Spotify Web API report

-API Testing using Postman-

Item	*Link*
GitHub Repository	Spotify API Testing
Postman Documentation	Spotify API Documentation
API Example	API Sample

Contents

INTRODUCTION

This report presents the testing of Spotify Web API endpoints, focusing on user library management and playlist manipulation. The goal of this testing was to verify the correct behavior of API requests.

Collection description

Collection name: Spotify Web API

Environment name: Spotify Web API

Collection structure:

Spotify Web API/

- Positive Endpoints/
  - Playlist/
  - User Profile/
  - Tracks/
- Negative Endpoints/
  - Playlist/

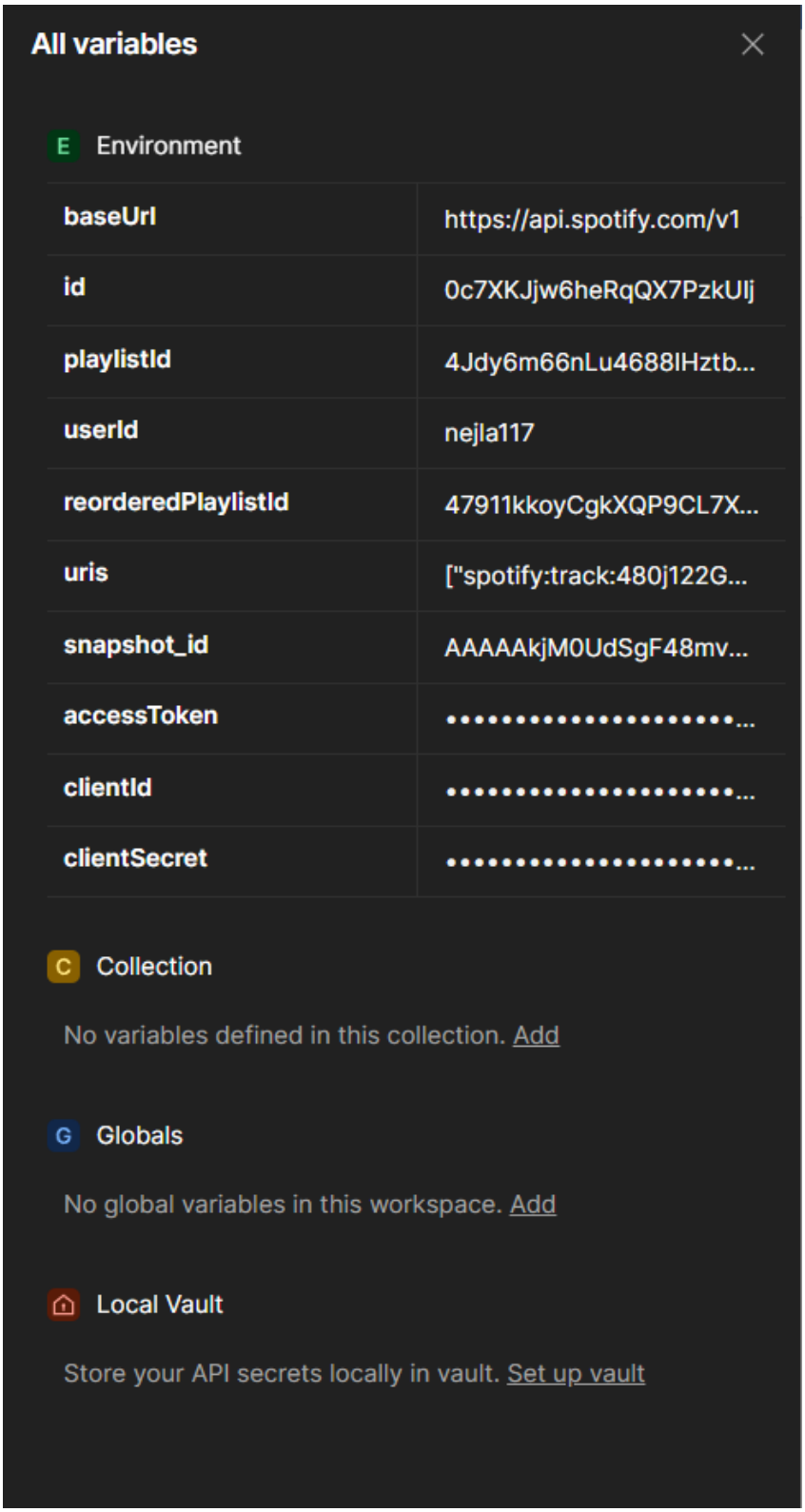

Figure 1 All variables

API Overview

{{baseUrl}} => https://api.spotify.com/v1

Endpoint	*Method*	*Description*
/me/playlists	GET	Retrieves all user's playlists.
/playlists/{{playlistId}}/tracks	PUT	Rearrange playlist's items.
/playlists/{{playlistId}}/tracks	POST	Adding items to the playlist.
/playlists/{{playlistId}}/tracks	DELETE	Removes items from playlist.
/users/{{userId}}/playlists	POST	Creates new playlist.
/me	GET	Retrieves user's informations.
/tracks/{{id}}	GET	Retrieves track by it's id.
/me/tracks	GET	Retrieves user's saved tracks.
/me/tracks	PUT	Saves Tracks for current user.
/me/tracks	DELETE	Removes user's saved tracks from library.

Authentication

Spotify uses **OAuth 2.0** for authentication. To access user-specific endpoints or modify playlists and library content, a **Bearer access token is required**. The Spotify access token **lasts for one hour**, it provides temporary access and must be refreshed or regenerated after expiration. Using Postman, you can obtain a new token by navigating to the Auth tab in the Spotify Web API collection and clicking Get New Access Token. The token must include the required scopes for the endpoints you want to test, such as:

- user-read-private
- user-read-email
- playlist-read-private
- user-library-read
- user-library-modify
- playlist-modify-public
- playlist-modify-private.

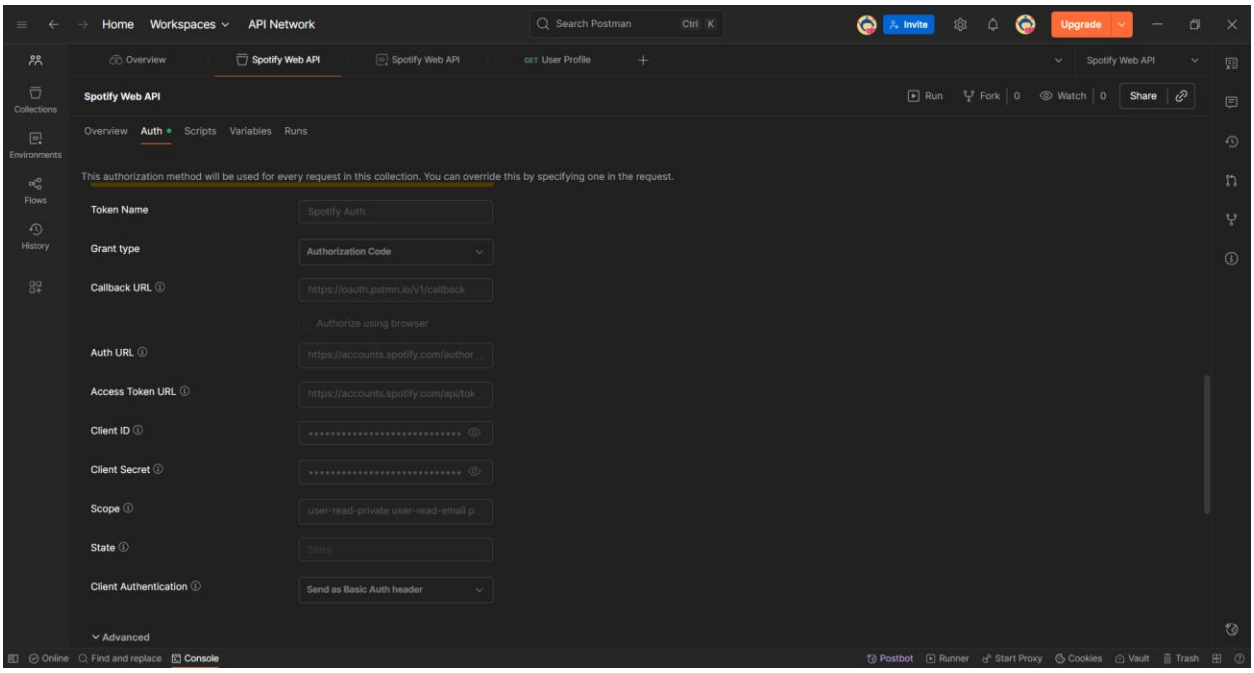

Figure 2 Authentication Configuration

When you click the **Get New Access Token** button in Postman, a browser window opens prompting you to log in to your Spotify account (Figure 3) and authorize the requested scopes. Once you log in and approve the permissions, Postman automatically captures the new token (Figure 4) and makes it available for use in your requests (Figure 5). This process ensures that your API calls have valid authentication to access user-specific endpoints.

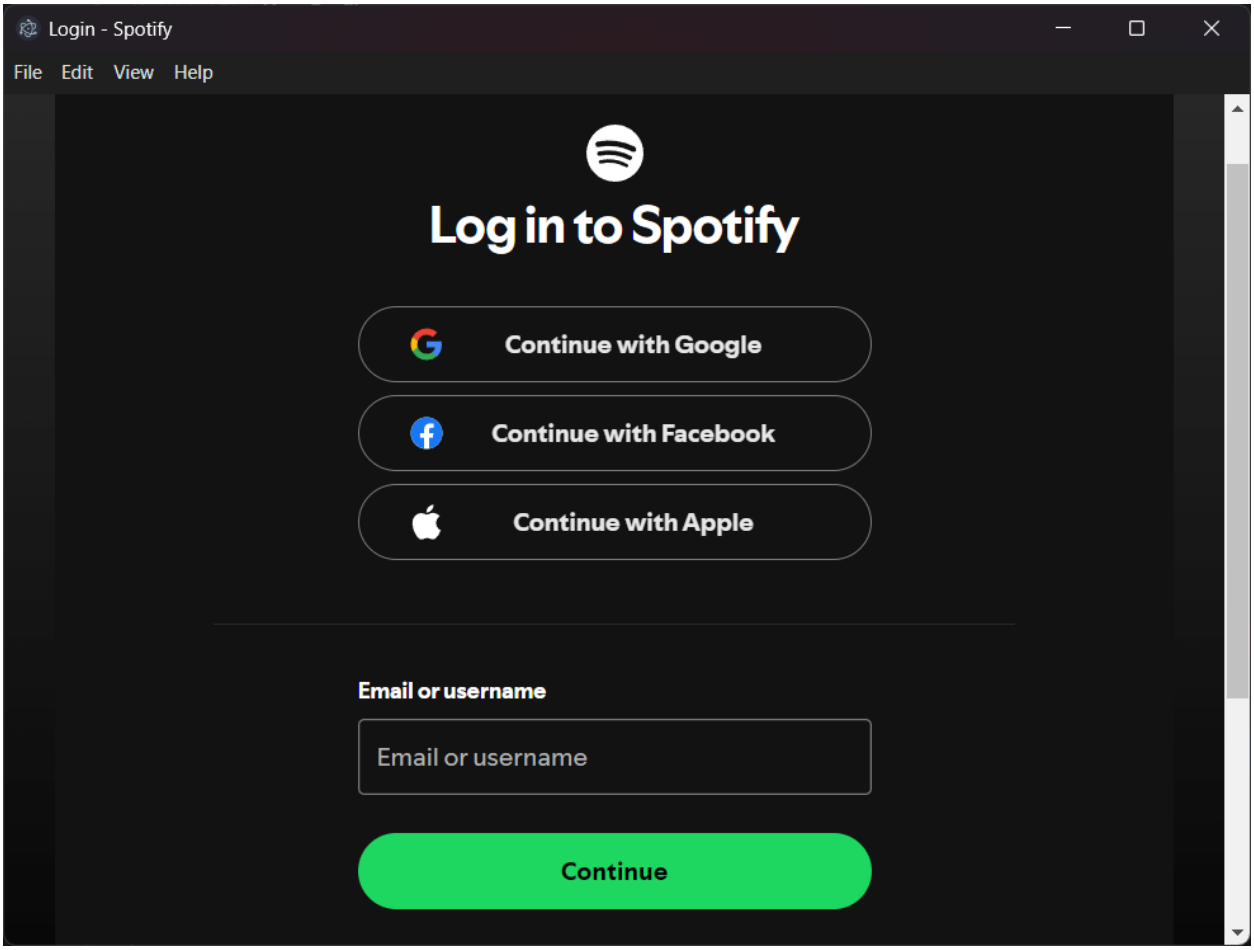

Figure 3 Login into Spotify Account

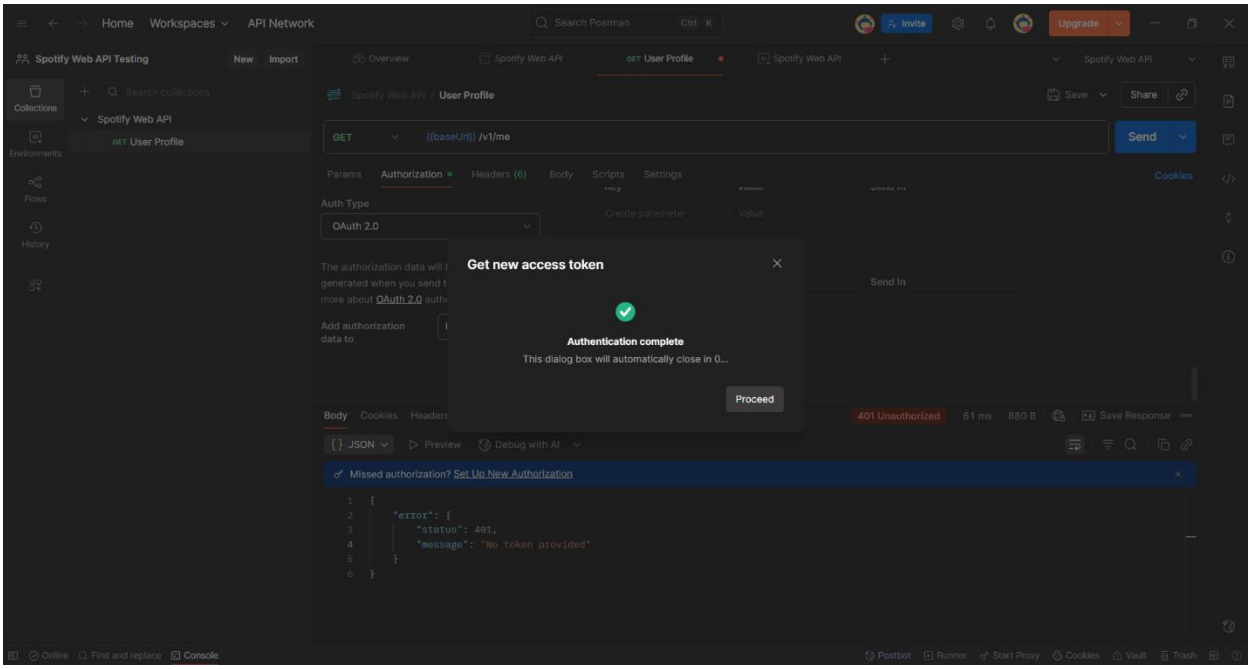

Figure 4 Authentication Completed

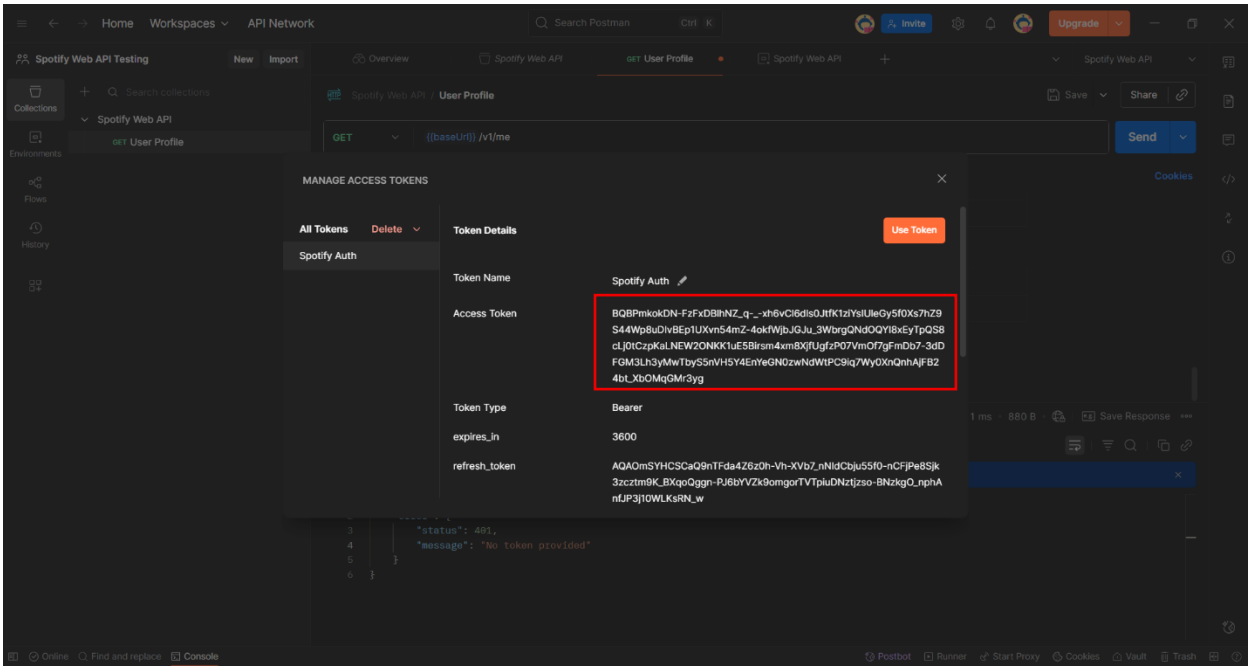

Figure 5 Access Token

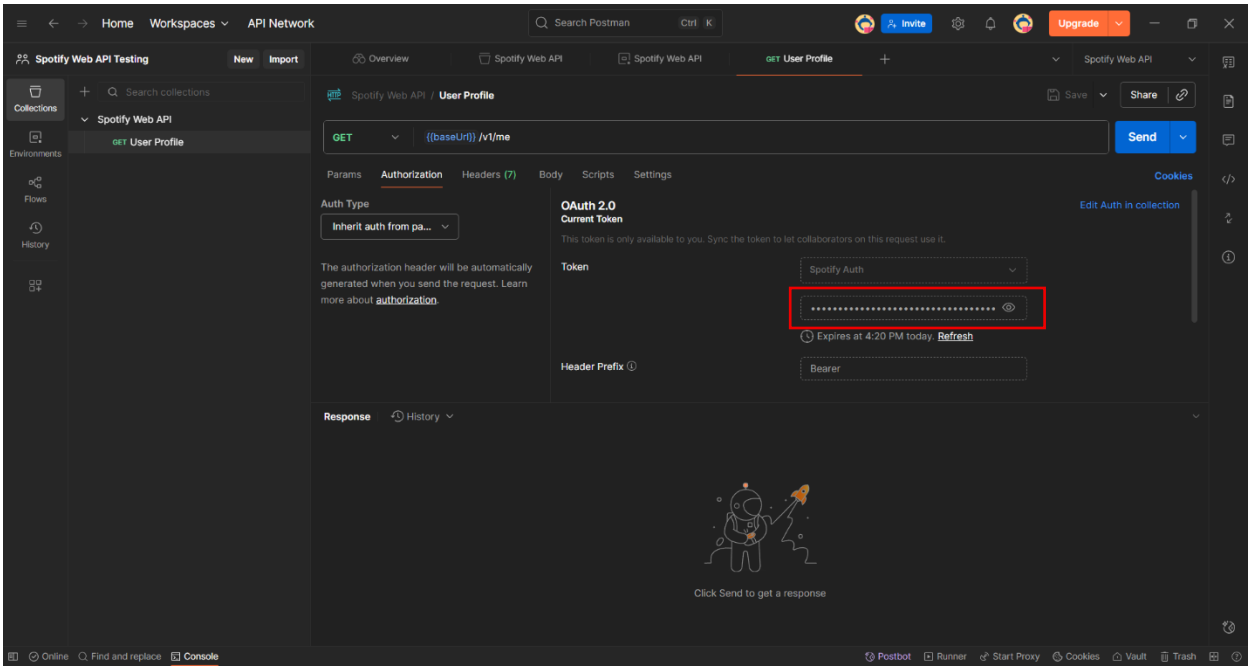

Figure 6 Example of OAuth2.0 use

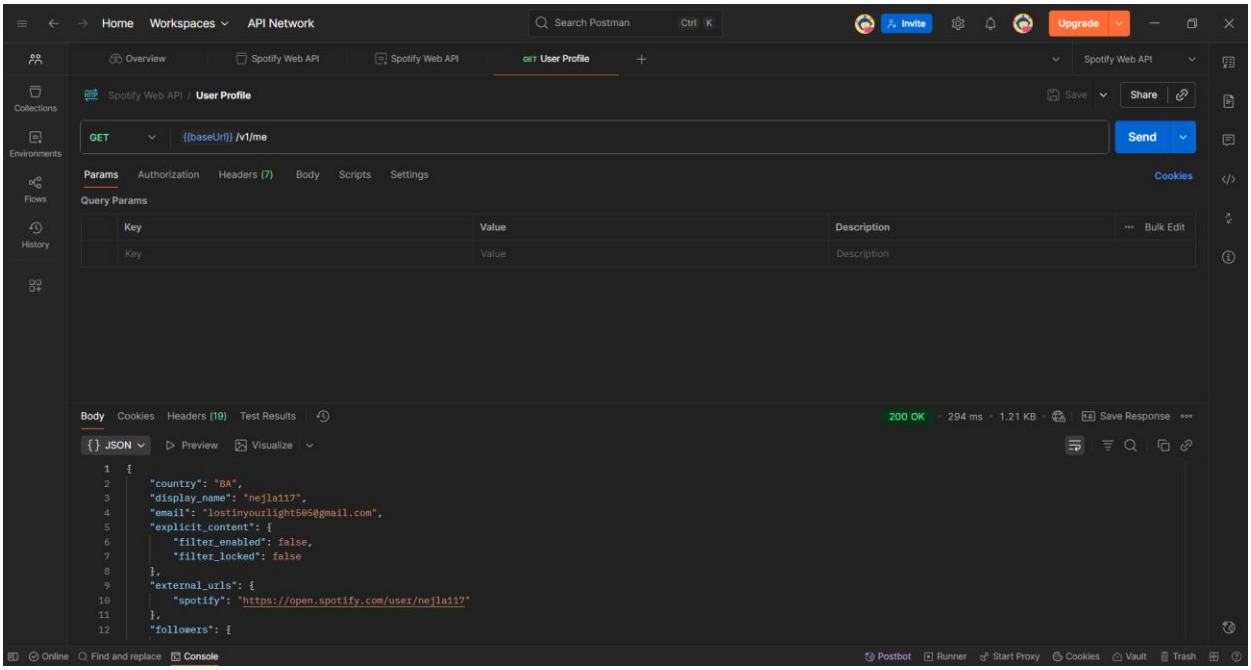

Figure 7 Authentication is successful

Positive endpoints tests

Functional Testing

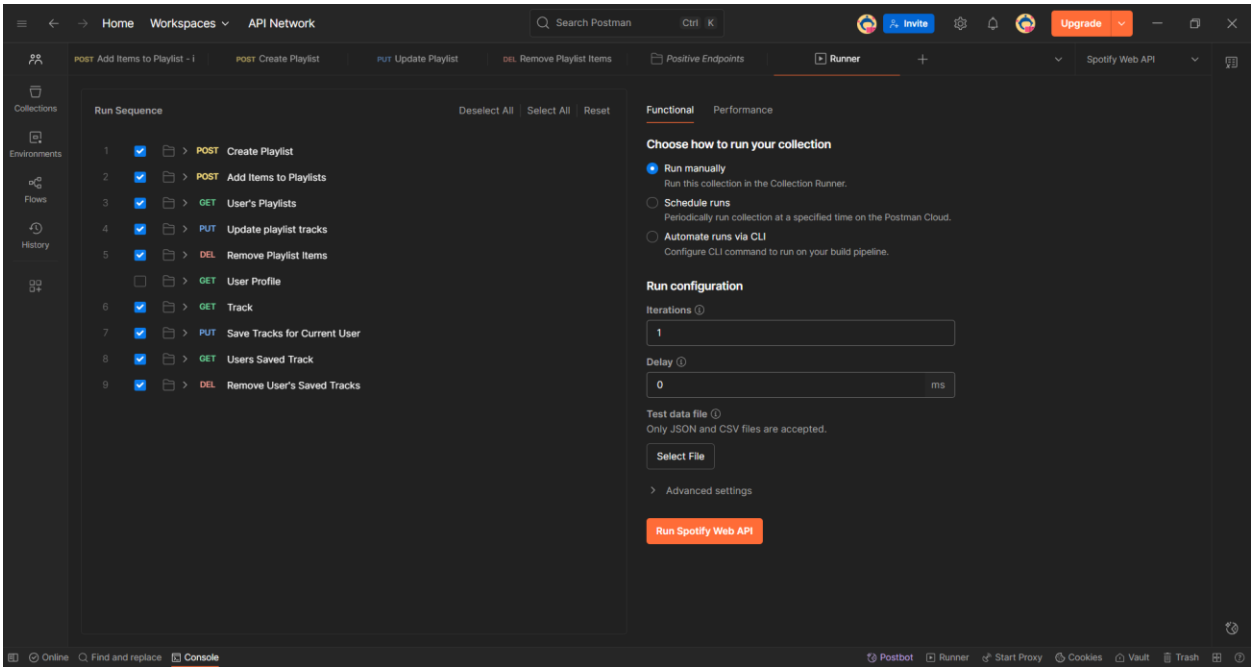

Figure 8 Functional testing set-up

Functional testing summary: MORAM POPRAVITI

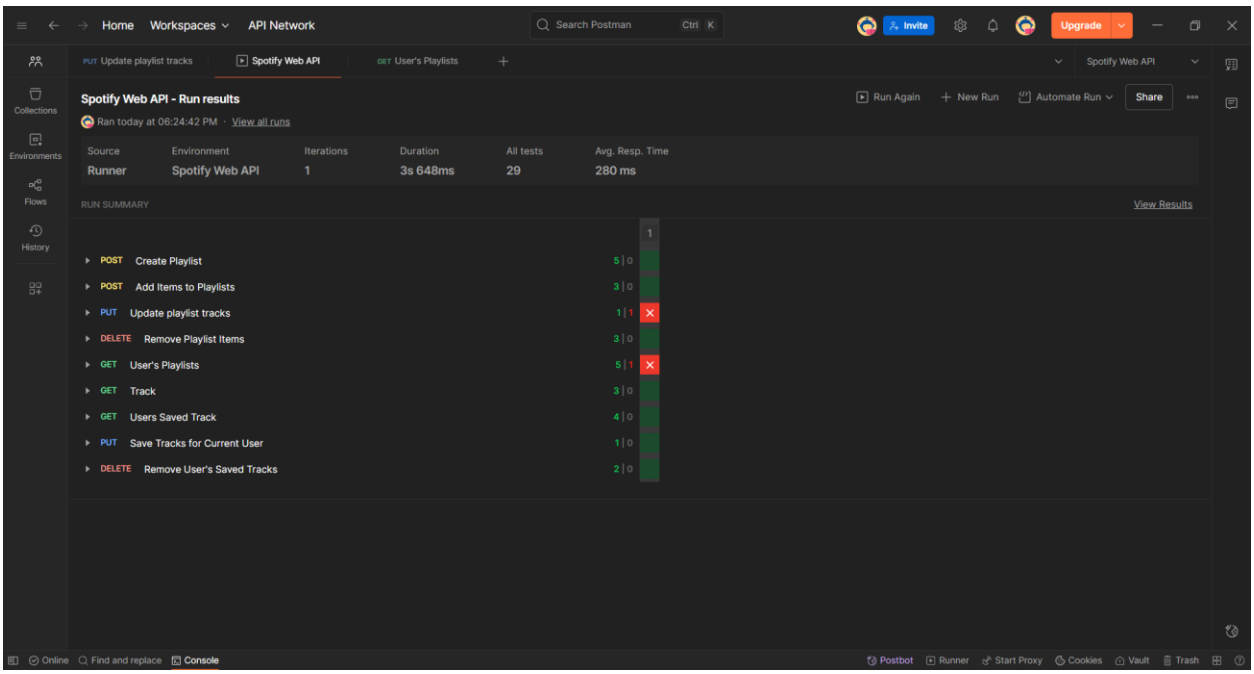

Figure 9 Functional Testing summary

Run informations:

Source	*Environment*	*Duration*	*All tests*	*Avg.Resp. Time [ms]*
Runner	Spotify Web API	3s 648ms	29	280

Passed	*Failed*	*Skipped*
27	2	0

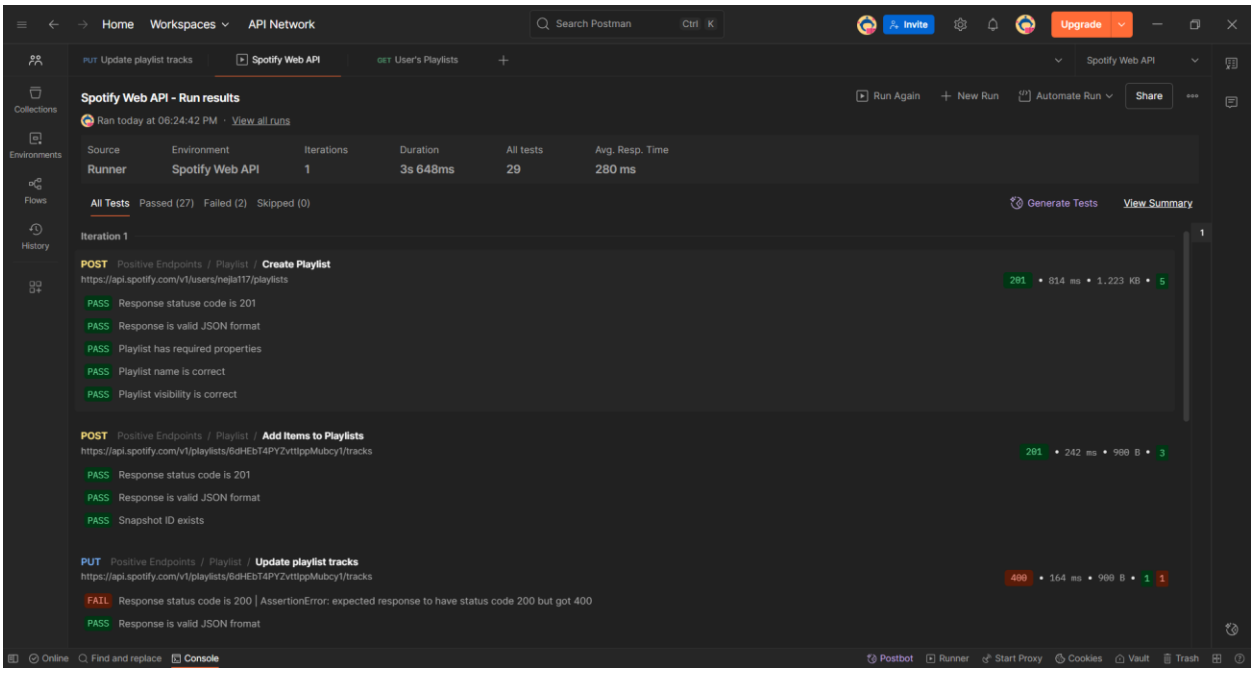

Figure 10 Results of Functional Testing

Failed Tests:

Test name: Response time is less than 200ms

Endpoint: {{baseUrl}}/me/playlists

Method: GET

Test Description: Verify that the API responds in less than 200ms, which is the expected performance requirement for this endpoint.

Expected Result: 200ms

Actual Result: 313ms

Status: Failed

Details:

The API returned a response, but the response time (313ms) exceeds the expected 200ms.

The response time may be influenced y factors such as:

- Current server load
- Number of items in the playlist
- Network latency

Test name: Response status code is 200

Endpoint: {{baseUrl}}/playlists/{{playlistId}}/tracks

Method: PUT

Test Description: Verify that tracks reorder is successful.

Expected Result: 200 OK

Actual Result: 400 Bad Request

Status: Failed

Details:

Test failed because only one track existed in the playlist at the time of testing, which makes reordering impossible.

The API returned a response, but the response was 400 Bad Request.

Performance Testing

Performance testing was conducted on the Spotify API endpoints related to playlist. The goal was to measure response time, throughput and stability under load.

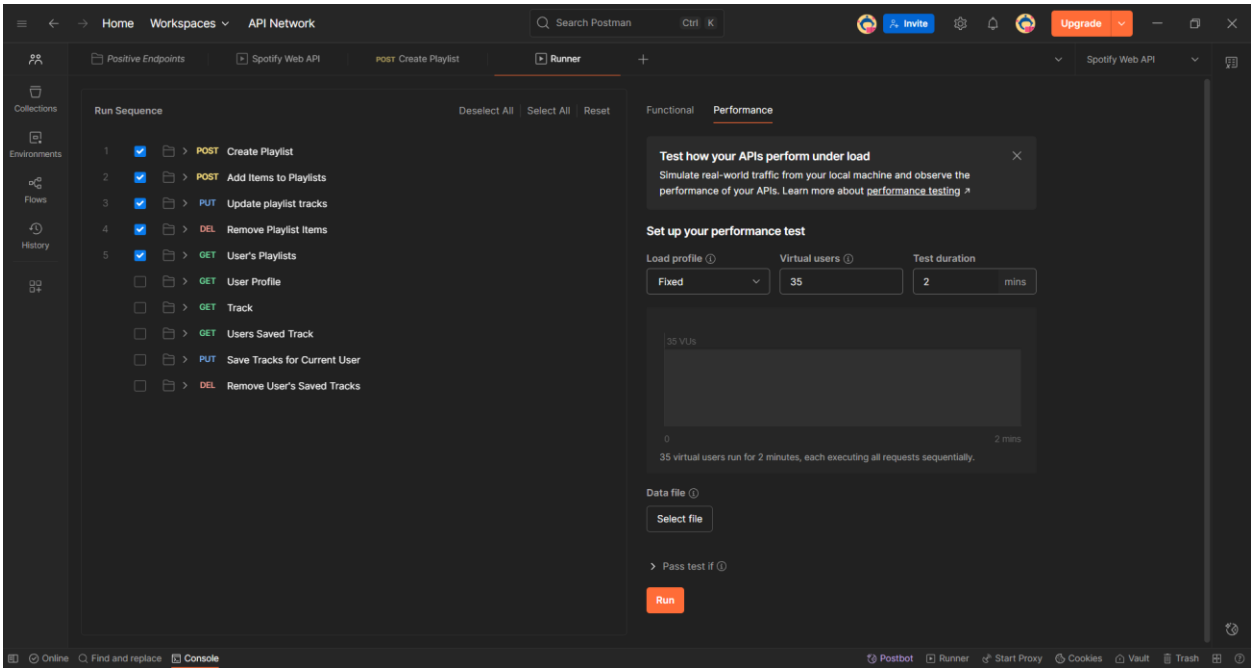

Figure 11 Performance testing on the Spotify API endpoints related to playlist

Load Profile	*Number of Virtual Users*	*Test duration [min]*
Fixed	50	2

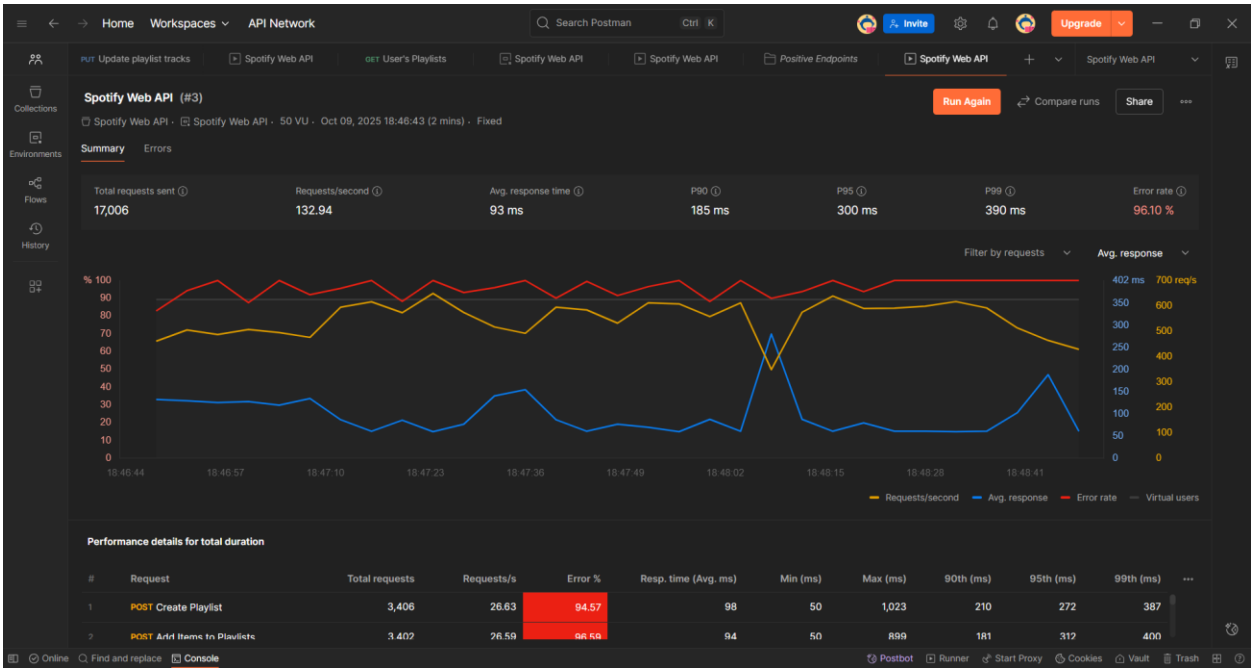

Figure 12 Performance Testing - results

Errors:

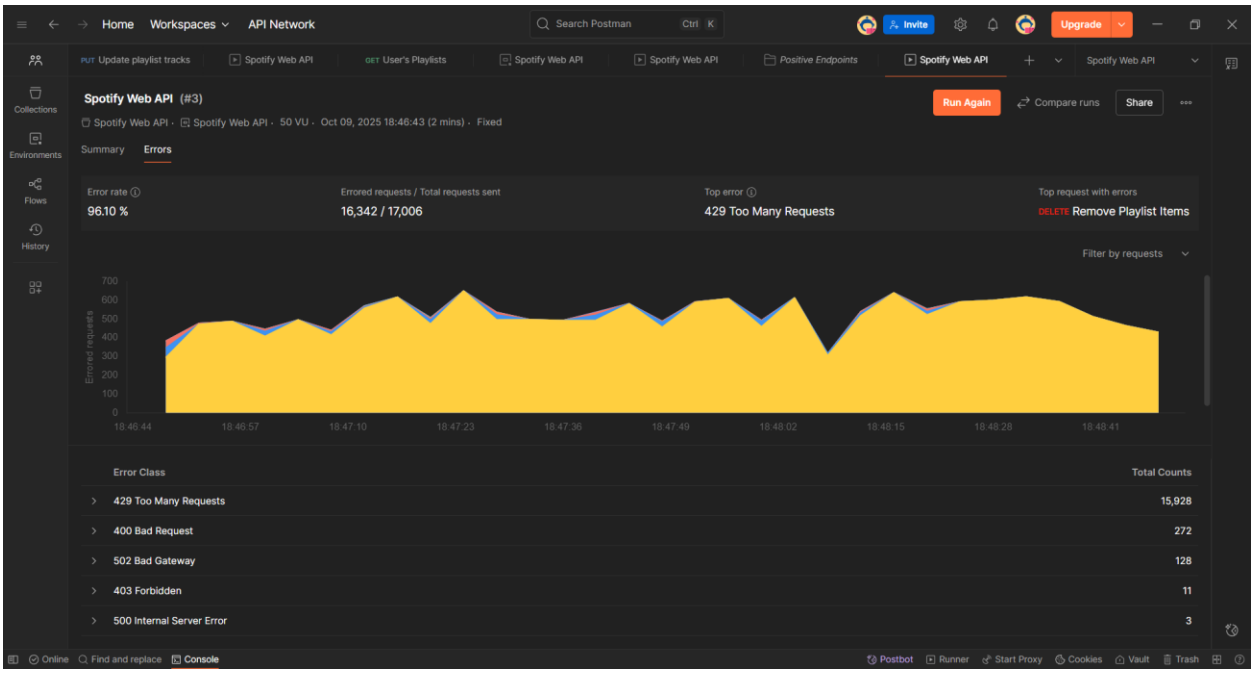

Figure 13 Errors

The average response time (93ms) and P90/P95 latencies are within acceptable limits for typical user interactions. Error rate of 96.10% is critically high, indicating that the system cannot handle this load.

The high error rate may be caused by:

- Overloading the API with too many concurrent requests

Negative endpoints tests

An example of negative endpoint test is reordered tracks that are out of bounds. The main goal of this test was to verify how the API handles a request to reorder tracks when the provided range of tracks is outside the valid bounds of the playlist. The playlist currently contains only one track. The API should return 400 Bad request status with an error message indicating tat the selected tracks are out of bounds. This negative test case confirms that the API properly validates track positions during the reorder operation and prevents invalid index manipulation.

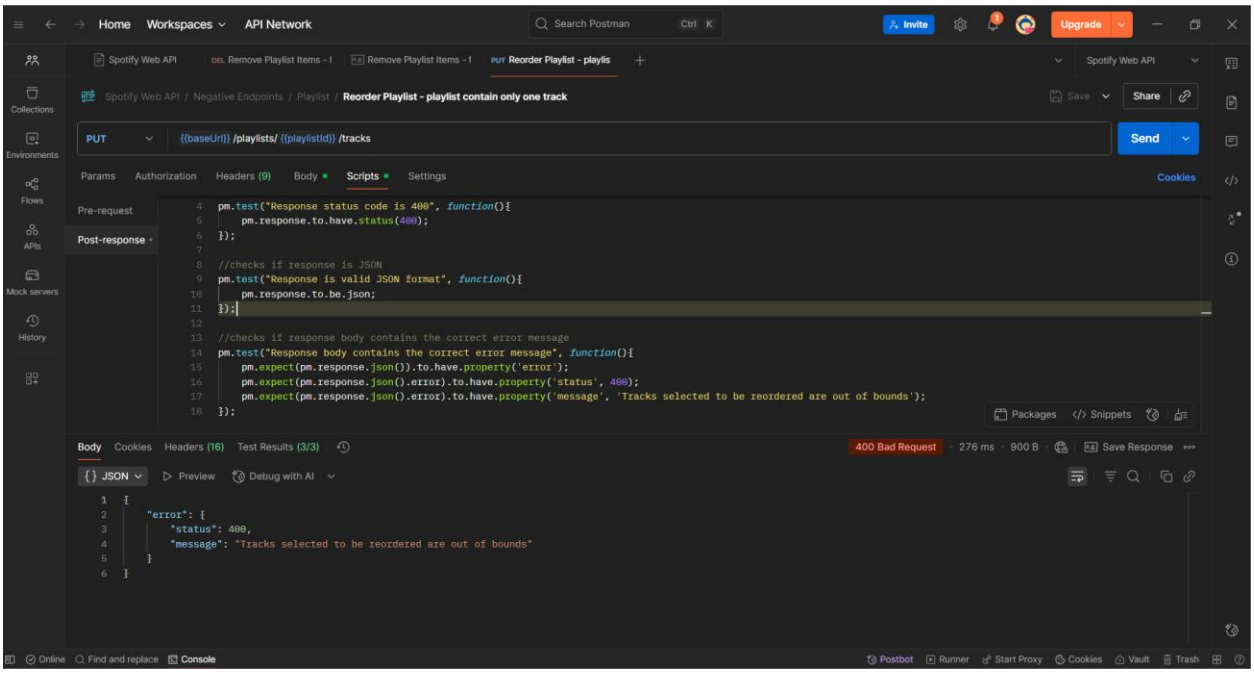

Figure 14 Response

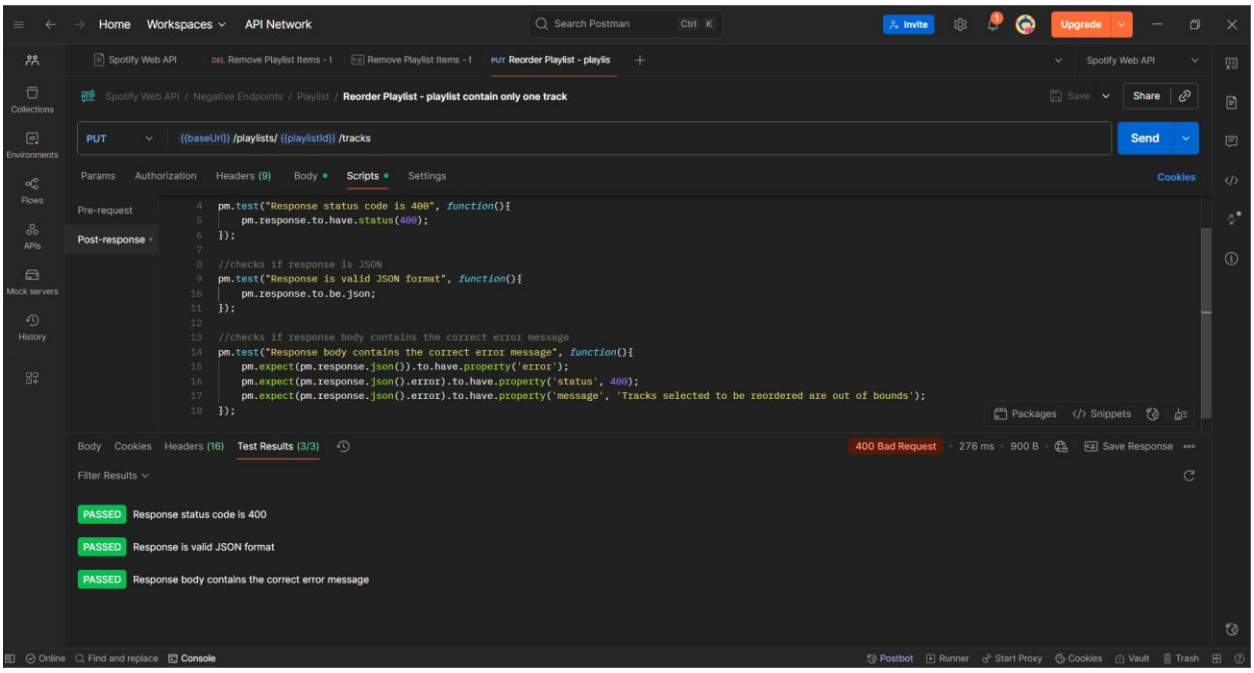

Figure 15 Negative endpoint tests results

Attachments