by identifying coverage gaps by conducting a more thorough review of files or modules with low test coverage to pinpoint gaps, prioritizing those with important business operations or high user interaction. We can also target critical functions by writing tests that are concentrated on essential functions and methods that impact the application's reliability and user experience, ensuring these components are thoroughly tested.

2. Increase Integration Tests

Integration tests are necessary to verify that modules communicate correctly and that data flows across the application as expected. These tests are particularly crucial given the 3,600 lines of code identified as untested. More tests will hence be designed to cover interactions between modules, validating the coherence and reliability of cross-module data exchanges.

3. Leverage Test Automation Tools

Automating regression testing and aligning it with continuous integration (CI) processes can ensure sustainable coverage as the codebase evolves. Automated Regression Testing allows development and automation of core functional tests, the software can maintain a stable quality baseline, capturing any regressions with each update. CI Integration for Real-Time Coverage Metrics, integrating testing with the CI pipeline will allow real-time coverage tracking and enforce a higher standard for quality with each build.

4. Address Maintainability and Code Duplication

The SonarQube report highlights a duplication rate of 23.7%, signaling a potential area for improvement in maintainability. Addressing these duplications will reduce redundancies and improve code manageability. For example, we can opt to refactor duplicated code**.** To minimize redundancy, duplicate code segments will be refactored into reusable components, thereby reducing code clutter and promoting consistency. Furthermore, we can enhance the modular design for better maintainability**;** large functions or classes will be broken down into smaller, modular components, making the codebase easier to navigate, test, and maintain over time.

Conclusion

We aim to increase the overall and elevate the code's quality, maintainability, and security. Implementing these measures ensures our software becomes more resilient to bugs, facilitating a much smoother user experience with lower critical vulnerabilities.