Practical Assignment: Software Testing and Quality for MERN, Spring Boot, or .NET Applications

1. Test-Driven Development (TDD) & Behavior-Driven Development (BDD)

- Create a simple application using Spring Boot or .NET.
- TDD**:**
    1. Identify at least two core features (e.g., add task, validate user input).
    2. Write unit tests first using JUnit (Spring Boot) or NUnit (.NET) before implementing the features.
    3. Follow the Red-Green-Refactor cycle:
        - Write a failing test (Red).
        - Implement minimum code to pass the test (Green).
        - Refactor the code to improve quality while keeping tests green.
- BDD**:**
    1. Write feature files in Gherkin syntax describing at least one user story (e.g., "As a user, I want to add a new task so that I can track my work").
    2. Implement step definitions in Java with Cucumber or .NET BDD framework.
    3. Automate the scenarios and run the BDD tests.
    4. Demonstrate the test run and results.

2. Test Automation & Continuous Integration

- Write the following automated tests:
    - 2 Selenium UI test scripts:
        1. Identify two UI scenarios (e.g., login, add item).
        2. Implement tests using Selenium WebDriver in Java (Spring Boot) or C# (.NET), JavaScript or TypeScript for MERN.
        3. Run tests locally and confirm they pass.
    - 2 API test cases:

1. Use Postman or REST Assured (Java) to create automated tests for two API endpoints.
2. Validate response codes, payloads, and error handling.
3. Export Postman collection or REST Assured test code.

- 2 Automated unit tests (can overlap with TDD tests).

- Set up a CI/CD pipeline using either GitHub Actions or Jenkins:
  - Configure the pipeline to:
    1. Build the project.
    2. Run all unit and automation tests.
  - Demonstrate a successful pipeline run in the viva.

3. Performance, Security, and Usability Testing

- Load Testing with JMeter:
  1. Choose one critical API endpoint.
  2. Create a JMeter test plan simulating concurrent users.
  3. Run load test and capture key metrics (response times, throughput).
  4. Analyze results and identify bottlenecks.
- Security Testing (OWASP Top 10 basics)**:**
  1. Review your app for at least two OWASP Top 10 vulnerabilities.
  2. Demonstrate how to fix these issues.
  3. Provide evidence of fixes (code snippets or screenshots).

4. Defect Tracking and Bug Management

- Use Jira or Bugzilla to:
  1. Log at least two bugs found during your testing:
     - Include severity (Critical, Major, Minor).
     - Document detailed steps to reproduce the bugs.
  2. Perform root cause analysis on one bug:
     - Explain why it happened.

        - How it was fixed.
        - How to prevent similar bugs in future.
- Prepare to demonstrate your issue tracker entries during viva.

5. Software Quality Metrics and Standards

- Defect Density:
    1. Choose a module/component in your application, Count the lines of code (LOC) in this module, Calculate defect density
    2. Provide data for at least one module/component. From your defect tracking tool (Jira, Bugzilla), count the number of bugs found in this module during testing.
- Mean Time to Failure (MTTF):
    1. Simulate bug injection or theoretically estimate MTTF based on your testing cycles.
    2. Explain the concept and provide your calculation or reasoning.
- SonarQube Analysis:
    1. Run your project through SonarQube.
    2. Analyze and document:
        - Code smells found.
        - Duplicate code detected.
        - Vulnerabilities identified.
    3. Show remediation steps taken to improve code quality.