CIS565 - Software Quality Assurance
Final Project Report

1. Title: Implementation of Various Testing Techniques for Software Quality Assurance

2. Contact Info:

- Britta Chowdhury (brittac@umich.edu)
- Sangjun Lee (neilslee@umich.edu)
- Vinutha Gowdru Parameshwarappa (vinutha@umich.edu)
- Vishakha Kumar (vishakum@umich.edu)

3. Project specifications:

3.1 Introduction

As source code in any language is written in text form, it is complex and is hard for humans to read or analyze, especially when the logic is complicated and involves a large number of classes. Tightly coupled source code written by the programmers can run over hundreds of pages and can be extremely hard to understand and unit tested by the Software Quality assurance and testing team.

Application testing and quality assurance is at times more important than the application development itself. Meeting customer requirements and satisfaction solely depends on this phase. Therefore, testing an application as a whole - Front-end UI, backend, database, load balancing and scalability testing, plays an important role. It is important to select and employ one or more of the numerous test tools to assure proper working of the application as a whole and meet customer requirements.

3.3 Project Progress

3.3.1 Unit Testing - Completed

Project:https://github.com/ssoad/BankingSystem

Unit Testing - Writing a piece of code to ensure the quality by verifying every unit of the code. For Unit testing, developers use manual or automated tests to ensure that each unit in the software meets the customer's requirement. In Manual Testing, the tester manually executes test cases without using any automation tool. Here, each stage of the test is executed manually. In Automated Testing, software testing automation tools are used to automate test/test cases. The automation tool can record and save your test and it can be replayed as many times as needed without any further human intervention.

TestNG(Next Generation) is an open source automated testing framework inspired from JUnit and NUnitt,however it introduces some new functionalities that make it more powerful and easier to use. TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration. It supports

dependent test methods, parallel testing, load testing, and partial failure. It supports Flexible plug-in API,Support for multi threaded testing and Flexible runtime configuration. TestNG uses more Java and OO features.
In this section we are aiming to integrate the junit test cases with TestNG and run the test cases as TestNG. Reporting is the most important part of any test execution, as it helps the user understand the result of the test execution, point of failure, and the reasons for failure.TestNG, by default, generates a different type of report for its test execution. This includes an HTML and an XML report output.

JUnit is an open-source unit testing framework used for the Java programming language. JUnit is important in the development of test-driven development, and is one of a family of unit testing frameworks. JUnit Provides annotations to identify the test methods and also provides assertions for testing expected results. JUnit checks the correctness of the test cases and produces a test report, indicating test cases that passed, test cases that failed and some summary statistics.
In this section we are aiming to implement the unit test cases using the JUnit tool for the Banking System which is an open source java project available in github. The aim is to achieve 70% and above coverage to all the java classes in the BankingSystem project. Eclipse IDE is used to write the JUnit test cases for the java classes.

3.3.2 DesigniteJava - Completed

Project - https://github.com/hkristanto/JHotDraw

DesigniteJava is a code quality assessment tool for code written in Java. It detects numerous architecture, design, and implementation smells that show maintainability issues present in the analyzed code. It also computes many commonly used object-oriented metrics. It helps to reduce technical debt and improve maintainability of the software.

Supported implementation smells :

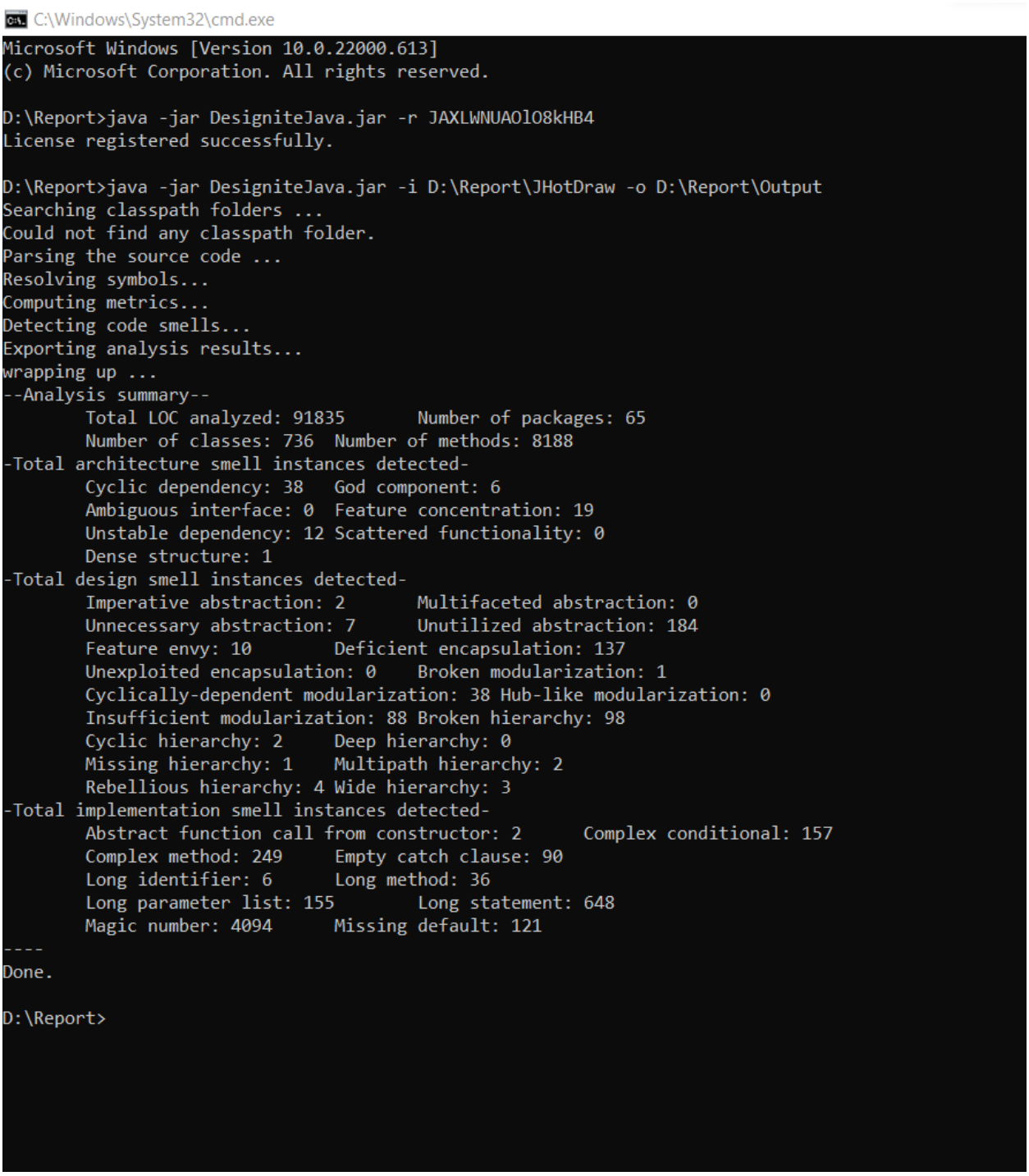
- Abstract Function Call From Constructor
- Complex Conditional
- Complex Method
- Long Identifier
- Empty catch clause

Supported object-oriented metrics:

- LOC (Lines Of Code – at method and class granularity)
- CC (Cyclomatic Complexity – Method)
- PC (Parameter Count – Method)
- NOF (Number of Fields – Class)
- NOPF (Number of Public Fields – Class)
- NOM (Number of Methods – Class)

Supported design smells:

Abstraction Design Smells.



3.3.4 Logging and Tracing using Zipkin and Sleuth - Completed

Project: https://github.com/piomin/sample-spring-microservices-advanced

Zipkin is a Java-based distributed tracing system to collect and look up data from distributed systems such as Microservices. It helps gather timing data and other metrics information needed to troubleshoot latency problems in service architectures. Zipkin features include both the collection and lookup of this data. It lets us track and record requests from their point of origin to their destination and the systems through which they pass. In a distributed system that is developed using a microservices architecture, implementing Zipkin system helps track and understand the application flow from one service to another. A single request can be traced through multiple services, that aids in analysis and efficient dependency management in the project. It provides easy control of the flow of traffic and API calls between services, simplifies configuration of service-level properties like circuit breakers and timeouts.

Sleuth: Sleuth is a tool from the Spring cloud family. It is used to generate the *trace id*, *span id* and add this information to the service calls in the headers, so that it can be used by tools like Zipkin and ELK etc. to store, index and process log files. Spring Cloud Sleuth adds two types of IDs to your logging, one called a trace ID and the other called a span ID. The span ID represents a basic unit of work, for example sending an HTTP request. The trace ID contains a set of span IDs, forming a tree-like structure. The trace ID will remain the same as one microservice calls the next.

The aim in this project is to implement Zipkin and Sleuth in the Source code to and test the various advantages it provides. Logging and tracing information can be used to better analyze, understand the flow, and test the code to enhance the Software quality assurance.

In this project we have implemented Zipkin for tracing using the dashboard to observe and analyze the metrics and code flow in distributed system. Sleuth is implemented at the backend level and helps produce automatic logging for each entry and execution of method with traceID, spanID and other details particular to the executed method.

3.3.5 SonarLint - Completed

Project: https://github.com/piomin/sample-spring-microservices-advanced

SonarLint is an open-source plugin that can be integrated into the IDE - like eclipse and IntelliJ to provide an on-the-fly code analysis and support for hundreds of deep static analysis rules to detect common mistakes, programming errors, tricky bugs, and security issues. Install the SonarLint plugin in the IDE and analyze the code files using SonarLint. This will dramatically improve the code quality by highlighting the programming errors and suggesting the efficient alternative for each of them.

Eg: Unused import statements, null pointer checks, uninitialised variables etc.

3.3.6 SonarQube - Completed

Project: https://github.com/spring-projects/spring-petclinic.git

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality, to perform automatic reviews with static analysis of code to detect bugs, code smells on 20+ programming languages. It is a Code Quality Assurance tool that provides reports for the code quality of the project. It combines static and dynamic analysis tools and enables quality to be measured continually over time. SonarQube is a central server that provides a single platform for all processes which cover full code analysis, measure the Reliability, Security, and Maintainability of all the languages in the project. Sonarqube needs to be triggered by the various SonarQube Scanners. Once the code is scanned it provides a full fledged report on coding formats and standards, duplicated code, unit tests, code coverage, code complexity, code bugs, and security vulnerabilities.

SonarQube integrates with Eclipse, Visual Studio, Visual Studio Code, and IntelliJ IDEA development environments through the SonarLint plug-ins. It also provides easy integration with continuous integration engines such as Jenkins, Azure DevOps, TeamCity, Bamboo etc.

3.3.7 Performance Testing

Performance Testing is one of the software testing procedures mainly accustomed for testing the software speed, response time, stability, reliability, scalability and resource usage under a certain amount of work. To find and eradicate the software performance blockage is the primary goal of performance testing.
Github link: https://github.com/TechPrimers/spring-boot-swagger-exampl

Apache Jmeter- completed

Github link: https://github.com/TechPrimers/spring-boot-swagger-exampl

Apache Jmeter is an open source java software testing tool that is generally accustomed for experimenting and assessing the performance of software applications or services. It performs load testing, performance testing of a web application and can also transcribe a heavy workload by implementing a vast amount of concomitant users to the web browser. JMeter helps testers test the performance of both static and dynamic resources like HTML, JavaScript, JSP, Servlets and ajax.

Jmeter usually reproduces a group of users for sending requests to a target server or application. It returns the statistical view of the performance testing for the application through tables and graphs. In this project, Jmeter is used for two open source rest applications found in GitHub. The objective of this testing is to get correct response through listener after sending request via sampler. Individual thread groups with multiple threads were assigned for individual methods.

Gatling-completed

https://github.com/orangehrm/orangehrm

Gatling is intended for consecutive load testing and participates with the development pipeline. Gatling contains a web recorder and generates visually appealing reports. It is an open source tool written in scala. Gatling makes the load scripts simple and a huge amount of traffic can be tested by Gatling on a single machine and avoids the necessity of a complex framework. Gatling is used with continuous integration tool for load running and it can be verified through the version control system.

3.3.8 UI Testing

User interface (UI) testing is a process used to test if the application is functioning correctly. UI testing can be performed manually by a human tester, or it can be performed automatically with the use of a software program. This section of the project aims to automate the UI test of **WordPress**'s Android app.

WordPress (WP, WordPress.org) is a free and open-source content management system (CMS). Its Android version of the application is also an open source project which started in 2015 and about 170 developers have contributed. The app has been downloaded over 10 million times from Google Play. The project consists of about 330 KLOC in 2,755 files in WordPress-Android\WordPress\src\main folder.

In order to automate UI testing for the application, **Appium** will be used. Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms. Appium facilitates **Selenium WebDriver** to wrap distinct UI Automation frameworks for each platform. The UI Automation framework that Android provides is **UiAutomator/UiAutomator2**. Appium runs a server through which other programs or users can control Appium.

Android Studio is the official IDE for Android app development, based on IntelliJ IDEA. For this project, Android Studio will be used to install and config Android Virtual Device (AVD, Android emulator). Android Studio will also be used for downloading and building the target application package, org.wordpress.android. Appium will use **Android Debug Bridge** (ADB) to communicate with the AVD. A GUI inspector named **Appium Inspector** will be used to inspect UI components. **TestNG** is a testing framework that will be used for writing and executing test cases. TestNG will be installed as an **Eclipse** plug-in.

3.4 Results

In this project we were able to successfully achieve the following:

1. Completed automatic logging functionality using sleuth logging that will auto-generate the logs as soon as a request is sent to a microservice. This will print each logger tagged with Span-Id and Tracing Id.
2. Completed Zipkin tracing, by setting up Zipkin jar, running it in local and viewing the results on the localhost dashboard.
3. Completed static analysis of code within the IDE - IntelliJ using the SonarLint plugin. This shows the code smells, severity and suggestions as per standards to prevent future code bugs.
4. Able to Install and run the SonarQube server on the local machine.

5. Able to Login, Register the service and generate a unique key for authentication. Use this key to analyze/scan the entire project.
6. Able to load the results in the localhost dashboard that provides a visual representation of all the code smells, bugs and vulnerabilities discovered during the scan.
7. Implemented Junit test case for the open source java project and was able to achieve 100% coverage for most of the classes.
8. Integrated the test cases from Junit to TestNG in Eclipse IDE and was able to generate the xml and html result files without any failure.
9. Able to install and run the DesigniteJava using the license and generate the different code smells.
10. Able to install and run the DbFit in the local server and execute the different queries.
11. Completed setting up an environment for UI testing automation using multiple tools
12. Completed generating test cases for Automated UI testing for some specific features of WordPress - Android app (Logging in, Settings, Notifications)
13. Accomplished the performance testing of REST API project with jmeter applying Thread group, sampler and listener successfully.
14. Able to install and run gatling successfully.
15. Record the open source project and save it as a HAR file.
16. Successfully uploaded it on a gatling recorder for generating script and executed it for getting reports with active users, max time,min time.
17. Successfully tested Various API methods with simulation scripts and executed them to get a report/result.

3.5 Challenges

For Zipkin and Sleuth, although the setup and installation was straightforward, one limitation it has is this can be used only on Spring framework and Microservices architecture.

SonarLint was not able to cover all code smells - Eg: unused variable was not detected by the analyser.

SonarQube - Scanning a large project takes time and sometimes the server might crash. The authentication token does not work at times and new token was created in such cases.

For UI testing, composing test cases for an Android App using Appium API was much more time-consuming than expected because it was hard to find UI component identifiers such as resource-id, xpath, etc. manually. Using tools such as Appium Inspector helped save some time, but the lack of readability in UI component identifiers made it not easy. Writing expected results for test case assertion was complex because a single test input may trigger multiple UI components to change at a time.

For DbFit, there was no proper resource available to understand how to connect to the mysql database and execute the test cases. It was difficult to resolve the issues as there are not many resources.

For DesigniteJava , was initially trying to execute the project with the free jar file but was not able to generate any code smells. After analyzing for a long time I was able to request for an academic demo and was able to generate the different types of code smells.

For Gatling, It is totally script based tools and a code based tool, so it is difficult to understand all the code flow for someone who is new to gatling. Another challenge is lack of supported materials for understanding this tool in a better way.

3.6 Conclusion

Implementation of this project helped us understand and gain hands-on experience on all the concepts learnt during the coursework and much more. It was a great learning experience to explore multiple tools available for testing and quality assurance of an application. It helped us broaden our perspective and skillset in the area of quality assurance. It helped us make an informed decision on which tools and technologies to apply for a particular type of testing in future.

3.7 References:

- "Introduction and Importance of Software Testing in SDLC." http://www.onestopsoftwaretesting.com/introduction-and-importance-of-software-testing-in-sdlc/. Accessed 10 Feb. 2022.
- "Software development process - Wikipedia." https://en.wikipedia.org/wiki/Software_development_process. Accessed 10 Feb. 2022.
- https://www.lambdatest.com/blog/9-of-the-best-java-testing-frameworks-for-2021/
- Habib A, Pradel M (2018) How many of all bugs do we find? a study of static bug detectors, IEEE, https://0-dl-acm-org.wizard.umd.umich.edu/doi/10.1145/3238147.3238213
- "Database Testing Types and its Best Tools | Quick Guide - XenonStack." 28 Dec. 2021, https://www.xenonstack.com/insights/what-is-database-testing. Accessed 11 Feb. 2022.
- https://zipkin.io/
- https://spring.io/projects/spring-cloud-sleuth
- https://www.loginradius.com/blog/async/sonarqube/
- "Automated User Interface Testing · Devbridge." https://www.devbridge.com/articles/automated-user-interface-testing/. Accessed 27 Mar. 2022.
- "WordPress for Android - GitHub." https://github.com/wordpress-mobile/WordPress-Android. Accessed 27 Mar. 2022.
- "Introduction - Appium." https://appium.io/docs/en/about-appium/intro/. Accessed 27 Mar. 2022.
- "Meet Android Studio." https://developer.android.com/studio/intro. Accessed 27 Mar. 2022.
- https://en.wikipedia.org/wiki/EvoSuite
- https://www.loginradius.com/blog/async/sonarqube/
- https://www.youtube.com/watch?v=lqQo8-9NmY8
- https://en.wikipedia.org/wiki/SonarQube
- https://www.designite-tools.com/designitejava/
- https://en.wikipedia.org/wiki/TestNG