REQRES.IN API TESTING

TEST PLAN

Table of Contents

INTRODUCTION

PURPOSE OF THE DOCUMENT

In this API test, different types of operations are tested for the reqres.in and make sure each request returns appropriate response. The test cases are like get users, create users, update users filtered by firstname, lastname, Partially Update users, Delete users and so on. There have many test cases based on the operations. After conducting these test cases with different body parameter, we can see if all the results are as expected or not. By this test project any stakeholders can have a clear idea about what could be the possible test cases to cover all the features/operations for the booking solution.

TEST ITEM

PROJECT DESCRIPTION

The ReqRes API is a demo API service that simulates various HTTP requests and responses for testing and development purposes. It provides a set of endpoints to interact with mock data, allowing developers to test their applications' functionality without accessing real user data.

ENDPOINTS AND FUNCTIONALITY:

The ReqRes API offers endpoints for performing CRUD operations on users and other mock resources. These endpoints include:

1. /api/users: This endpoint enables CRUD operations on user data, including retrieval, creation, updating, and deletion of user records.

2. /api/unknown: Similar to the /api/users endpoint, the /api/unknown endpoint allows CRUD operations on mock resources other than users.

3. /api/login: This endpoint simulates user authentication by providing an authentication token upon successful login.

4. /api/register: The /api/register endpoint facilitates user registration by accepting user data and creating new user records. This endpoint complements the authentication flow provided by the /api/login endpoint.

These endpoints collectively serve as a comprehensive toolkit for developers to mock API interactions and test various scenarios, including user management, resource manipulation, and authentication processes.

AUTHENTICATION AND AUTHORIZATION

The ReqRes API does not implement any authentication or authorization mechanisms. As a demo API, it does not require users to authenticate or obtain access tokens to access its endpoints. All endpoints are publicly accessible without any restrictions.

Since authentication and authorization are not applicable to the ReqRes API, developers can directly interact with its endpoints without the need for authentication tokens or special permissions. This simplifies the testing and development process, allowing developers to focus on testing application functionality rather than managing user authentication.

TEST FEATURE

The following test features should be covered under this test project:

1. Get all users.
2. Get a single user.
3. Create users.
4. Update users.
5. Partially update users.
6. Delete users.
7. Register users.
8. Login users.

SCOPE

ITEMS TO BE TESTED

To cover up this test task the following items must be tested:

Item to Test	Test Description
Get user with no parameters	This is conducted by GET request to the API. The request should return users data from the first page.
Get user with valid parameters	This is conducted by GET request to the API. The request should return requested users' data.
Get user with invalid parameters	This is conducted by GET request to the API. This test case should not pass and proper error response should be found. Response message could be "Not Found".

Create user with valid parameters	This is conducted by POST request to the API. With valid user email, first_name, last_name and the request will return a valid id in response body in postman.
Create user with invalid username and password	This is a POST request to the API. With invalid username and password, the request will not return any valid id in response body. It should return proper error message “Bad credentials”.
Create user with all empty body parameter	This is conducted by POST request to the API. This test case should not pass and proper error response should be found. Response message could be “Bad Request”. Because with all empty parameters, the request should not be processed in API. For each field the API shouldhave a proper validation.
Update with all correct parameters	This is conducted by PUT request to the API. In this case user will be passed through URL. This test case will update the user information of the provided ID. The response body will return all the updated information. If ID does not exists, then error message will be “Method Not Allowed”.
Delete user of existing id	This is conducted by DELETE request to the API. This test case will delete the provided ID and give proper response. The ID will be passed through URL. If ID does not exists then error message will be “Method Not Allowed”.
Get resources with no parameters	This is conducted by GET request to the API. The request should return user list from the first page.
Get resources with valid parameters	This is conducted by GET request to the API. The request should return requested recourse.
Get resources with invalid parameters	This is conducted by GET request to the API. This test case should not pass and proper error response should be found. Response message could be “Bad Request”.
Update resources all correct parameters	This is conducted by PUT request to the API. In this case user will be passed through URL. This test case will update the resource which is given in the body.
Update resources invalid parameters	This is conducted by PUT request to the API. This test case should not pass and proper error response should be found. Response message could be “Bad Request”.

Delete resources with valid parameters	This is conducted by Delete request to the API. In this case user will be passed through URL. This test case will Delete the resource which is given in the body.
Delete resources invalid parameters	This is conducted by Delete request to the API. This test case should not pass and proper error response should be found. Response message could be "Bad Request".
Register user with valid parameters	This is conducted by POST request to the API. With valid user email, first_name, last_name and the request will return a valid id in response body in postman.
Register user with invalid parameters	This is conducted by POST request to the API. This test case should not pass and proper error response should be found. Response message could be "Bad Request".
Login user with valid parameters	This is conducted by POST request to the API. With valid user email, first_name, last_name in the body and the request will return a valid token in response body in postman.
Login user with invalid parameters	This is conducted by POST request to the API. This test case should not pass and proper error response should be found. Response message could be "Bad Request".

ITEMS NOT TO BE TESTED

Non-functional test cases are not tested. Performance Test, Load Test, Stress Test are not covered in this test project.

TEST ENVIRONMENT AND INFRASTRUCTURE

REQUIRED INFRASTRUCTURE

- Software requirement: Postman v10.23.10, newman v6.1.1
- OS requirement: Windows 10/11
- Browser: Google Chrome
- Documentation: MS Word

TEST APPROACH

FUNCTIONAL TEST INCLUDING POSITIVE AND NEGATIVE TEST CASE USING AUTOMATION.

Process of Testing:

- Create all the test case in postman under a collection, write automation scripts in Tests section in postman. Also add an environment in postman.
- Open Collection Runner and run the collection.
- From postman Runner export the result, a JSON file will be found for the test result.
- A JSON file will be found for exporting the collection.
- Open CMD and run the JSON collection with newman (newman run "path/to/your-collection.json").
- Detail result will be seen in the CMD.

GENERATE TEST RUN REPORT USING NEWMAN.

Process of Test Run Report creation:

Newman is used to generate test run reports. Newman **htmlextra** reporter is used here.

1. Go to JSON file directory.
2. In CMD run command (newman run "path/to/your-collection.json" -r htmlextra --reporter-htmlextra-title "Title Name of the Report")
3. This will generate the report in the same directory under newman folder.

TEST PASS/FAIL CRITERIA

PASS CRITERIA

If the test case is passed then the status code, response message, status messages are checked. If the response body contains data, then these are also verified with assertions.

FAIL CRITERIA

If the test case is failed, then the same approach is applied. The test cases are failed intentionally because the outcomes do not match with expected outcomes. Outcomes means, the response status code, response status message and response body.

TEST DELIVERABLES

Test Plan, Test Case, Test Result JSON file and Postman test collection JSON file.

TEST SUSPENSION / RESUMPTION CRITERIA

If the API is up and running only then all the further functional test case will be executed. And if the API is not available then the test case will be suspended.

STAFFING / TRAINING NEEDS

Need proper idea of testing in postman. Also need know how to run postman collection JSON. If it's needed to integrate with CI/CD tool like CircleCI, individuals need to know how to work with CircleCI integrating with Git.

RISK AND MITIGATION

RISK	PRIORITY	IMPACT	MITIGATION
If the API endpoint is unavailable, then no test will result properly	Low	High	Devops team is responsible to keep the server always up and running
If the test case orders are changed then there might cause problem, and no proper result will be found.	Low	High	Test case order should not be changes. Just run the test case in an order as delivered. QA team will be responsible to keep order of test cases.
Delayed testing due to new issues	Medium	High	During the testing phase, it's common to uncover new defects that may require significant time to address. Unclear document specifications can also lead to defects during testing, which might escalate into larger issues requiring resolution time. Should these issues become critical and impact the project schedule, our defect and issue management procedures are in place to promptly address them

COMMUNICATION PLAN AND TEAM ROSTER

ROLE EXPECTATIONS

The following list defines in general terms the expectations related to the roles directly involved in the management, planning or execution of the test for the project.

SNO.	ROLES	NAME	CONTACT INFO
1	Project Manager		
2	Test Lead		
3	Business Analyst		
4	Development Lead		
5	Testing Team	Enam Hossain	
6	Development Team		
7	Technical Lead		

PROJECT MANAGEMENT

- Project Manager: reviews the content of the Test Plan and signs off on it.

TESTING PLANNING

- Ensure entry criteria before start the execution.
- Develop test plan and the guidelines to create test conditions, test cases, expected results and execution scripts.
- Provide guidelines on how to manage defects.
- Communicate to the test team any changes that need to be made to the test deliverables or application and when they will be completed.
- Provide functional (Business Analysts) and technical team to test team personnel (if needed).

TEST TEAM

- Develop test conditions, test cases, expected results, and execution scripts.
- Perform execution and validation.
- Identify, document, and prioritize defects according to the guidance provided by the Test lead.
- Re-test after software modifications have been made according to the schedule.
- Prepare testing metrics and provide regular status.

TEST LEAD

- Acknowledge the completion of a section within a cycle.
- Provide authorized to proceed with the next level of testing.
- Facilitate defective communications between testing team and technical / development team.

DEVELOPMENT TEAM

- Review testing deliverables (test plan, cases, scripts, expected results, etc.) and provide timely feedback.
- Assist in the validation of results (if requested).
- Support the development and testing processes being used to support the project.
- Certify correct components have been delivered to the test environment atthe points specified in the testing schedule.
- Keep project team and leadership informed of potential software delivery date slips based on the current schedule.
- Define processes/tools to facilitate the initial and ongoing migration of components.
- Conduct first line investigation into execution discrepancies and assist test executors in creation of accurate defects.
- Implement fixes to defects according to schedule.