Assignment 1: API Testing, Coverage, and Benchmarking

Objective:
Students will build a simple REST API, implement tests for its endpoints, ensure code coverage, and conduct basic performance testing. This will involve using tools like HTTP files, Rest Clients, and JMeter (or similar). Additionally, students will perform basic benchmarking of their API's performance.

Instructions:

1. **Make a Test plan for the project**
    - make sure the Test plan keep up with the Test Strategy
2. **Create a Simple REST API:**
    - Continue to Develop/Refactor to a small REST API for a basic CRUD application like a **Task Manager**.
    - The API should include:
        - Create, Read, Update, Delete (CRUD) operations for a single entity (e.g., tasks).
    - The API can be built using **Java, C#, or any preferred language**. If using Other language than the tools mentioned are for, you must find equivalent tools to those mentioned in the instructions (e.g.,in the case of C# **NUnit** instead of **Junit** in Java).
3. **Unit and Integration Testing:**
    - Write unit tests for core service logic (e.g., adding/removing tasks, checking valid inputs).
    - Implement at least **three integration tests** that simulate real API interactions.
    - **Code Coverage:** Ensure a **minimum of 80% test coverage** using tools such as:
        - **Jacoco** for Java with Maven.
        - **Coverlet** for .NET with C# Or Visual Studio Build in functions.
    - Generate a coverage report and include it in the deliverables.
4. **API Testing Using HTTP Files and REST Client:**
    - Use **HTTP files** (or Postman) to test all API endpoints (e.g., GET, POST, PUT, DELETE).

- Ensure you test for:
    - Correct status codes (200, 404, 500, etc.).
    - Valid data returned in the response.
- Include the HTTP test scripts in the deliverables.
  Or Postman equivalent.

5. **Basic Load Testing with JMeter (or similar tool):**
    - Perform a basic load test on one of your endpoints (e.g., the "GET" endpoint).
    - Simulate **50-100 concurrent users** and record basic metrics (e.g., response time, throughput).
    - If using C#, students can use **Artillery** or **Gatling** as alternatives to JMeter. Jmeter can also be used for C#.
    - Analyze and document your results, focusing on the API's performance under moderate load.
6. **Reflection on Coverage and Performance:**
    - Reflect on how you ensured code coverage and maintained a balance between unit and integration tests.
    - Briefly discuss the **importance of code coverage** and how you could optimize performance based on your load testing and benchmarking results.

Deliverables:

- Source code for the REST API.
- Unit tests, integration tests, and HTTP test files.
- Code coverage report (minimum 80%).
- JMeter test plan and load test results (or equivalent).
- A brief reflection Maximum 1 A4 Page.
- Test Plan