UNIT IV – TESTING AND QUALITY IN SOFTWARE [9 hours]

Types of Testing: Unit, Integration, System, Automated Testing using JUnit or PyTest, Introduction to Selenium (UI testing basics), Code Quality Tools: SonarLint/SonarQube basics , Importance of Testing in Real Projects

---

CODE QUALITY TOOLS - SONARQUBE

SonarQube is a very powerful platform that plays an important role in the quality and maintainability of software projects, by providing a highly integrated set of tools for continuous inspection of source code, so the problems are detected and solved early in their development.

SonarQube identifies bugs, vulnerabilities in the code through source code analysis. It analyzes the code in terms of its quality. The tool generates a detailed analysis report of the code within the SonarQube server. It performs full scanning of the whole code for bugs, duplications, security vulnerabilities, security hotspots, and bad written statements of the code.

SonarQube scans several programming languages, such as Python, Java, and many more. This tool is deployed by multiple organizations for finding bugs and problems with the code. SonarQube allows a CI/CD pipeline to get the reports automatically generated on code analysis.

Importance of SonarQube:

SonarQube is a powerful platform, playing a critical role for both the quality and maintainability of software projects. It enables developers to find issues right from the start of the development process, which in turn changes the quality of the corresponding code base very quickly. SonarQube is an essential tool for modern software development teams because of

- Improved Code Quality
- Enhanced Code Maintainability
- Improved Security
- Enhanced Collaboration
- Increased Productivity
- Compliance Adherence
- Risk Mitigation

Editions of SonarQube:

- **Community Edition:** This is the basic version, perfect for those starting their journey with code quality in CI/CD pipelines.
- **Developer Edition:** This edition provides enhanced application security and derives maximum value from SonarQube across different branches and pull requests.

- **Enterprise Edition:** Ideal for managing a large application portfolio, it helps enable code quality and security at an enterprise level.
- **Data Center Edition:** Designed for high availability in global deployments.

Installation of SonarQube:

Installation of SonarQube can be done in two ways: through a ZIP file or a Docker image. Here's how to do both:

A. Installing from a ZIP File

- **Download the Community Edition:** Get the ZIP file from the SonarQube Downloads page.
- **Unzip the File:** As a non-root user, extract the ZIP to a location of your choice
- **Start the Server:**
  - On Windows: Navigate to 'C:\sonarqube\bin\windows-x86-64\' and run 'StartSonar.bat'.
  - On other operating systems: Execute '/opt/sonarqube/bin/[OS]/sonar.sh console'.
- **Access the Dashboard:** Open your web browser and go to [http://localhost:9000] Log in with the default credentials: admin/admin.

B. Installing from a Docker Image

- **Find the Community Version:** Go to [Docker Hub's SonarQube page](https://hub.docker.com/_/sonarqube/).
- **Run the Server:** Execute the following command in your terminal:
  docker run -d --name sonarqube -p 9000:9000 <image name>
- **Log In:** Just like with the ZIP installation, access [http://localhost:9000] with the credentials admin/admin.

Analyzing a Project with SonarQube

After login to SonarQube, the project is analyzed as follows:

- **Create a New Project:** Click the Create New Project button.
- **Choose Creation Method:** Select Manually when prompted.
- **Set Up Your Project:** Enter a Project Key and Display Name, then click Set Up.
- **Generate a Token:** Under Provide a Token, select Generate a Token. Name your token, click Generate, and then hit Continue.
- **Choose Your Language:** Select your project's main language under Run analysis and then analyze the code. When Maven or Gradle is used, the Scanner will be automatically downloaded.
- **View Results:** After the analysis completes, Project's first report can be viewed on SonarQube.

Architecture and Integration

The key components in SonarQube

1. SonarQube Server:

- Web Servers:Allow developers and managers to browse quality snapshots and configure SonarQube instances.
- Search Server: Powered by Elasticsearch, it supports UI searches.
- Compute Engine: Responsible for processing code analysis reports and saving them in the database.

2. SonarQube Database:

- Configuration of the SonarQube instance.
- Quality snapshots of projects and views.
- Installed plugins, including those for different languages and integrations.

3. SonarScanners: These run on your CI servers to analyze projects.

How SonarQube Works?

SonarQube is the platform that continuously reviews source code quality. It scans the source code to find any problematic issues, then reports actionable information directly to the developers who developed the code.

1. Source Code Analysis

- Integration with Version Control Systems: SonarQube can integrate with popularly used version control systems like Git, SVN, and Mercurial.
- Code Scanning: When there is a commit of code changes into a repository, then SonarQube analyzes automatically for issues in the code.
- Analysis Engines: The analysis engines used by SonarQube include:
  - Static Analysis: It analyzes the code without its execution and identifies potential errors, vulnerabilities, and code smells.
  - Code metrics, such as cyclomatic complexity, code coverage, and technical debt, are computed.
  - Security analysis identifies the possible security risks.
  - Quality gates define how much quality code should assure before merging or releasing into production.

2. Issue Detection and Reporting

- Issues Identified: SonarQube detects several issues such as:
  - Bugs: A potential error in the code.
  - Vulnerabilities: Potential security flaws that can be exploited.
  - Code smells: Signs of poor quality or maintainability of the code.
  - Hotspots: Fragments of the code that must be payed attention to because they are complex or have technical debt.
- Reporting Issues: The software provides detailed reports of the issues found such as:
  - Issue Description: A good description of the problem.

- Severity: The degree of an issue (minor, major, critical).
- Location: Exact location in the source code.
- Proposed Solutions: The suggested solutions to this problem are.

3. Quality Gates

- Customizable Quality Gates: SonarQube provides an ability to define custom quality gates - thus defining the minimum acceptable levels for various metrics.
- Project Evaluation: SonarQube evaluates the project based on these quality gates, which means there is a clear indication of the overall quality of the project.

4. Visualization and Dashboards

- Project Dashboards: SonarQube also offers customizable dashboards, which help display key metrics and trends for individual projects.
- Global Dashboards: You can also create global dashboards that help you track the overall quality of your codebase.

5. Integration with Other Tools

- CI/CD Integration: It supports integration with CI/CD pipelines to auto-run code quality checks.
- IDE Plugins: The development teams can leverage the plugins through the popular IDE to easily access SonarQube.

SonarQube and DevOps

SonarQube is another interesting DevOps tool since it offers a common platform for continuous source code quality inspections. Integration of SonarQube in the context of the practice of DevOps improves all the processes involved in developing software and delivers better products.

Benefits of SonarQube with DevOps teams:

- **Continuous Integration (CI):** SonarQube integrates very smoothly with CI pipelines for one to realize analysis of changes in code coupled with immediate feedback on quality of code.
- **Continuous Delivery:** In CD, SonarQube provides potential errors early in the development cycle so that only the best code reaches the production environment.
- **Quality Gates:** SonarQube can be set to define quality gates that must be met before the code could be allowed to merge or release; this means only such code meeting predefined quality standards gets promoted to higher environments.
- **Risk Mitigation:** Software risk mitigation also reduces the risks of failure and disturbance beforehand by detecting the potential problems.
- **Improved Coordination:** SonarQube has a common platform where teams can share metrics of code quality, thus supporting team coordination toward continuous improvement.

Benefits of SonarQube Source Code Coverage

This too scans source code continuously. Its coverage on the source code will help you know the percentage of your code that has been executed in the testing. That makes this a really useful metric for showing the efficiency of test suites and identifying areas of vulnerability to bugs or security vulnerabilities.

1. Improved Code Quality

- Identification of Untested Code: Identification of Untested Code High coverage gives you a guarantee of the fact that all your code is tested, leaving little chance for any undetected bugs or errors.
- Targeted Testing: Targeting low coverage areas, you can ensure priority testing while putting minimal emphasis on improving the code.

2. Enhanced Code Reliability

- Early Detection of Issues: With high coverage, the possibility of catching problems at the earliest development stage as a potential flaw, they don't then become severe defects, is high.
- Reduced Risk of Failures: A more thoroughly tested codebase will fail less in production and interrupt fewer customers, making it much cheaper.

3. Increased Confidence in Codebase

- Objective Measurement: Objective Measurement Source code coverage is an objective measure of the quality of code that can be delivered for smooth operation with high reliability and stability of the codebase.
- Improved Collaboration: Thus, shared understanding of code coverage will likely help people in a team collaborate better and ultimately end up with better quality code.

4. Enhanced Security

- Detection of Vulnerabilities: This may give a good coverage and identify security vulnerabilities that may have escaped the detection of either the tool or by a human.
- Reduced Attack Surface: The more your code is tested the lesser the attack surface, and hence your application becomes more resistant to various forms of security threats.

5. Improved Maintainability

- Clearer Codebase: This may give a good coverage and identify security vulnerabilities that may have escaped the detection of either the tool or by a human.
- Reduced Technical Debt: The more your code is tested the lesser the attack surface, and hence your application becomes more resistant to various forms of security threats.

Features of SonarQube

SonarQube is the backbone of all the modern software development teams because it has come prepackaged with a lot of cool features. Here are the key ones in short:

1. Multi-Language Support: The system supports many programming languages. This allows it to make the right analysis and recommendations about the codebase for each of them.

- Broad range of languages: SonarQube is actually supportive of a wide range of programming languages. These include Java, C#, C++, JavaScript, Python, PHP, among many more.
- Language-Specific Rules: Language-Specific Rules This provides language-specific rules and metrics, giving you the precise analysis with recommendations tailor-made for your specific needs.

2. Detection of Tricky Issues: Detection of Sneaky Problems: The tool can identify all sorts of sneaky problems, ranging from bugs and vulnerabilities to even code smells in the code.

- Code Smells: SonarQube can identify the so-called code smells. They must smell like some sign of bad code quality or bad maintainability.
- Security Flaws: It detects potential security vulnerabilities such as SQL injection, XSS, and buffer overflows.
- Performance Issues: Performance Issues It highlights performance issues, and if needed, can even serve to direct optimization.
- Code duplication: It identifies copied codes, that is a problem in maintenance and causes more errors.

3. Integration with CI/CD: SonarQube integrates fairly easily into the Continuous Integration/Continuous Delivery pipeline, allowing for code quality checks to be automated and useful feedback to be provided to developers at every step of the development process.

- Seamless Integration: This tool, as it truly integrates very well with continuous integration/continuous delivery pipelines.
- Automated Code Analysis: Automated Code Analysis can be set up so that changes in the code are analyzed automatically with immediate feedback provided to the developer before the code merges or gets released.

4. SonarLint IDE Integration: SonarLint IDE Integration Get immediate feedback on code quality issues directly in your development environment, so that you can detect and fix problems early, at the time of writing.

- Real-time Feedback: Real-time Feedback Integrate SonarLint with IntelliJ IDEA, Eclipse, or Visual Studio and get real-time feedback about your code's quality as you write.

- Quick Fixes: Offer quick fixes to problems identified that improve the quality of the code as you go along.

5. Plugin Ecosystem: SonarQube supports a very rich set of plugins that contribute significantly to considerable functionality, sometimes even custom to specific needs.

- Extensibility: It has been made pretty easy to extend its functionality through the plugin suite.
- Custom rules: You may add custom rules that can refer to specific coding standards or requirements.
- Third Party Integration: There are plugins, which, actually speaking, are a kind of integration with other tools and technologies.

6. SonarQube Rules: This encompasses broad ranges of predefined yet customizable rules implemented by SonarQube that analyzes the source code and comments on real issues based on bugs, security vulnerabilities, code smells, and performance issues.

- Predefined Rules: The predefined rules of SonarQube include a gigantic library covering all of the most current coding issues.
- Customizable Rules: It enables alteration of pre-set rules or creates a new one in order to make SonarQube mould suitably to your needs.
- Rule Categories: There are some general categories under which the rules fall, bugs, vulnerabilities, code smells, and hot spots.

SonarLint:

SonarLint (recently rebranded as SonarQube for IDE) is a free, open-source IDE extension that provides real-time feedback on code quality and security issues. It functions much like a spell checker, highlighting bugs and "bad code" as you type.

SonarLint analyzes the code in real time and identifies issues such as:

- Bugs – Errors that may cause the program to fail.
- Code Smells – Poor coding practices that make code hard to maintain.
- Security Vulnerabilities – Code patterns that may cause security risks.
- Unused Variables or Imports – Code that is unnecessary.
- Complex Code – Code that is difficult to understand or maintain.

Key Features

- **On-the-fly Detection:** Issues are identified immediately as you write code, before you even commit it.

- **Smart Education:** When an issue is raised, SonarLint provides detailed documentation explaining why it is a problem and how to fix it with code examples.
- **Connected Mode:** You can bind your IDE to a SonarQube or SonarCloud server. This ensures that the rules and quality profiles used in your IDE match those used by the rest of your team.
- **Security Focus:** It specifically looks for security vulnerabilities (SAST) to prevent potential exploits early in the development cycle.

Supported IDEs

SonarLint integrates with most major development environments:

- JetBrains: IntelliJ IDEA, PyCharm, WebStorm, CLion, etc.
- Microsoft: Visual Studio and Visual Studio Code.
- Eclipse: Standard Eclipse and Spring Tool Suite (STS).

Comparison: SonarLint vs. SonarQube

Feature	SonarLint (SonarQube for IDE)	SonarQube
Purpose	Real-time, local analysis for individual developers.	Centralized server for team-wide project analysis.
Feedback	Instant feedback as you type.	Post-commit/CI feedback via dashboard reports.
Installation	Installed as a plugin/extension in the IDE.	Installed on a server with a database.

Example Program with Code Issues (Detected by SonarLint)

Java Code

```
import java.util.Scanner;

public class StudentMarks {
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     int total = 0;
     int i = 0;
     int unused = 0;   // unused variable
     System.out.println("Enter number of students:");
```

```
    int n = sc.nextInt();
    int[] marks = new int[n];
    for(i = 0; i <= n; i++) {   // logical error (ArrayIndexOutOfBounds)
       System.out.println("Enter mark:");
       marks[i] = sc.nextInt();
       total = total + marks[i];
    }
    int average = total / n;
    if(average >= 90){
       System.out.println("Excellent");
    }
    else{
       if(average >= 75){
          System.out.println("Very Good");
       }
       else{
          if(average >= 50){
             System.out.println("Pass");
          }
          else{
             System.out.println("Fail");
          }
       }
    }
    System.out.println("Average marks: " + average);
  }
}
```

Issues Detected by SonarLint

When this code is analyzed by SonarLint in an IDE like Eclipse IDE or Visual Studio Code, several warnings appear.

1. Unused Variable

int unused = 0;

Issue: Variable declared but never used.

Suggestion: Remove unused variable.

2. Loop Condition Error

for(i = 0; i <= n; i++)

Issue: May cause ArrayIndexOutOfBoundsException.

Correct Version

```
for(i = 0; i < n; i++)
```

3. Resource Not Closed

```
Scanner sc = new Scanner(System.in);
```

Issue: Scanner resource not closed.

Correct Version

```
sc.close();
```

4. Nested If Statements (Code Smell)

SonarLint suggests simplifying nested if statements.

Better Version

```
if (average >= 90) {
  System.out.println("Excellent");
} else if (average >= 75) {
  System.out.println("Very Good");
} else if (average >= 50) {
  System.out.println("Pass");
} else {
  System.out.println("Fail");
}
```

Corrected Version of the Program

```
import java.util.Scanner;
public class StudentMarks {
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int total = 0;
      System.out.println("Enter number of students:");
      int n = sc.nextInt();
      int[] marks = new int[n];
      for(int i = 0; i < n; i++) {
          System.out.println("Enter mark:");
          marks[i] = sc.nextInt();
          total = total + marks[i];
      }
      int average = total / n;
      if (average >= 90) {
          System.out.println("Excellent");
```

```
        }
        else if (average >= 75) {
            System.out.println("Very Good");
        }
        else if (average >= 50) {
            System.out.println("Pass");
        }
        else {
            System.out.println("Fail");
        }
        System.out.println("Average marks: " + average);
        sc.close();
    }
}
```

Advantages of SonarLint

- Detects errors during development
- Improves code readability and maintainability
- Helps developers follow coding standards
- Reduces debugging time
- Enhances software security
- Easy to install and use

Limitations of SonarLint

- Works mainly at the individual developer level
- Cannot perform full project analysis like server-based tools
- Requires IDE integration
