1. Project subject selection process

I decided to focus on JAVA-based projects, to ensure a common understanding of the general programming language running in the project. This was also done to keep the bug detection tools consistent with every project and work together with the tools and see the differences in each project and see what kind of bugs the tools will detect, in the same category of the projects. The projects selected are very close to some of the real-life examples I would see including an e-commerce website, desktop application and JAVA application. When there was difficulties with building and running one of the projects, another was chosen instead which was able to be run and consistent with the other projects chosen.

2. Project details

Project 1 : Inventory management system
Project description: This Java-based desktop supply chain management application for stores, offers a quick and efficient solution to keep a record of products, suppliers, sales information and customers and their purchase details. Moreover, the application records currently available products in the store. This application provides both admin and customer access, with the key difference that only the admin can access and view the sales report and manage other users. **Project link:** https://github.com/sazanrjb/InventoryManagementSystem **Stars and forks:** This project has 123 forks and 187 stars.
Language used in the project: This application is built in JAVA swing (Frontend), and JAVA (Backend) programming language. it utilizes MySQL as its underlying database system.
The number of classes/packages: The inventory management system is built using a Model View Controller (MVC) architectural style and Data Access Object (DAO) design pattern. Data obtained using DAO are stored in Data Transfer Object. It contains 5 packages and 26 classes.

Project 2 : Online BookStore
Project description: A user-friendly Online Bookstore project in which users may log in or register, examine available books, pick books and quantity, and purchase them. Users can also receive payment receipts after making a successful payment. The administrator may also utilize the project to add new books, remove books, raise, and reduce the number of books, adjust the price of the books, and keep the selling history of books.
Project Link: https://github.com/shashirajraja/onlinebookstore **Stars and forks:** 287 stars and 1.4k forks.
Language used in the project: Front-End Development: HTML, CSS, Javascript, BootStrap Back-End Development: Java [JDK 8+], JDBC, Servlet, Database: MySql.
Number of classes/packages: 1 Main/Java package having 7 sub-packages. With 27 Classes in total divided into the packages.

Project 3 : Hotel-Management-OOP-Project
Project description: This is a Hotel Management program that can be used to handle activities such as saving client information, reserving four various types of rooms, ordering meals for specific rooms, unbooking rooms, and displaying the bill. It may also be used to view various room characteristics and availability. It is a menu-driven programme that continues till the user closes it. Once the programme quits, file handling is used to save the current state of the hotel (client data, booked rooms, food ordered) in a file so that the old details are not lost when I resume the

programme. When the programme resumes, it scans the file to determine the prior status of the hotel. If the user attempts to book an already reserved room, a user-defined exception is thrown. This project is using Inheritance, File Handling with Objects, ArrayList, implementing Interface, User defined exception and Exception handling.
Project Link: https://github.com/shouryaj98/Hotel-Management-Project-Java/tree/master
Stars and forks: 337 forks and 248 stars
Language used in the project: This project is built using JAVA language and is a java based application. The project's design is built on a basic command-line interface (CLI) format, in which users interact with the programme via text-based input and output on the terminal.
Number of class/package: This project only has one default package which includes multiple classes (nested classes) like food, Doubleroom, holder, Hotel, SIngleroom, NotAvailable etc in Main.java.
Number of classes/packages: This project only has one default package which includes multiple classes (nested classes) like food, Doubleroom, holder, Hotel, SIngleroom, NotAvailable etc in Main.java.

Project 4 : Train Ticket Reservation System
Project description: This Java Project is for a Web-based application of a Train Ticket reservation system, accessed either by an Admin user or a Regular user with different functionalities for each. The basic features for users include viewing trains and their schedules, searching for trains based on their train number and booking train tickets. The features for the admin user include adding/removing trains and updating train information. Users can also create new user accounts with a username and password and are able to update their password as well. Train information, User profiles, Admin profiles, and User History are stored in an Oracle SQL database.
Stars and forks: This project currently has 129 Forks and 87 Stars.
Language used in this project: This project uses an MVC(Model View Controller) Architectural style with Java classes communicating with the HTML code on the website. Java for the Backend, HTML and CSS for the Frontend.
Number of classes/packages: 6 packages, 54 classes

Project 5 : Online Shopping Cart (E-commerce website)
Project description: This is java-based E-commerce website in which an electronic store is created and a user can visit the store, check the products that are available which can be added to the cart and proceed to checkout for payment to take place. Some of the features include admin functionality that can remove, add, and update any products in the store and also see the status of items whether they have been shipped or delivered. Another feature is the email messages for confirmation and registering on the website once the person orders a product or registers on the website they will receive a confirmation mail.
Project Link: https://github.com/shashirajraja/shopping-cart
Stars and forks: 79 stars and 102 forks
Language used in the project: Front-End (HTML, CSS, Javascript, Bootstrap), Back-End (Java[JDK 8+], JDBC, Servlet, JSP) Database - (MySQL) **Number of classes/packages:** 6 packages, 37 classes.

3. Bug detection tool selection overview

For this project, I have selected 3 static bug detection tools.

1. FindBugs
2. PMD
3. SonarQube

These tools are static analysis tools and help us find general bugs and code smells in the code.

For Easy catching and identification: I wanted to get easy and basic bug detection first to see any major issues in the syntax. Findbugs is able to provide "warning" in 4 different categories to help us determine the severity of it which makes it easy to identify false positives as well. (Link:https://www.methodsandtools.com/tools/findbugs.php#:~:text=Findbugs%20is%20an%20open%20source,the%20code%20for%20the%20analysis.)

For Code style violations and performance issues: I wanted to check for bugs like code smells or any code writing style violations, so I used PMD to make sure that the best coding standards have been applied to the project while also checking the presence of any performance issue as PMD scans the source code instead of scanning the java bytecode.
(Link: https://pmd.github.io/ ,https://www.cs.cmu.edu/~aldrich/courses/654-sp07/tools/hsu-pmd-07.pdf)

For Security checks: I wanted to select a tool that would find any security bugs in the projects. Also, I wanted to test out the reliability and also long-term support maintenance of the project. I found the perfect tool for the job, SonarQube which is able to test everything statically but also provides almost all the different types of analysis going forward and unlike many tools had greatly detailed report generation to understand.

4. Selection Process

I started by making a list of any static analysis tools that worked for Java and the Windows/Mac OS operating systems which I are working on. Our initial list included FindBugs, Klocwork Insight, PMD, Daikon and Coverity. I then proceeded to try each of the tools on our respective projects and systems. I quickly learned that Daikon is intended for Linux-based systems and decided to choose another tool instead. I also discovered that Klocwork Insight is a paid tool intended for corporate use so in the end, I decided on FindBugs and PMD as they are easy to use and have plugins on both Eclipse and IntelliJ. I also found SonarQube to be best for testing the security rules of our projects and proceeded to use that as well.

Tool	Description
Findbug	Findbugs scans the bytecode for the projects so having any issues in the generated bytecode gets "caught". It is also able to provide "warning" in 4 different categories to help us determine the severity of it. FindBugs checks the code based on multiple categories like correctness, malicious code vulnerability, security, and Dodgy code. FindBug plugin in Eclipse exports the bugs detected by FindBug in the XML format. However, it is quite hard to read the reported bugs and compare them with actual code. IntelliJ also provides a FindBugs plugin using QAPlugs which generates the report in HTML format which is easy to read and follow along with the code.

PMD	PMD is a static source code analyzer, that finds common programming flaws such as unused variable/object creation, and empty/unnecessary use of try-catch. PMD reporting is quite similar to FindBugs as it mainly lists the numbers of bugs found in text format, it provides detailed information about each rule that was broken with reference to the class and line of the code it was found. PMD reports each bug in 5 different categories which helps determine the bug severity.
SonarQube	SonarQube is a static analysis bug detection tool which analyzes software for overall code quality and security standards. It provides Quality Gates and a detailed, wellformatted report on its analysis of the software.

5. Tool application process

Project 1 (Chirayu): While importing the project I faced the DataBase related issue. The reason was that the JAR provided in the project import description was not up to date with the MySQL version. I solved this problem by importing "mysql-connector-j-8.1.0", and successfully imported this project. Another issue I faced was that initially, I was using the IntelliJ IDE to run my project, after adding the FindBug plugin from QAPlug. I run the FindBug detection tool, However, it didn't report any bugs. So, I decided to try running the same bug detection tool on Eclipse IDE to confirm the result. Then I run into another issue as I was using the community version of Eclipse and I was not able to add the bug detection tool successfully. So, I switch the Eclipse IDE from the Community to the Enterprise version, and I was able to install the FindBug into Eclipse Enterprise Version. I run the FindBug in Eclipse IDE and I obtain the same result as IntelliJ (0 bug reported). Also, I asked my group members to run my project on their end and generate the FindBug report. However, they also got the same results as me. Another issue I faced is with the SonarQube tool installation. After trying different methods and following different tutorial videos I was not able to run sonarQube on my Mac machine. I decided to run the tool on Saksham's machine as he was able to make this detention tool run on his Windows machine.

Project 2 (Saksham): I found difficulties in setting up the project with the Tomcat server as there was a specific version requirement and the same was with MySQL starting and stopping the server with workbench. Another problem was the correct PMD plugin to use on the project as there are multiple versions of it on Eclipse marketplace. SonarQube also caused a lot of setup issues as the report-generating plugin required LTS versions of SonarQube (By CNES report) in order to run.

Project 3 (Prabhkirat): Difficulties faced was trying to use the correct version of Java with the plugins as PMD required a higher version of Java than FindBugs to run on and constantly gave errors. Eclipse enterprise version also didn't run Findbugs correctly due to some internal errors so had to revert to a version capable to run with Java 8 only. Sonar Scanner and SonarQube had issues running on my system as I had to manually set environment variables for them. Trying to run SonarQube with Maven caused errors for not being able to run it if not configured properly.

Project 4 (Luke): After correctly importing the project, installing all dependencies, and setting up/building the project, I began applying FindBugs to the entire scope of the project. While very easy to run, at first, there was no output from the FindBugs tool, even after showing the FindBugs views in Eclipse. After changing the configuration of Findbugs to include all bug categories in the reporting and setting the minimum severity level to include warnings, I was able to detect reported

bugs in the FindBugs views. SonarQube required much more setup and troubleshooting as I had to switch to Java 17 to run it, download specific versions of SonarQube and SonarScanner, and set up the project on the SonarQube localhost interface. Once I was able to successfully run the tool, I downloaded the report and could view the analysis easily.

Project 5 (Rajat): Project 5 was an online shopping cart E-commerce website based on Java as the main language and other Front-End and Back-End supporting languages for the website. Findbugs was not working on Eclipse as it needed the Eclipse enterprise version to run FindBugs. So, after downloading the enterprise edition for Eclipse and importing the project while downloading JDK[8+] and MySQL for the database and TOMCAT Apache for running the website on a local host. I downloaded the extension for FindBugs and PMD and generated the static code reports for bug detection in my project. Another difficult part was setting up SonarQube for the project as it was needed to run on JDK[17] and running the code on localhost through SonarQube which generated whole bug report all through by running the commands on PowerShell.

6. Evaluation results

Project	FindBug	PMD	SonarQube
Project 1	0 bugs	2138 Violations (29 blocker, 10 critical, 2042 urgent, 57 important)	2 bugs (1 major, 1 blocker) 522 Code Smells (243 minor, 242 major, 13 critical)
Project 2	11 bugs (1 scary possible null pointer)	1089 total (17 Blockers,11 Critical,1056 Urgent,5 important)	26 (5 minor, 21 major) 100 code smells (17 minor, 28 major, 55 critical)
Project 3	6 bugs (1 scary and high confidence)	760 total (24 Blockers, 192 Critical, 530 Urgent, 14 important)	7 bugs (4 minor, 3 blockers) 140 Code Smells (22 minor, 98 major, 20 critical)
Project 4	65 Bugs (11 scary, 54 of concern)	1063 Violations (32 blockers, 20 critical, 1010 urgent, 1 important)	43 bugs (6 minor, 23 major, 14 blockers) 140 Code Smells (45 minor, 31 major, 63 critical)
Project 5	21 bugs (1 scary, 20 of Concern)	1240 violations (0 blockers, 12 critical,1228 Urgent, 0 important, 0 warning)	83 bugs (18 minor, 54 major, 11 blockers) 108 Code smells (16 Minor, 60 Major, 32 Critical)

7. Manual bug evaluation

As I are using the same tool for all our projects to compare the static analysis.

For PMD, I found a similar pattern from each project analysis report that PMD reports the 5 types of violations (blocker, critical, urgent, important) each of these violations are not actual bug. PMD reports style errors and bad practices. Such as “comment required”, “method naming convention”

or “At least one constructor”, none of these violations cause bugs when running the code but are simply recommendations for readability and applying best practices.

For the Train Ticket Reservation System project, FindBug flagged the cross-site scripting vulnerability for certain classes. After observing this I tried running the cross-site script “<script>alert(“Attack”)</script>” on the running project and I was able to produce the bug on every text input field. Therefore, I can say that FindBugs can discover important bugs that was not otherwise detected by the other tools.

8. Bug reporting process

For listing any type of issue or bug to the developer of the source code, I can go to the GitHub repository and on the top of the main page, in the issues section, I can create a new issue. Under that, I can select the Bug report option, enter the title, and details of the bug with steps to reproduce it and submit the files or screenshots to better illustrate the issue and submit it, which can be reviewed by the developer. Either I can fix the issue and push the code or give him the necessary information for fixing the bug.

9. New bugs reported

1. FindBug reported a security vulnerability in Train Ticket Reservation System. Using this report I tried performing Cross-Site Scripting injection "<script>alert("ATTACK")</script>" on text input boxes. This injection did produce a security bug.
2. Online Shopping Cart (E-commerce website) and Train Ticket Reservation System projects are from the same repository. So, I decided to perform a Cross-Site Scripting injection on Online Shopping Cart, and I found a security vulnerability.
3. The inventory management system had a hard-coded SQL query to authenticate the login. This query authenticates only one type of user.

10.Bug detection tool analysis

Advantages and disadvantages of using PMD: PMD checks the code quality and provides the best practices for writing clean code, which helps developers to improve the code quality. It is easy to integrate the PMD add-in in JAVA IDEs such as Eclipse or IntelliJ. Another advantage of PMD in Eclipse is that it allows users to save the PMD analysis report in .txt format, which provides information about the class and line number of the code where the rule was violated. However, most bugs reported in the PMD are technically false positives as it checks the typical programming mistakes such as unused variables or unnecessarily created objects as well as empty try-catch blocks, the PMD report had all bugs similar to this.

Advantages and disadvantages of using FindBugs: FindBugs detection tool helps to identify potential bugs and security vulnerabilities. In terms of potential bug detection FindBugs provide vulnerabilities related to programming errors, null pointer, wrong string comparison issues, etc. Moreover, FindBugs also points out security-related bugs. For the Train Ticket Reservation System project, FindBugs did point out the XSS (cross-site scripting) vulnerabilities and I was able to produce the bug. Same as PMD, the FindBug add-in is easy to integrate into JAVA IDEs (Eclipse,

IntelliJ). Even though FindBugs reports fewer false positive bugs compared to PMD. However, the bug report generated by FindBugs is in XML format which is quite hard to read and understand.

Advantages and disadvantages of using SonarQube: The advantages are that it allows for seamless integration as it can be used outside of any IDE and requires a separate environment and gives a greatly detailed and comprehensive dashboard for understanding the reports. It can be automated to scan at regular intervals and allows customization of coding rules and profiles that would be required by the developer.
The disadvantages that I faced was a very complex setup process and high learning curve. Also, being a static analyzer, it can still produce false positives and false negatives. Being an independent tool, it is very resource intensive too as it requires a local server spin-up on localhost to view bug reports and analyze them.

11.Biggest issues with each tool

FindBugs and PMD: Both FindBugs and PMD plug-ins work on the enterprise edition of Eclipse. Also, both tools are static bug detection tools that provide false positive bugs and generate lots of recommendations and warnings that may not be useful to the developer. Also, in FindBugs I need to turn on certain options such as security, and malicious code vulnerability options to improve bug detection.
SonarQube: If the source code is lengthy and complex, setting up SonarQube will be more challenging. The open-source SonarQube version has few features, which makes maintenance challenging. Additionally, SonarQube is a resource-intensive tool that uses more processing power if the code is large and complex, so using it might have an impact on the project's growth and slow down testing.

12.Hardest project to test

Project 5 was the hardest in terms of setting up the tomcat apache server and setting up the MySQL database server to connect the website to the local host as MySQL8.0 had issues with starting the MySQL server from MySQL workbench as it is a known bug in the workbench due to which I had to start the server from window services manually and initialize with all the SQL commands. The same issue was also faced with one of the other SQL projects but was quickly troubleshooted once Project 5 was up and running. To detect bugs for this project, I had to manually direct SonarQube to look for ".class" files because it was deep inside one of the package's subfolders. In Project 5, as the project was based on an online e-commerce website, it had the important functionality of sending a confirmation email whenever a product was ordered or a new user that was registered. So, for the mailing functionalities, I had to generate a16 digit password under the name of "electronic cart" so that the username, email, and password values was changed for MySQL credentials for the website to send the confirmation email.

13.Conclusion

Overall, I was each able to successfully run our selected bug detection tools on each of our projects, analyze the detected bugs found by each tool, and generate reports from said tools. After examining the outputs of every tool, I was able to compare their respective setups, how each tool reports bugs and which types of bugs was reported, as well as compare the generated reports by each tool. While

the most difficult to set up and configure, Klocwork Insight proved to offer the most comprehensive bug reports which was well organized, easy to read, and included charts on the bug metrics. PMD mostly reported style and bad practice errors and was the least useful in finding bugs in the software. While FindBugs found the least number of bugs, it did help in uncovering security vulnerabilities in two of the projects that was not reported before. Using these tools consistently with all our projects helped us better understand the differences between these bug detection tools and how they can be easily applied to software to find a wide variety of bug types.

14.Suggestions for others

1. Try to keep the environment constant with project requirements.
2. Always read the documentation for the bug detection tools and the project.
3. Try to use tools that are most established and have a good source of resources.
4. Use tools that are updated frequently.
5. Use manually written test cases to avoid false positives.
6. Try to categorize bugs in order of priority and severity.
7. Make sure to analyze false positives to better tune the tools for accuracy.
8. Try to use different and trusted bug detection tools on the code so that different areas of the code can be covered.
9. Using so many bug detection tools of the same type may lead to confusion and duplication of errors which could be time-wasting.
10. Warnings should not be ignored every time but carefully analyzed manually so that major blockers could be corrected.