Report: API Test Automation Challenge

Objective

The objective of the challenge was twofold:

1. **Task 1:** Automate the testing of the Swagger Petstore RESTful API using Java, RestAssured, and JUnit.
2. **Task 2:** Conduct performance testing on the Swagger Petstore API using K6 to evaluate its performance under different load conditions.

Environment Setup

Project Directory Structure

- Created a main directory named `yellowPaper`
- Inside `yellowPaper`, created two subdirectories:
    - `first-task` for API test automation
    - `second-task` for API performance testing

Clone the Swagger Petstore Repository

- Cloned the Swagger Petstore repository using:
  `git clone https://github.com/swagger-api/swagger-petstore.git`
- Navigated to the cloned directory:
  `cd swagger-petstore`

Running the Swagger Petstore API Locally

- Attempted to run the server using Maven:
  `mvn package jetty:run`
- Encountered an error indicating that the `openapi-inflector.yaml` file was missing. The present file was named `openapi.yaml`.
- Renamed `openapi.yaml` to `openapi-inflector.yaml` and re-ran the command:
  `mvn package jetty:run`
- Successfully started the server on http://localhost:8080 and verified it in a web browser for task 1.
- Successfully started the server on http://localhost:8081 and verified it in a web browser for tak 2.

Dependency Management

Added the following dependencies to `pom.xml` for API testing with RestAssured and JUnit:

piar código

```
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <version>5.3.1</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <version>5.8.2</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <version>5.8.2</version>
  <scope>test</scope>
</dependency>
```

Test Cases Automation

Created the test cases in `PetStoreTest.java` located at `src/test/java/swagger/petstore/` with the following tests:

- **Creating a Pet:** Validates the creation of a new pet in the store.
- **Retrieving a Pet:** Confirms retrieval of the created pet.
- **Updating a Pet:** Verifies updating an existing pet's information.
- **Deleting a Pet:** Ensures deletion of a pet and verifies it is removed.
- **Retrieving a Non-Existent Pet:** Checks that a non-existent pet returns a 404 status.

- **Updating a Non-Existent Pet:** Validates that attempting to update a non-existent pet returns a 404 status.
- **Deleting a Non-Existent Pet:** Confirms that deletion of a pet with an invalid ID returns a 400 status.

Execution of Automated Tests

- Ran the automated tests using:
  `mvn test`
- **Test Results:**
  - All tests were executed successfully, with no failures or errors.
  - Summary of results:
    ```
    [INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
    [INFO] BUILD SUCCESS
    ```

Performance Testing Report

Objective

The goal of the performance testing task was to evaluate the Swagger Petstore API's performance under different load conditions using K6. The tests focused on three endpoints: Create Pet, Get Pet By ID, and Update Pet.

Setup

- **Repository Cloned:**
  - Repository URL: https://github.com/swagger-api/swagger-petstore.git
  - Repository Directory: `swagger-petstore`
- **Configuration Changes:**
  - Renamed `openapi.yaml` to `openapi-inflector.yaml`
  - Changed the server port from 8080 to 8081
- **Test Scripts Created:**
  - `createPetTest.js`
  - `getPetByIdTest.js`
  - `updatePetTest.js`

Test Scenarios

- **Create Pet API Test (`createPetTest.js`):**
  - **Scenario:** Simulate up to 200 concurrent users creating new pets.
  - **Stages:**
    - Ramp-up to 100 VUs over 1 minute
    - Maintain 200 VUs for 2 minutes
    - Ramp-down to 0 VUs over 30 seconds
- **Get Pet By ID API Test (`getPetByIdTest.js`):**
  - **Scenario:** Simulate up to 200 concurrent users fetching pets by ID.
  - **Stages:**

        - Ramp-up to 100 VUs over 1 minute
        - Maintain 200 VUs for 2 minutes
        - Ramp-down to 0 VUs over 30 seconds
- **Update Pet API Test (`updatePetTest.js`):**
    - **Scenario:** Simulate up to 200 concurrent users updating pets.
    - **Stages:**
        - Ramp-up to 100 VUs over 1 minute
        - Maintain 200 VUs for 2 minutes
        - Ramp-down to 0 VUs over 30 seconds

Results

- **Get Pet By ID API Test:**
    - **Duration:** 3 minutes 30 seconds
    - **Total Requests:** 23,672
    - **Successful Requests:** 23,672 (100%)
    - **Failed Requests:** 0 (0%)
    - **Average Response Time:** 9.2 ms
    - **Maximum Response Time:** 143.82 ms
    - **Average Throughput:** 112.38 requests/second
    - **Average Iteration Duration:** 1.01 seconds
    - **Notes:** Warnings about high cardinality metrics were observed.
- **Update Pet API Test:**
    - **Duration:** 3 minutes 30 seconds
    - **Total Requests:** 23,656
    - **Successful Requests:** 23,656 (100%)
    - **Failed Requests:** 0 (0%)
    - **Average Response Time:** 9.3 ms
    - **Maximum Response Time:** 76.97 ms
    - **Average Throughput:** 112.22 requests/second
    - **Average Iteration Duration:** 1.01 seconds
    - **Notes:** Warnings about high cardinality metrics were observed.
- **Create Pet API Test:**
    - **Duration:** 3 minutes 30 seconds
    - **Total Requests:** 23,744
    - **Successful Requests:** 23,534 (99.1%)
    - **Failed Requests:** 210 (0.88%)
    - **Average Response Time:** 5.5 ms
    - **Maximum Response Time:** 253.81 ms
    - **Average Throughput:** 112.59 requests/second
    - **Average Iteration Duration:** 1.01 seconds
    - **Notes:** Some failures in requests, but the majority were successful.

Conclusion

- **Task 1 Conclusion:** The automation framework was successfully set up and implemented using Java, RestAssured, and JUnit. All test cases executed successfully, demonstrating the stability and reliability of the Swagger Petstore API in

the local environment. The solution provides a robust foundation for API test automation following best coding and design practices.

- **Task 2 Conclusion:** The performance tests show that the Swagger Petstore API performs well under load conditions with up to 200 concurrent users. Average response times for all tests were within acceptable limits, with most requests succeeding. However, occasional failures in the Create Pet test suggest areas for further investigation or optimization. The warnings about high cardinality metrics indicate a need for improvements in metric tagging or grouping to manage memory usage effectively.