HW06 – API Testing

HW06 – API Testing

1. General Information

Exercise ID	**HW06-AI**
Duration	10 hours
Deadline	Please refer to the submission link on Moodle
Form	Individual Assignment
Submission	Moodle (report)
Lecturers & TAs	Dr. Lam Quang Vu / Dr. Tran Duy Hoang / MSc. Tran Thi Bich Hanh / MSc. Truong Phuoc Loc / MSc. Ho Tuan Thanh
Contact	lqvu@fit.hcmus.edu.vn / tdhoang@fit.hcmus.edu.vn / ttbhanh@fit.hcmus.edu.vn / tploc@fit.hcmus.edu.vn / hthanh@fit.hcmus.edu.vn
AI Policy	Open — a declaration and an attached AI Audit Report are **mandatory**
Required Bloom-AI Level	G9.1 → G9.6, depending on the homework (see the *CLO Mapping*)

2. Guiding Principles

These principles define how you are expected to work throughout the series of assignments in this course. Read them carefully before you begin, as your submission will be evaluated against them.

- **AI-First strategy.** You are required to apply AI to the testing techniques covered in class. However, this does not mean issuing a single, generic prompt such as *"generate all the API test cases from the spec and run them."* Instead, you must guide the AI through every step of the technique as it was taught, using the AI as a disciplined assistant rather than a black box.
- **Human review.** Every result produced by the AI must be carefully reviewed by you, the student. You are fully responsible for the correctness of these results. You are expected to make any necessary corrections and

refinements — submitting the raw AI output without review is not acceptable.

- **AI Audit Report.** The entire process of using AI must be recorded in a complete log. You are encouraged to build Agent Skills that can automatically perform these activities on similar exercises. If you do **not** use AI, you must still declare this explicitly.
- **Documentation.** The whole working process must be documented in a text-based format such as Markdown.
- **Quality over completion.** Your work will be graded not merely on whether it is complete, but on the quantity and quality of the deliverables: the test cases, the AI audit, the Postman collection and Newman report, bug reports, the test-generator design, and referenced links.

3. Learning Outcomes

By completing this assignment, you will be able to:

- Use AI to generate API test cases from the SUT's API specification, then audit and extend them.
- Design API tests covering domain partitions, state transitions, security, and schema validation.
- Detect bugs the AI missed, particularly around security and state transitions.
- Design an AI-driven test generator for the SUT.
- Demonstrate Bloom-AI competencies at levels **G9.2 (Apply)**, **G9.3 (Analyse)**, **G9.4 (Collaborate)**, and **G9.5 (Create)**.

4. System Under Test (SUT)

SUT: EShop — a Vietnamese e-commerce demo application designed for testing practice.

Repository: https://github.com/ttbhanh/eshop-sut

The application's features are organised into the following pools:

- **Pool A — Authentication, Categories, and Products**
  - FR-01: Account registration

    - FR-02: Login and account lockout
    - FR-03: Forgot password and password reset (two steps)
    - FR-04: Personal profile management
    - FR-05: Product listing and search
    - FR-06: Product detail view
- **Pool B — Shopping Cart and Checkout**
    - FR-07: Shopping cart
    - FR-08: Checkout
    - FR-09: Discount coupons
    - FR-10: Order state machine
    - FR-11: Order history view (user)
- **Pool C — Web Admin**
    - FR-12: Access control
    - FR-13: Dashboard
    - FR-14: Category management (CRUD)
    - FR-15: Product management (CRUD)
    - FR-16: Product import from CSV
    - FR-17: Coupon management (CRUD)
    - FR-18: Order management (admin)
    - FR-19: User management (admin)
- **Pool D — Mobile App**

The SUT ships an API specification in the repository (`api_specification.md`); consult it for the available endpoints. The specification also defines security requirements **SEC-01–SEC-07**.

5. API Selection

- Select **three (3) APIs**, one implementing a feature from **each of Pool A, Pool B, and Pool C** (Pool D, the mobile app, is not used here, because this

homework targets the backend API). Consult the API specification to find the endpoints behind each chosen feature.

  - **Pool A** — for example, login (FR-02) or product listing / search (FR-05).
  - **Pool B** — for example, the shopping cart (FR-07) or checkout / order creation (FR-08, FR-10).
  - **Pool C** — for example, an admin product or order operation with a state change (FR-15, FR-18).

- As in the previous assignments, ensure that your selection is **not duplicated** among the members of your group: no two members may choose the same three APIs.

6. Requirements

For each of your three selected APIs, complete the following pipeline. Document your process in the main report and attach the required evidence. Review the relevant course lectures on API testing before you begin.

1. **Generate with AI.** Provide the SUT's API specification to an AI tool and drive it — step by step, not with a single generic prompt — to generate test cases for the API (target **≥ 35 per API**). The cases must cover: **domain partitions** on every parameter (e.g., email format, password complexity, price > 0), **state transitions** (FR-10: pending → confirmed → shipping → delivered, plus cancelation rules), **security** (SEC-01–SEC-07, e.g., SQL injection, IDOR, role escalation), and **schema validation** (the response shape exactly matches the spec).
2. **Audit (human review).** Label each AI-generated test case **VALID / INVALID / INCOMPLETE** with reasoning, and correct the invalid or incomplete ones. You are fully responsible for the final test cases.
3. **Extend.** Add **at least five** test cases of your own that the AI missed — especially around security and state transitions — and explain *why* the AI missed them (prompt quality, model limitations, or characteristics of the API).
4. **Execute.** Run the test cases with Postman + Newman (or Karate / RestAssured). Every request must carry the header `X-Student-Id: {StudentID}` (for example, via a pre-request script). Produce the Newman / HTML report.

5. **Report bugs.** Report any genuine bugs you find — including bugs the AI missed — both in the Markdown report and on your GitHub Issues page, with a screenshot attached to each issue.

In addition, the following technical requirements apply across your whole test suite:

- **Exercise as many Postman features as you reasonably can** — for example: workspaces, collections, variables, environments, data-driven runs (the Collection Runner with a data file), monitors, and mock servers. **List the Postman features you used in your report.** *(Users of Karate / RestAssured provide the equivalent tool features.)*
- **Integrate into CI/CD.** Add your API test cases to a CI/CD pipeline for the SUT (for example, run Newman in GitHub Actions in your repository), and write a short **CI/CD report** describing the pipeline configuration and the two runs below, with screenshots and links. Provide **two sample commits**: one whose pipeline run shows **all** API test cases passing, and another whose pipeline run shows **one** test case failing.

7. Agent Skill

- For the Create level (G9.5), design an **AI-driven API test generator** for the SUT: given the API specification, it produces test cases automatically. Provide a **self-drawn diagram** and **pseudocode** of the design. ("Self-drawn" means you make the design decisions; any diagramming tool is fine, but the diagram itself must not be AI-generated.)
- You are encouraged to implement it as a reusable Agent Skill and submit a demonstration video (YouTube link) showing it generate tests for one API.

8. Allowed Tools and Bloom-AI Level

You may use the following tools, and you must declare them in your AI Audit Report:

- Any AI tool of your choice (e.g., ChatGPT, Claude, Gemini, Copilot, Cursor).
- Postman + Newman (default) or Karate / RestAssured (alternative).
- Optionally, LLM-testing tools (Promptfoo, DeepEval, Ragas).

The required Bloom-AI level for this homework is **G9.2 (Apply)**, **G9.3 (Analyse)**, **G9.4 (Collaborate)**, and **G9.5 (Create)**.

9. AI Audit Report (Mandatory Appendix)

Attach the AI Audit Report as an appendix. Use the content of the given AI Templates if needed.

- If you did not use AI, declare: *"I do not use any AI help in this exercise."*
- If you did use AI, declare: *"I use AI tools for the following tasks,"* and include the following information for each interaction:
  - Name of the AI tool
  - Date and time
  - Your prompt
  - The AI output

To simplify this process, you are encouraged to create a skill or rule that extracts the information above automatically after an AI session.

10. AI Critique (200–300 words, Mandatory)

Write a paragraph of 200–300 words critiquing the AI. Address the following questions: Where did the AI get something wrong, biased, or incomplete? Why did it fail to catch the issue? What principle have you learned about collaborating with AI during this assignment?

Use the content of the given AI Templates if needed.

11. Anti-AI-Cheat Constraints

This homework relies on real, attributable execution evidence. The following must not be AI-generated or fabricated, and the TAs verify them during grading:

- The `X-Student-Id: {StudentID}` header, evidenced by a console screenshot from your pre-request script.
- The Newman run output, whose hostname matches your deployment (`localhost` / `127.0.0.1` is accepted).
- The AI test-generator diagram, which must be self-drawn — designed by you, not generated directly by an AI.

12. Git Commit Log

- Create a new Git commit for each step of the procedure (for example: generation, audit, extension, and execution for each API).
- Provide the Git commit log in a text-based file format.

13. Oral Defense

A randomly selected **30% of students** may be invited to a 5–7-minute oral defense during the week following the deadline, to explain how they completed this homework.

14. Submission Regulations

- **Filename format:** `<StudentID>_HW06_AI_API_<SelfAssessedGrade>.zip`
  - *SelfAssessedGrade:* a 3-digit number in the range [000, 100].
  - *Example:* `25127001_HW06_AI_API_090.zip`
- **Required contents of the `.zip`:**
  - Main report (Markdown + PDF), including the API-testing report and your AI audit.
  - The public GitHub repository link (collections, scripts, and reports).
  - The Postman collection (`.json`) and the Newman report (HTML), plus the list of Postman features you used.
  - A short CI/CD report: the pipeline configuration and the two sample pipeline runs (one all-passing, one with a failing test case), with screenshots and links.
  - The Excel test cases and test summary.
  - The AI test-generator diagram and pseudocode (PNG / Mermaid + `.md` / `.py`).
  - Optionally, the API specification converted to OpenAPI (`.yaml` / `.json`); if AI-generated, audit it as well.
  - Bug report, with screenshots of the bugs on the GitHub Issues page.
  - AI Critique and AI Audit Report (Markdown + PDF).
  - Git commit log (text file).

    - A `README.md` containing the self-assessment table (below) and a test summary report: number of APIs; test cases generated, added, executed, passed, and failed; and number of bugs.
    - Any other supporting materials.
- Submit to Moodle. For the deadline, refer to the submission link.

15. Assessment Template

No.	Criteria	Grade	Self-Assessed Grade
1	API 1 — full pipeline (generate + audit + extend + execute + bugs)	30	
2	API 2 — full pipeline (same criteria)	30	
3	API 3 — full pipeline (same criteria)	30	
4	Agent Skills (AI-driven test generator)	10	
	Total	**100**	

16. References

- ISTQB Foundation Level Syllabus (latest edition).
- Hardman, P. (2025). *A Post-AI Learning Taxonomy.*
- Fuster Rabella, M. (2025). *OECD Education Working Paper No. 338.*
- Anthropic (2025). *Building Reliable AI Test Agents* — engineering blog.
- DeepEval & Promptfoo documentation — LLM testing frameworks.

17. Other Regulations

- Late submission is **not permitted**.
- Missing any required document results in **0 points**.
- Copying between students — **including prompts** — results in a **grade of 0 for both parties**.