Exercise 9 – API Testing

Goal:

In this exercise, you will:

- Design and implement a basic REST API endpoint in the FocusFlow backend project.
- Learn how to manually test API endpoints using standard tools.
- Automate the validation of API functionality using test frameworks.
- Apply performance and load testing techniques using industry tools (Apache JMeter or Gatling).
- Document their testing process and gain insights into test execution, quality metrics, and scalability challenges.

Preconditions / Requirements:

- A working backend setup for FocusFlow (in Java/Spring Boot, Python/Django, or TypeScript/Node.js).
- Basic understanding of REST principles and HTTP methods.
- Access to API testing tools (e.g., Postman, curl) and performance testing tools (JMeter or Gatling).
- GitHub repository for version control and deliverables.

Exercise 9.1 (10 Points): Develop an API Endpoint

Task: Implement a new REST API endpoint in your FocusFlow backend. The endpoint should provide at least two operations (e.g., create a task, retrieve tasks, update task status).

Deliverables:

- Source code available in your project repository.
- Clear documentation (e.g., in README or separate file) on how to start the backend and access the new API (including base URL, endpoints, methods, and expected request/response formats).

Exercise 9.2 (10 Points): Manual API Testing

Task: Use a manual API testing tool (e.g., Postman, curl, Talend, or HTTPie) to interact with your implemented endpoint. Perform at least:

- One successful request
- One request with invalid data (to observe error handling)

Deliverables:

- Documentation indicating which tool was used (screenshots or CLI logs).
- A brief summary of your testing experience and outcomes (e.g., response correctness, error behavior, ease of use).

Exercise 9.3 (10 Points): Automated API Test Cases

Task: Write automated test cases to validate your endpoint using an appropriate test framework for your stack:

- Java: Spring Boot MVC Tests + JUnit
- Python: Django REST Framework + pytest/unittest
- TypeScript/Node.js: Jest + SuperTest or similar

Focus on:

- Testing both success and failure cases
- Ensuring endpoint behavior matches expectations

Deliverables:

- Source code of test cases in your repository.
- Documentation explaining how to run the tests (e.g., CLI command, test runner setup).

Exercise 9.4 (10 Points): Load & Performance Testing

Task: Evaluate how your API behaves under load using **Apache JMeter** OR **Gatling**, following the setup instructions and the links provided in the slides

- Perform at least:
- One test with a constant load
- One test with a ramp-up or spike load

Measure:

- Response time
- Throughput
- Error rates (if any)

Deliverables:

- Test configuration (JMeter .jmx file or Gatling script).
- Documentation of the setup process, load profiles used, and key observations or performance bottlenecks identified.

(Optional) Exercise 9.5 (10 Points): CI Integration

You may optionally integrate your test cases (Task 9.3) into your GitHub Actions CI workflow (as prepared during Exercise 8), running them automatically on every push or pull request.