**Test Plan for FakeREST API**

**1. Introduction**

This document outlines the test plan for testing the FakeREST API. The objective is to verify the correctness, reliability, and performance of the API using automated testing with RestAssured and TestNG.

**2. Scope**

Testing will cover the core functionalities of the FakeREST API, including:

* Operations for books, authors, and activities
* Data validation and integrity
* Functional, integration, performance and basic error handling aspects of the API.

**3. Testing Strategy**

**3.1 Types of Testing**

**Functional Testing**:

Focus: Verify that the API endpoints return the expected data and perform the intended operations.

Techniques:

* Positive testing: Verify successful responses with valid inputs.
* Negative testing: Verify appropriate error responses with invalid inputs.
* Tools: RestAssured, TestNG.,Mvn

**Integration Testing**:

Focus: Verify interactions between different API endpoints and data consistency.

Techniques:

* Test data flow between related endpoints (e.g., create a resource and then retrieve it). Tools: RestAssured, TestNG,Mvn

**End-to-End Testing**:

Test complete workflows, including data creation and retrieval.

**Performance Testing**:

Focus: Evaluate the API's response time.

Techniques:

* Response time measurement: Track the time taken for API responses.

.

 Tools: RestAssured.

**Error Handling Testing:**

Focus: Verify that the API returns correct error codes and messages when invalid data is provided

* Tools: RestAssured, TestNG.,Mvn

**3.2 Test Execution Approach**

* **Automated Testing**: Use RestAssured with TestNG for API validation.
* **Test Execution Order**:
  1. Validate GET endpoints
  2. Test POST requests for data creation
  3. Verify PUT and DELETE operations
  4. Perform authentication and security checks
  5. Execute performance testing

**4. Test Environment**

* **Tools**: RestAssured, TestNG, Maven
* Operating Systems: Windows, macOS
* Devices: Desktop, laptop
* Java version: "23.0.2.
* RestAssured, TestNG, Log4j2, and Jackson libraries.
* Postman.

**5. Test Data**

Mock data for books, authors, and activities

BaseUrl: https://fakerestapi.azurewebsites.net/api/v1

**6. Entry/Exit Criteria**

* Entry: API documentation available, test environment set up.
* Exit: All high-priority test cases passed, acceptable performance metrics.

**7.Postman Collection**

* A Postman collection was created with folders for Books, Authors
* Each folder contains requests for GET, POST, PUT, and DELETE operations.
* Environment variables were used for the base URL
* Basic test assertions where added to the postman collection.
* The postman collection can be exported as a JSON file.

**7. Test Deliverables**

1. **Test Cases Document** – Prioritized test cases covering functional and performance testing.
2. **Test Execution Report** – Summary of executed tests and results.
3. **Postman Collection** (optional) – For manual verification of API requests.
4. **Automation Scripts** – Implemented using RestAssured and TestNG.

**Test Execution Run**

* All tests were executed using TestNG and Maven.
* The testng.xml file was configured to run tests within the "regression test" group.
* Test results were logged to the console and target/test.log using Log4j2.
* Test Reports are available in the target/surefire-reports folder.

**8. Testing Evidence:**

* The target/surefire-reports folder contains the TestNG HTML reports.
* The target/test.log file contains the log output.
* The Postman collection JSON file can be provided.
* The java test code, the pom.xml file, and the log4j2.xml files are all available

**9. Summary Report**

**9.1 Test Execution Summary**

* **Total Test Cases**: 22
* **Passed**: 22
* **Failed**: 0
* **Blocked**: 0

**9.2 Key Findings**

* CRUD operations work as expected with valid data.
* Unauthorized requests return correct status codes.

**9.3 Recommendations**

* Optimize API response times for large datasets.
* Improve error handling for invalid requests.

.

### 10. Conclusion

This API test plan defines a structured approach to validate the FakeREST API. Using automated tests with RestAssured and TestNG ensures consistent verification of API functionality and performance.