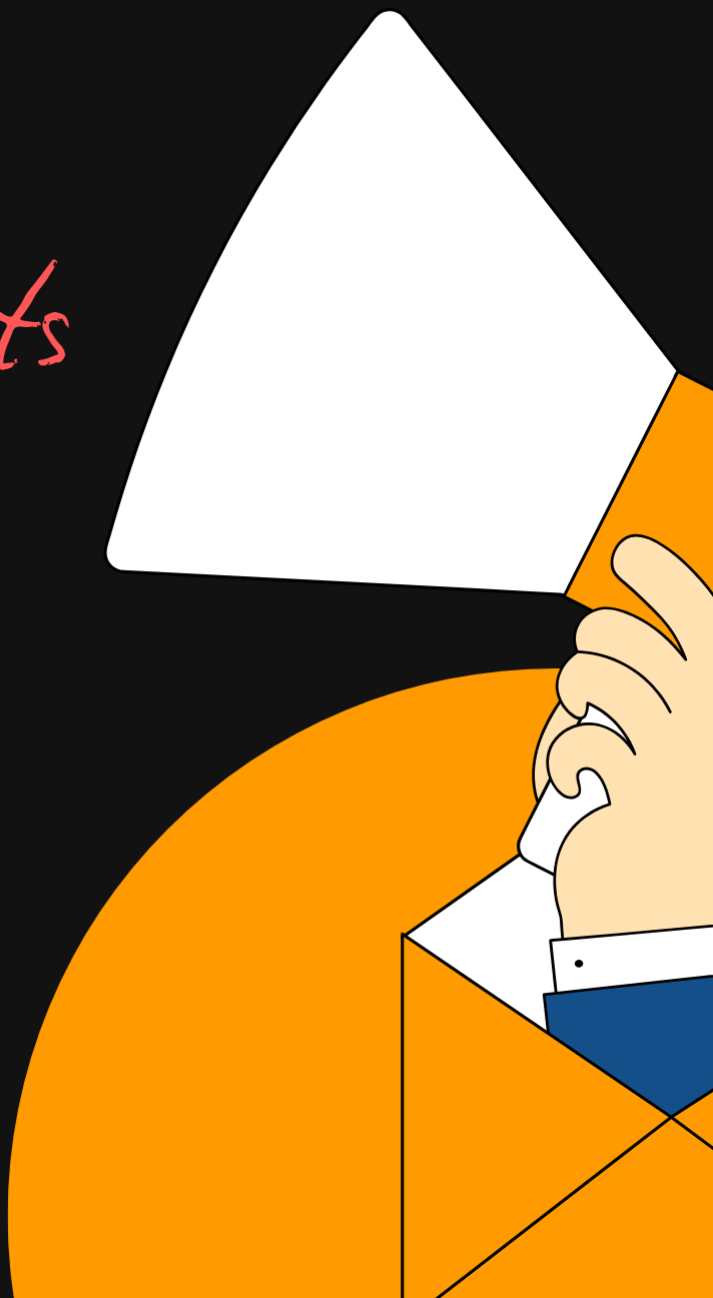
TESTING IN POSTMAN

Beginner-friendly guide to writing Postman tests for API endpoints

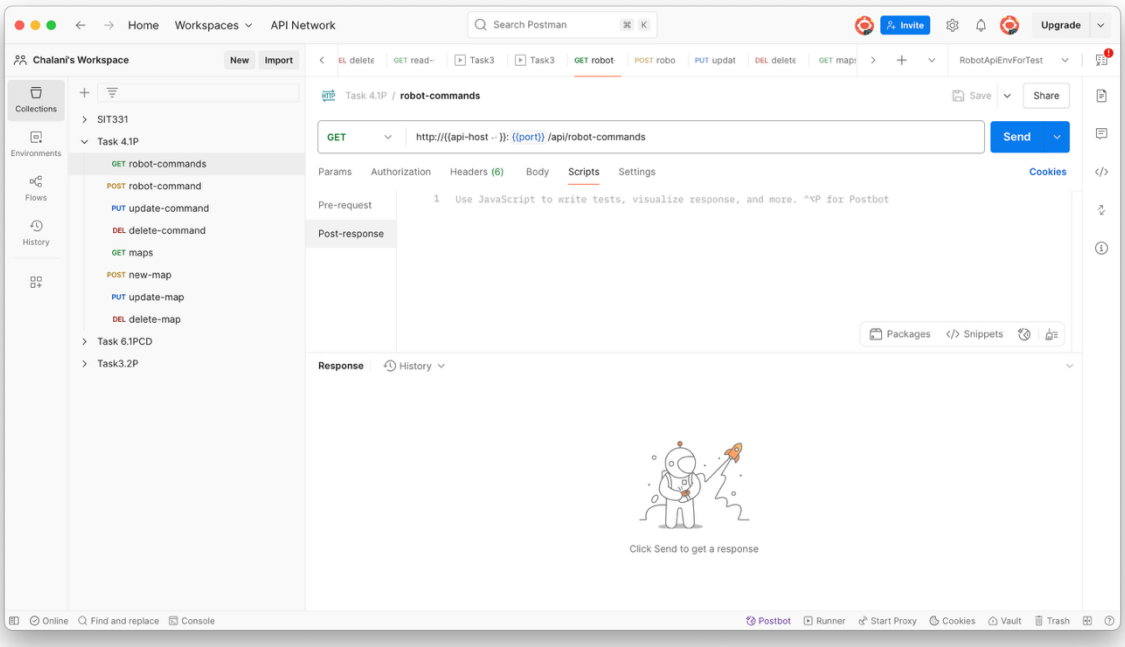


POSTMAN

<Chalani Lamahewa>
<SIT331>
<223817302>

Postman tests? What is it ?

Small JavaScript scripts that we can build in Postman's "Tests" tab (under "Scripts") are called tests, and they automatically check to see if your API operates as intended. (If you have a newer version of the Postman then there will not be a separate tester and the Post-response section under Scripts tab acts as the new tests section.



These tests that we create are run after the request is sent and after the response id received.

In Postman there is two scripting sections under the script tab, which are Pre-request script and post-response script. The Pre-request script runs before the request is sent and the post-response script which are the tests that run after the response is received.

Basic Concepts

pm	pm stands for postman and gives us access to all built-in functions within Postman.
pm.test(name, function)	This enables us to create a test within a function with a clear name and logic.
pm.expect ( )	This enables the writing of assertions, (what you believe to be true) Ex: expect(value).to.eql(expected)
pm.response	This represents the returned HTTP response
pm.response.code	This represents the code for the response Ex: 200, 404
pm.response.status	Text status of the response Ex: Ok, Not found
pm.response.json ( )	Parses the response body as JSON
pm.response.text ( )	Gets the raw reponse in plain text
pm.environment.get("variable")	Reads an environment variable by name
pm.environment.set("variable", value)	Sets or updates an environment variable
pm.collectionVariables.get("variable")	Reads a variable from the collection
pm.globals.get("variable")	gets a global variable accessible from any collection
pm.request	Refers to the request object that was sent to the server.
pm.request.body	Accesses the request body content.
pm.variables.replaceIn(string)	Replaces variable placeholders ({{userId}} in a string.

Common built in libraries

console.log()	Prints data to Postman's console
Math, Date, etc	basic JavaScript built-in objects
lodash (_)	A utility library for working with arrays, objects and more
moment	Date/time manipulation

Most common mistakes

Using a variable before setting it – make sure to check if the variable you want use exist within the environment

Not using .json before accessing data – use pm.response.json() to convert response into a usable object

Forgetting .to.be.a() or .eql() – make sure to use assertions correctly as otherwise test does nothing and passes.

Test Script

Every test in Postman follows a simple structure:

```
pm.test(" test name", function () {
    // test logic goes here
});
```

pm – Postman

.test ( ) – this is a method to define a test

Test name – this is what will be shown in the test results tab

function ( ) { } – assertion

Basic Assertions using pm.expect ( )

Postman uses ChaiJS which is a JavaScript assertion library.

Check status code - pm.expect(pm.response.code).to.eql(200);

Check status text - pm.expect(pm.response.status).to.eql("OK");

Check header exist - pm.response.to.have.header("Content-Type");

These are a few of the basic assertion that is commonly used.

Example for advanced assertion

Looping through an array response

```
let data = pm.response.json();
pm.test("All items have 'name'", () => {
   data.forEach(item => {
      pm.expect(item).to.have.property("name");
   });
});
```

Key testing theories and principles that apply to Postman testing

Test-Driven Development (TDD)

Definition: Write tests before writing the actual code.

In TDD, you might first define how the API should behave using Postman tests.

Once the backend is written, you run these tests to validate functionality.

Assertion-Based Testing

Definition: Use assertions to compare actual vs expected outcomes.

In Postman, pm.expect() is how you assert.

Assertions help you define the truth of your system.

Black Box Testing

Definition: Test the system without knowing the internal code.

In Postman, you're testing an API from the outside — sending inputs and checking outputs.

You don’t care how the backend is written — just that it works as expected.