Flash Assignment Quality Assurance

Task 3 - API Testing for Catfact Ninja

1. What Would You Test About This API?

When testing the Catfact Ninja API, I would focus on the following key aspects:

- **Functionality** – Ensuring that the API returns the correct responses for valid requests.
- **Data Integrity** – Verifying that the cat facts are returned as expected and are not malformed or missing data.
- **Performance** – Checking response times and ensuring the API meets expected performance benchmarks.
- **Error Handling** – Testing how the API responds to invalid requests or edge cases.
- **Security** – Validating authentication (if applicable) and ensuring there are no vulnerabilities such as injection attacks or excessive data exposure.
- **Rate Limiting** – Checking if the API enforces rate limits if applicable.
- **Contract Testing** – Ensuring that the API response adheres to its schema as defined in the documentation.

---

2. Why Would You Test This?

Testing the API ensures that:

- It functions as expected, delivering accurate and reliable cat facts.
- It handles various inputs correctly, including edge cases and invalid requests.
- The performance remains optimal under different conditions.
- It is secure and does not expose vulnerabilities to potential exploits.
- It complies with the expected contract to maintain consistency and prevent breaking changes.

---

3. How Would You Test This?

I would use **Postman** and **Newman (for automation)** to test the API by following these steps:

1. **Understand the API Documentation** – Review available endpoints, request parameters, and expected responses.
2. **Set Up Postman Collections** – Create a collection to store different test cases for the API.
3. **Write Test Cases** – Cover functional, negative, performance, and security testing scenarios.
4. **Run and Validate Responses** – Execute the tests and validate responses against expected results.
5. **Automate API Tests** – Use Postman scripts and Newman CLI to automate test execution.
6. **Monitor & Report Issues** – Analyze test results, log defects if necessary, and generate test reports.

---

4. Three Example Tests Using Postman

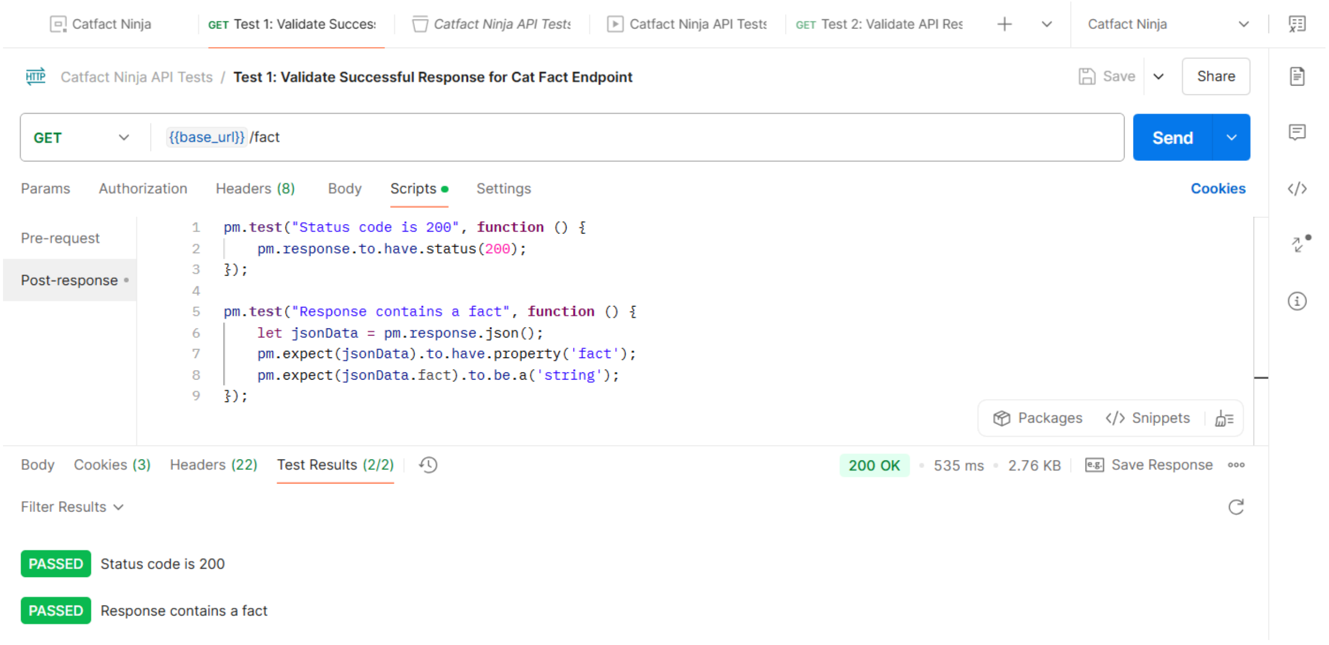
Test 1: Validate Successful Response for Cat Fact Endpoint

- **Endpoint:** `GET /fact`
- **Expected Behavior:** Should return a random cat fact with a 200 status code.
- **Test Steps:**
    1. Send a GET request to `/fact`.
    2. Validate that the response status is `200 OK`.
    3. Verify that the response contains a valid JSON object with a `fact` key.

Postman Test Script:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains a fact", function () {
    let jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property('fact');
    pm.expect(jsonData.fact).to.be.a('string');
});
```

- 

Catfact Ninja API Tests / Test 1: Validate Successful Response for Cat Fact Endpoint

GET {{base_url}} /fact

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains a fact", function () {
    let jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property('fact');
    pm.expect(jsonData.fact).to.be.a('string');
});
```

Test Results (2/2) 200 OK 535 ms 2.76 KB

PASSED Status code is 200

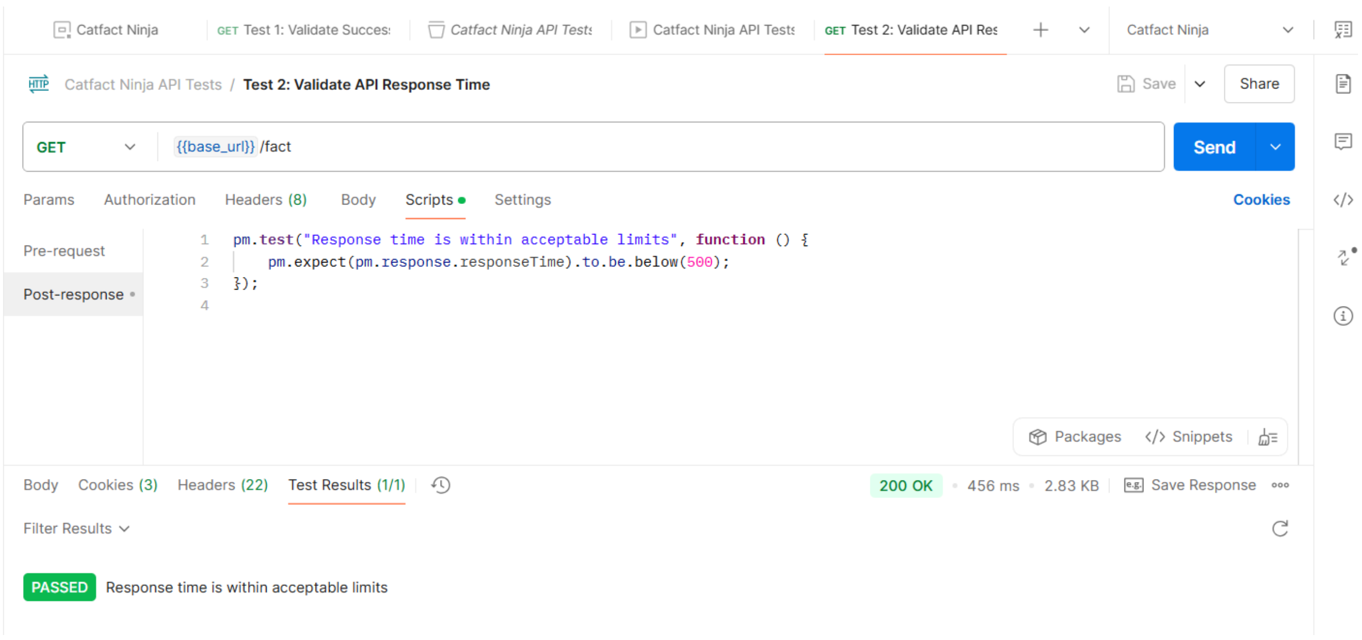
PASSED Response contains a fact

Test 2: Validate API Response Time

- **Expected Behavior:** API should respond within 500ms.
- **Test Steps:**
    1. Send a GET request to `/fact`.
    2. Validate that the response time is under 500ms.

Postman Test Script:

```
pm.test("Response time is within acceptable limits", function () {
    pm.expect(pm.response.responseTime).to.be.below(500);
});
```



```
pm.test("Response time is within acceptable limits", function () {
    pm.expect(pm.response.responseTime).to.be.below(500);
});
```

PASSED Response time is within acceptable limits

•

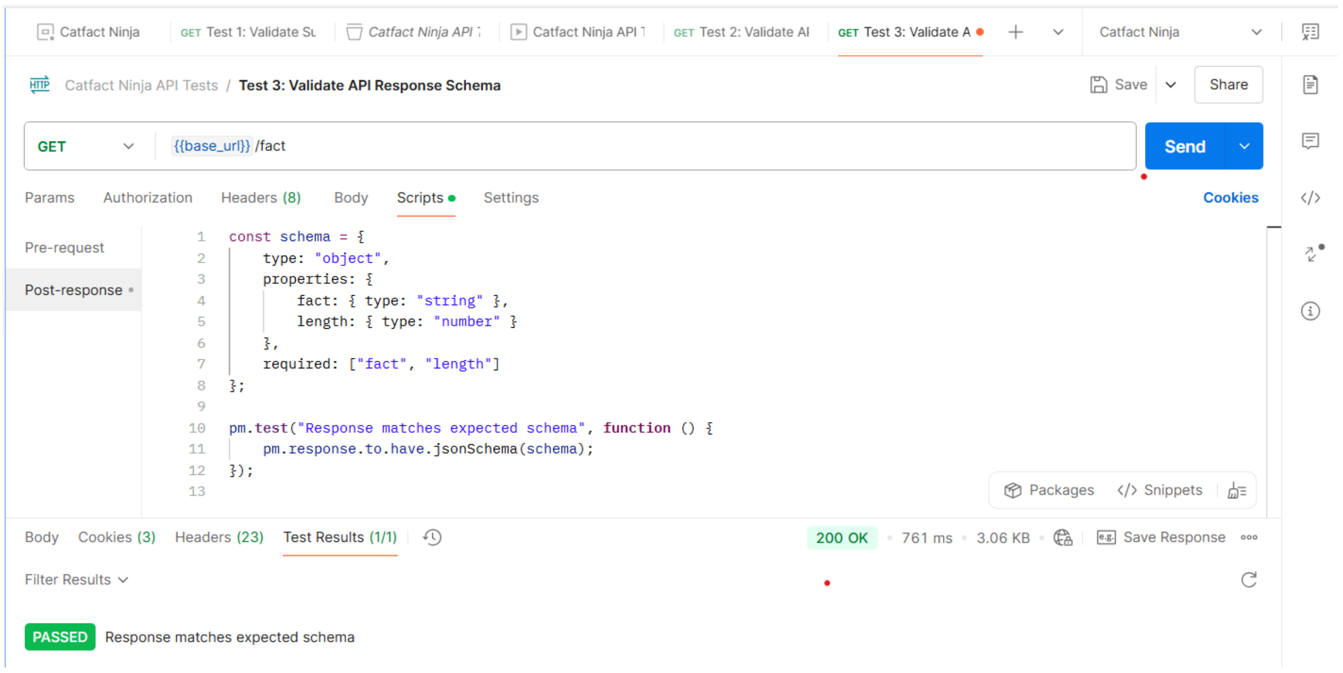
Test 3: Validate API Response Schema

- **Expected Behavior:** API response should match the defined JSON schema.
- **Test Steps:**
  1. Send a GET request to `/fact`.
  2. Validate that the response matches the expected JSON schema.

Postman Test Script:

```
const schema = {
    type: "object",
    properties: {
        fact: { type: "string" },
        length: { type: "number" }
    },
    required: ["fact", "length"]
};

pm.test("Response matches expected schema", function () {
    pm.response.to.have.jsonSchema(schema);
});
```

- 
Catfact Ninja API Tests / Test 3: Validate API Response Schema

GET {{base_url}} /fact

```
const schema = {
    type: "object",
    properties: {
        fact: { type: "string" },
        length: { type: "number" }
    },
    required: ["fact", "length"]
};

pm.test("Response matches expected schema", function () {
    pm.response.to.have.jsonSchema(schema);
});
```

200 OK • 761 ms • 3.06 KB

PASSED Response matches expected schema

---

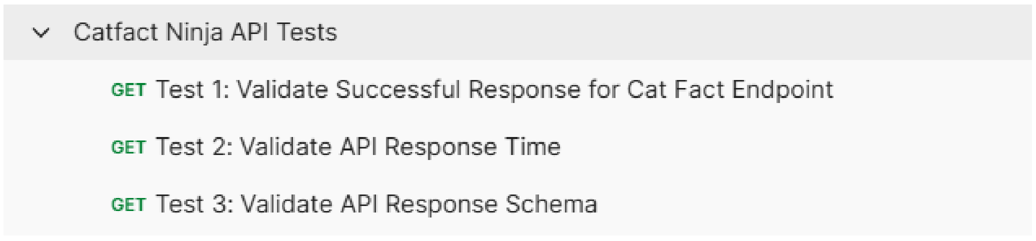
5. Process of Setting Up These Three Tests

1. **Create a New Postman Collection**

   ○ Created a collection named **"Catfact Ninja API Tests"** in Postman.

2. **Add Requests to the Collection**

   ○ Created a new request for `/fact` and saved it in the collection.

   ○ 

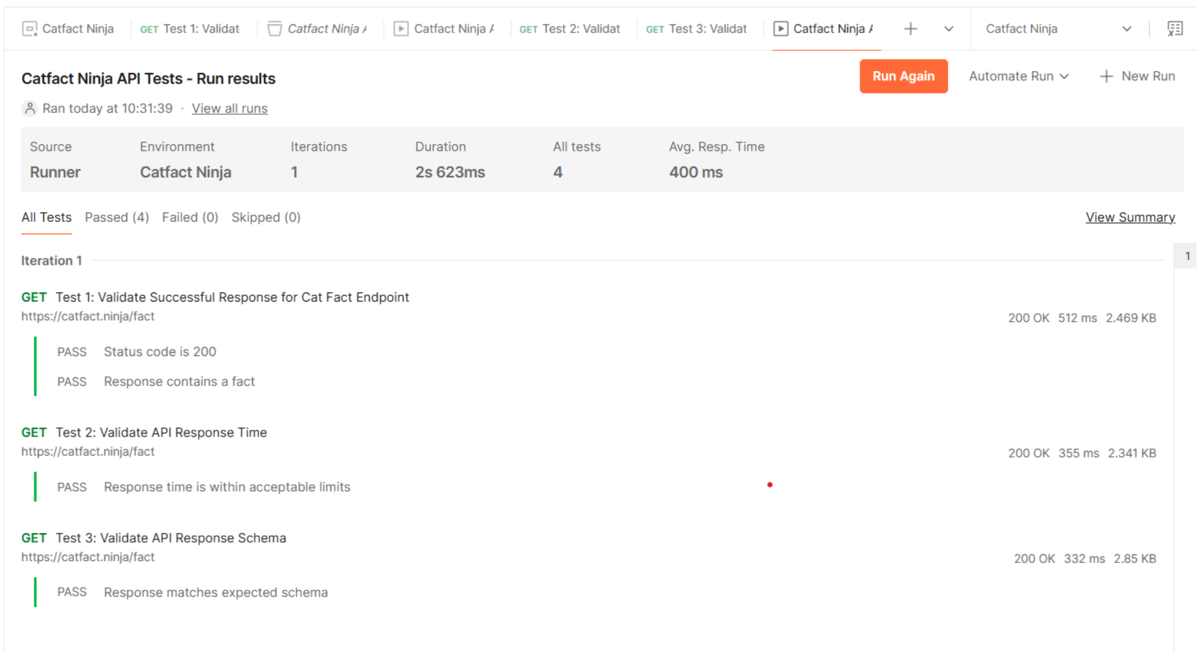
   Catfact Ninja API Tests
   - GET Test 1: Validate Successful Response for Cat Fact Endpoint
   - GET Test 2: Validate API Response Time
   - GET Test 3: Validate API Response Schema

3. **Write Test Scripts in the Tests Tab**

   ○ Implemented validation scripts in the **Tests** section of each request.

4. **Run Tests Using Collection Runner**

   ○ Executed all tests using the Postman **Collection Runner** to verify correctness.

- 

Catfact Ninja API Tests - Run results

Ran today at 10:31:39 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Catfact Ninja	1	2s 623ms	4	400 ms

All Tests Passed (4) Failed (0) Skipped (0)

Iteration 1

GET Test 1: Validate Successful Response for Cat Fact Endpoint
https://catfact.ninja/fact — 200 OK 512 ms 2.469 KB
PASS Status code is 200
PASS Response contains a fact

GET Test 2: Validate API Response Time
https://catfact.ninja/fact — 200 OK 355 ms 2.341 KB
PASS Response time is within acceptable limits

GET Test 3: Validate API Response Schema
https://catfact.ninja/fact — 200 OK 332 ms 2.85 KB
PASS Response matches expected schema

5. **Export Collection for Automation**

   - Exported the Postman collection as a JSON file for easy sharing and execution via Newman.
   -

---

6. Postman Collection Export & Necessary Files

I'll attach the exported Postman collection and any additional files needed.