Muhammad Ahsan Mehdi

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Test plan

UI & API

Contents

[Objective 2](#_Toc197459751)

[Tools & Technologies 2](#_Toc197459752)

[Scope 2](#_Toc197459753)

[UI Tests 2](#_Toc197459754)

[API Tests 2](#_Toc197459755)

[1. GET /posts 2](#_Toc197459756)

[2. GET /posts/{id} 3](#_Toc197459757)

[3. POST /posts 3](#_Toc197459758)

[4. PUT/PATCH /posts/{id} 3](#_Toc197459759)

[5. DELETE /posts/{id} 3](#_Toc197459760)

# Objective

The objective of the test is to test the UI of a public real website that contains file upload functionality and public APIs. The test suite aims to validate simple and edge cases for file upload. Apart from that, another aim is to test the authenticity of the public APIs named JSONPlaceholder.

# Tools & Technologies

The tool selected for the test suite is Playwright with the default test runner. The language utilized is JavaScript, and the test approach is BDD (Behavior-Driven Development).

# Scope

The test is designed to validate the UI and APIs.

## UI Tests

Following are the scenarios for file test suite:

* Open the URL
* Upload valid file
* Upload invalid file
* Upload large file
* Upload small file
* Upload file with different file types
* Upload file with variations of file name
* Upload file with different browsers

## API Tests

For each request following should be added to:

### 1. GET /posts

* Verify response status code is 200 (OK).
* Check response body is an array of posts.
* Validate structure of each post object (**userId**, **id**, **title**, **body**).
* Ensure number of posts matches expected count (100 posts).
* Measure response time.

### 2. GET /posts/{id}

* Verify response status code is 200 for valid IDs and 404 for invalid IDs.
* Check response body contains correct post object for valid ID.
* Validate structure of the post object.
* Ensure post matches expected data for given ID.

### 3. POST /posts

* Verify response status code is 201 (Created) upon successful creation.
* Test with valid data for successful post creation.
* Test with missing/invalid fields for appropriate error responses (400 Bad Request).
* Validate response body contains created post with **id**.
* Ensure post is created by retrieving it via GET.

### 4. PUT/PATCH /posts/{id}

* Verify response status code is 200 (OK) for successful updates.
* Test with valid data for successful post update.
* Test with invalid data for appropriate error responses.
* Validate response body contains updated post.
* Ensure changes are reflected when retrieving post via GET.

### 5. DELETE /posts/{id}

* Verify response status code is 200 (OK) for successful deletion.
* Ensure post is no longer retrievable via GET after deletion.
* Test deletion of non-existent post for appropriate error responses (404 Not Found).