Test Plan for ReqRes API Testing

1. Introduction

- **Project Name**: ReqRes API Testing
- **Testing Tool**: Postman
- **Testing Type**: API Testing (Automation)
- **Objective**:

The objective of this test plan is to thoroughly validate the functionality, performance, and security of the ReqRes API endpoints. This involves ensuring that each API endpoint behaves as expected under various conditions, returning the correct status codes, response body, and headers, while properly handling error cases. The goal is to ensure that the API is reliable, secure, and performs efficiently under different usage scenarios.

2. Scope of Testing

Verification of Endpoint Responses

- Ensure that each API endpoint (GET, POST, PUT, DELETE) returns the correct HTTP status codes for both valid and invalid requests.
- Validate that the response body for each endpoint matches the expected format, including data consistency and structure (i.e., JSON format, required fields).

Functional Validation

- Confirm that each API endpoint returns the correct data when provided with valid inputs (e.g., retrieving user data, creating or updating users).
- Test for appropriate behavior when invalid inputs are provided (e.g., non-existent user ID or invalid data format for POST/PUT requests).
- Validate the CRUD operations (Create, Read, Update, Delete) to ensure they function correctly on both valid and invalid data.

Error Handling

- Test that error cases, such as invalid user IDs, missing parameters, or incorrect request formats, result in appropriate HTTP error codes (404, 400, etc.) and informative error messages.

- Ensure that error messages provide clear, actionable feedback for the end user or API consumer.

Authorization and Security

- (If applicable) Verify that endpoints requiring authentication and authorization (e.g., Bearer token or basic auth) properly enforce security protocols.

- Test that unauthorized access to protected endpoints is blocked and returns appropriate error responses.

Performance Testing

- Measure the API response times for different endpoints to ensure they meet performance standards (e.g., response times under a certain threshold).

- Identify any performance bottlenecks or unexpected delays in the API responses.

API Consistency and Reliability

- Ensure that the API remains consistent across various requests and sessions (e.g., the data returned from GET requests remains accurate over time).

- Test for idempotency in certain operations (like PUT/DELETE), ensuring they can be repeated safely without negative side effects.

Compliance with API Specifications

- Ensure that the API responses conform to the API specifications, including correct HTTP methods, response codes, and headers, as outlined in the ReqRes API documentation.

3. Test Strategy

The strategy for automating the ReqRes API testing will focus on ensuring that the API behaves as expected across all functional and non-functional areas. Automation will be employed for consistency, repeatability, and efficient regression testing. This approach ensures comprehensive test coverage for the API, handling both valid and invalid requests, performance testing, and error handling. The strategy includes the following components:

Test Levels

- **Unit Testing**: Not part of this plan but should be performed by the development team to ensure each individual component of the API functions as intended.

- **Integration Testing**: Focus on testing the integration of API endpoints to ensure that different endpoints interact correctly (e.g., ensuring a POST request creates a user and a GET request retrieves that user).
- **System Testing**: Testing the API endpoints as part of the whole system to ensure they integrate correctly with other systems (e.g., databases, external services).
- **Regression Testing**: Re-running automated tests to ensure that new updates or features do not break existing functionality.

Test Types

- **Functional Testing**: Automation will verify that all API endpoints are functioning as expected, validating responses, status codes, and data formatting. Automated tests will be written for the following methods:
- **GET**: Fetch data (e.g., retrieving a list of users).
- **POST**: Create resources (e.g., creating a new user).
- **PUT/PATCH**: Update resources (e.g., updating a user's information).
- **DELETE**: Delete resources (e.g., deleting a user).
- **Negative Testing**: Automatically test invalid requests, such as missing parameters, invalid data formats, etc., ensuring proper error handling.
- **Security Testing**: Automation will include tests to check for basic security practices, such as ensuring unauthorized users cannot access sensitive endpoints.
- **Performance Testing**: Automated performance testing will evaluate response times and throughput using tools like **Newman** (Postman's command-line tool). Stress testing will be part of the automation to evaluate the API's behavior under heavy loads.
- **Regression Testing**: Each time the API is updated, the existing automated tests will be re-executed to ensure that new changes do not break existing functionality.
- **Error Handling Testing**: Automated checks will ensure that the API returns appropriate error codes (e.g., 400, 401, 404) and messages for invalid inputs.

4. Test Approach

- **Postman for Automated Testing**: Postman will be the primary tool for automation. Test cases will be written as Postman collections that can be executed against the API, with assertions in the test scripts to verify the responses.

- **Newman**: Newman will be used to run the Postman collections from the command line. This will allow for integration with CI/CD pipelines for automated test execution during development cycles.

- **CI/CD Integration**: Automated tests will be integrated into the CI/CD pipeline to run tests automatically with every code change or deployment. This helps identify issues early in the development process and provides immediate feedback on the API's functionality.

- **Scheduled Testing**: Periodic automated tests will be executed to check for regression or performance issues that may arise over time, even if no new code changes have been made.

5. Tools and Frameworks

- **Postman**: The main tool for writing and executing the automated test cases.

- **Newman**: For running Postman collections from the command line, making it easier to integrate tests into the CI/CD pipeline.

- **CI/CD Tools**: Jenkins, GitHub Actions, or any preferred CI/CD tool for automating test execution as part of the development lifecycle.

- **Jira**: For bug tracking, where any issues found during the tests will be reported and tracked until resolution.

6. Test Environment

- **API Endpoint**: Testing will be done on the public ReqRes API available at https://reqres.in/.

- **Platform**: Testing will be performed on both Windows and macOS environments.

- **Connectivity**: A stable internet connection will be required for communication with the API.

7. Test Deliverables

- **Postman collections** with defined test cases.

- **Test execution results**, including logs and reports.

- **Defect reports** for any failed test cases.
- A comprehensive **test summary report** detailing the overall coverage, pass/fail rates, and identified issues.

8. Test Exit Criteria

- All defined automated tests have passed.
- There are no critical defects that would block the functionality of the API.
- Performance tests meet predefined benchmarks for response times and scalability.
- Test execution has reached the required level of coverage and stability.

9. Tools and Resources

- **Tools Required**:
- **Postman**: For API requests and testing.
- **Newman (optional)**: For running automated tests in CI/CD pipelines.
- **Postman Collections**: For organizing and executing multiple test cases.

10. Conclusion

The test plan for the ReqRes API ensures that all endpoints are thoroughly tested for functionality, performance, and error handling. Any defects or issues found will be reported and resolved promptly. Postman provides a robust platform for managing and executing API tests, streamlining the automation and testing process.