**CS673 Software Engineering** 

**Team 1 - Spring Alert**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Jim Penner | Team Leader | *Jim Penner* | 06/17/2024 |
| Shenxiao (Sherry) Li | Requirement Leader | *Sherry Li* | 06/17/2024 |
| Chaozheng Zhu | Configuration Leader | *Chaozheng zhu* | 06/17/2024 |
| Jordany Arnaud | Design & Implementation & Security leader | *Jordany Arnaud* | 06/17/2024 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1** | **Jim Penner** | **05/22/24** | **Initial Draft** |
| **2** | **Jim Penner** | **06/09/24** | **Update summary and added S/sheet** |
| **3** | **Jim Penner** | **06/14/24** | **Added table for metrics testing** |
| **4** | **Chaozheng Zhu** | **06/17/24** | **Updated automated testing** |

[Testing Summary](#_heading=h.gjdgxs)

[Manuel Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

[Glossary](#_heading=h.tyjcwt)

# Testing Summary

In this section, you will summarize what was tested, who is involved in testing, testing techniques used, and testing result. You may have the following tests

We used a couple testing technologies in user service and notification service.

1. Chaozheng and Jim used Junit to test notification and user service, implementing testing starting from controllers functions and to other important functions. Also Mockito framework is used often to mock for testing purpose.

2. Jacoco(a springboots plugin) and sonarqube implemented by Chaozheng allows us having dashboard access to see testing detail, code coverage and improvement space

3. Swagg and Postman are used by Jordany and Sherry to manually test if developed API works, can all CRUD request get response as we expected.

* + Unit Testing - JUnit is being used as we are working in Java. ***Jordany has built out user stories ie: registration requests/invalid user account etc built out in project test folder. Also in the UserManagement Service Test a @beforeeach test is run. This triggers the annotated method to be executed before each test method. This establishes any common state or resources needed for individual test cases. This properly initialises before each test case runs. UserInvalid test run as well.***
  + Integration testing - ***Each microservice to be tested in isolation first. This may mean that we address integration testing at Iteration3. We need to identify all common scenarios where the microservices interact. In our case an example is User Registration triggering a notification. Focus on integrating different layers of the app.***
  + System Testing - Postman is used and in our presentation a screenshot of successful connection and email process is displayed.

# Manual Testing Report

In this section, you will give a detailed description of each manual test case performed and the result. If this is a previous You shall list what are existing tests developed in the previous semester and what are new tests developed currently.Postman has assisted with manual testing and built in with security in mind, along the lines of CIA principles. ***This has been started including building out the spreadsheet with all the various testing examples.***

Here is a sample template that can be used for each test case. For system tests or acceptance tests, you may also include some screenshots.

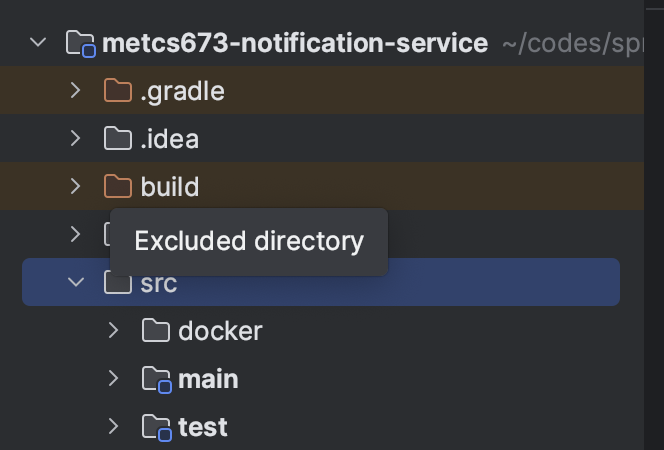
* Test case ID, name
* New or old:
* Test items: (what do you test )
* Test priority (high/medium/low)
* Dependencies (to other test case/requirement if any):
* Preconditions: (if any)
* input data:
* Test steps:
* Postconditions:
* Expected output:
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes: See link below for copy of Spreadsheet of main test details.

(You can use an additional spreadsheet for this section as well) <https://docs.google.com/spreadsheets/d/14g2J5Q4bLN1ECbS6Bc7x15YlCfojdPlg/edit?usp=sharing&ouid=116175152278549798180&rtpof=true&sd=true>

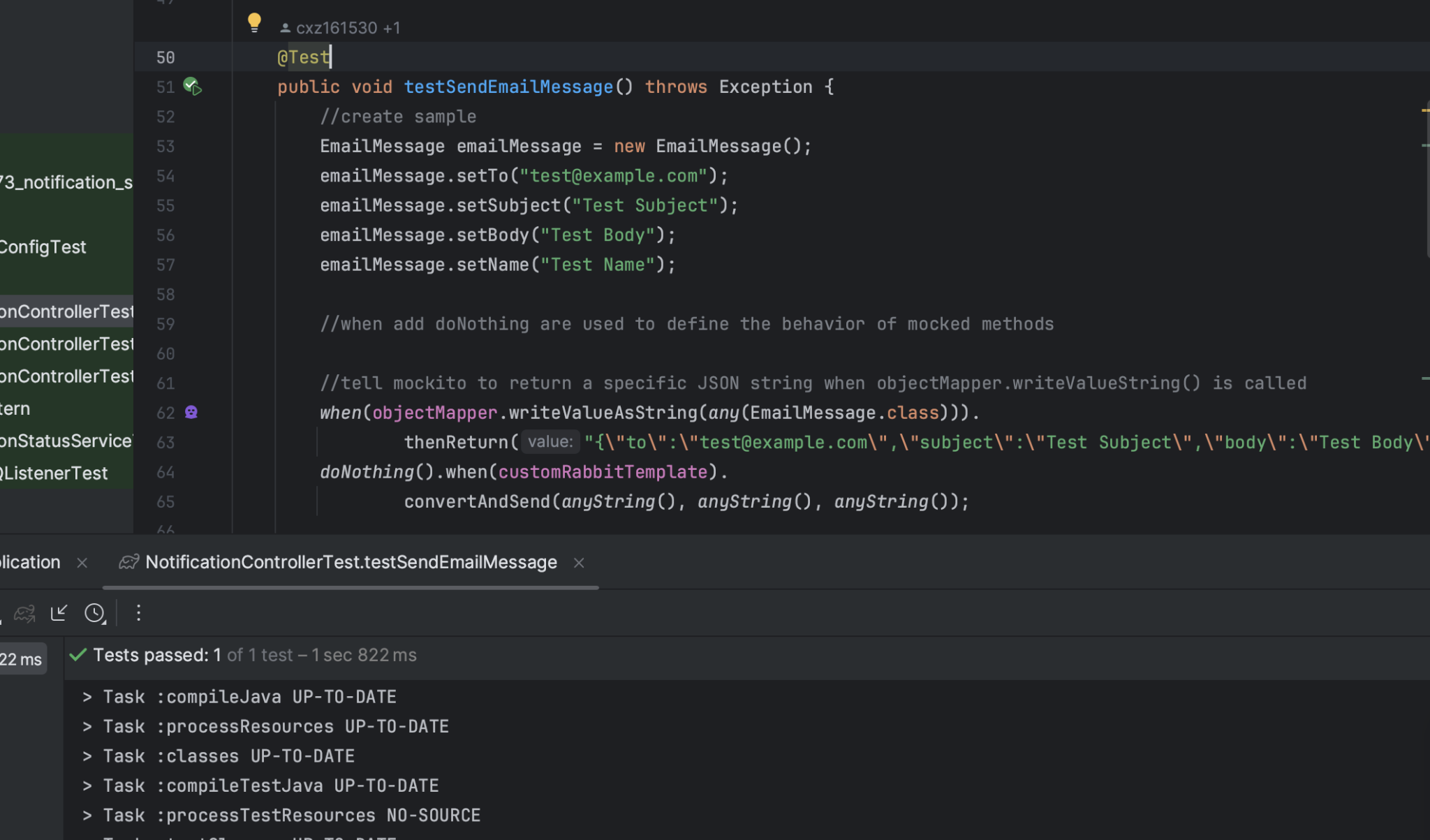
# Automated Testing Report

Describe briefly the automated testing you have done, including where the test code resides in your code repository, what test frameworks are used, and the screen shots or generated testing report.

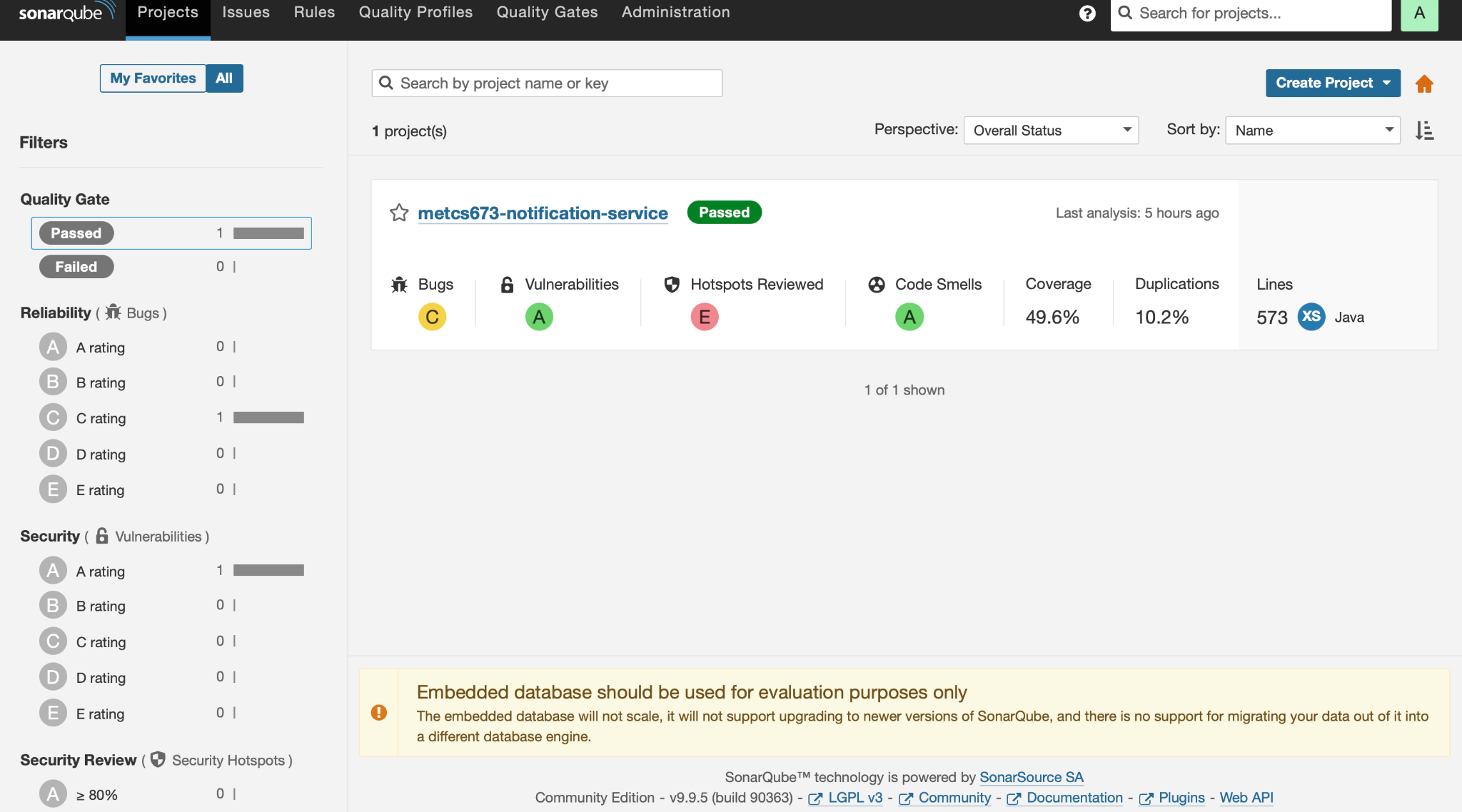
We used Junit ,to doing unit testing, Mockito framework to mock, Jacoco to generate report and Sonarqube is used to perform coverage report.

All testing code residents in here, the test file

Junit:



Sonarqube run all testing cases in project and scan results summary



# Testing Metrics

In this section, you shall report any metrics used for the evaluation, e.g. # of test cases, test coverage, defects rate, etc. - JP to expand on this before Iteration2 submission. Some of the testing metrics that we will cover are:

| Test Type | Notification Service notes (Chaozheng) | User Management notes (Jim) |
| --- | --- | --- |
| Lines of Code | 573 | 420 |
| # of Test cases | 6 | 62 |
| #of Successful test cases | 6 | 51 of 62 tests |
| Number of files | 6 | 69 |
| Memory usage |  | Used memory(Heap): 1171 |
| Javadoc method coverage | 49% | 6.52% |
| Avg Operational Complexity | 1.04 | 1.28 |
| MOOD metrics |  | 104% Coupling factor, 100% Polymorphism, 95% method hiding factor, see screenshot images in Github |

# References

* [A Complete Guide to Unit Testing - Software Automation Testing Consultancy Service (shiftasia.com)](https://shiftasia.com/column/a-complete-guide-to-unit-testing/)
* [Automation Testing in Agile Software Development - Software Automation Testing Consultancy Service (shiftasia.com)](https://shiftasia.com/column/automation-testing-in-agile-software-development/)
* [Automation Testing in Agile Software Development - Software Automation Testing Consultancy Service (shiftasia.com)](https://shiftasia.com/column/automation-testing-in-agile-software-development/#toc0)
* <https://sourcemaking.com/refactoring>
* <https://softwaretestingmaterial.com>
* API Integration testing: <https://www.youtube.com/watch?v=QI-d5Gc6R5g>
* <https://www.jetbrains.com/help/idea/create-tests.html>
* JUnit5 user guide: [junit.org/junit5/docs/current/user-guide](http://junit.org/junit5/docs/current/user-guide)
* [engineering.atspotify.com/2018/01/testing-of-microservices](http://engineering.atspotify.com/2018/01/testing-of-microservices)
* Metrics testing: <https://plugins.jetbrains.com/plugin/93-metricsreloaded/versions>
* Penetration testing: [What is Penetration Testing | Step-By-Step Process & Methods | Imperva](https://www.imperva.com/learn/application-security/penetration-testing/)
* Command injection: [Command Injection | OWASP Foundation](https://owasp.org/www-community/attacks/Command_Injection)
* Security testing: <https://allabouttesting.org/top-10-commands-prompt-commands-used-by-security-experts/>
* <https://www.schneier.com/blog/archives/2019/07/software_develo.html>
* <https://cwe.mitre.org/>
* Sonarqube examples: <https://www.youtube.com/watch?v=2EBRr0wGSwo&list=PLq3uEqRnr_2GuTTkLZL5GU1wZH2FqJRRP&index=18>

# Glossary

* SDL: Security Development Lifecycle
* MOOD: Metrics for Object-Oriented Design
* OWASP: Open Web Application Security Project
* ioT: Internet of Things
* CIA: Confidentiality, Integrity, Availability
* IAAA: In support of CIA (Above): Identification, Authentication, Authorization, Accounting
* DRY - Don’t Repeat Yourself
* ASVS: Application Security Verification Standard