

Ex Banking Services

API Integration and Testing

Overview

The purpose of this document is to explain how API integration and API testing have been conducted on the Ex-Banking services. This document is containing one or more of the following.

- Used tools and languages
- Installation instructions
- Identified nonfunctional and functional test cases
- Configuration of Mock service
- Process of nonfunctional test case automation

Used Tools and Languages

Used 'Postman' as the tool to integrate and automate the API test assert. And for the mocking, the APIs used 'Postman' as the tool. When asserting the APIs use JavaScript as the language.

Since 'Postman' is not directly supported to automate the nonfunctional test case that I have chosen, decided to use the 'K6' tool to test out the performance of the created APIs. To write down the identified functional and nonfunctional test cases used 'Google sheet'.

Installation Instructions

1. Install Postman
 - a. <https://www.postman.com/downloads/>
2. Install K6
 - a. <https://k6.io/docs/get-started/installation/>

Identified Nonfunctional and Functional Test Cases

Based on the given instruction and the APIs identified and designed the functional and non-functional test cases in the google sheet. (Test case document has attached along with this.)

Selected Functional Test Cases for the Automation

Create User

1. Send create user request with valid first name, last name, date of birth, nationality, username, password, national identity number and email
2. Send create user request with invalid email
3. Send create user request without username

Deposit

1. Send deposit request with valid first name, last name, account number, amount and national identity number
2. Send deposit request with invalid account number
3. Send deposit request with amount which is greater than to the balanced amount

Withdraw

1. Send withdraw request with valid first name, last name, account number, amount and identification number
2. Send withdraw request without account number
3. Send withdraw request with amount as a floating number

Get Balance

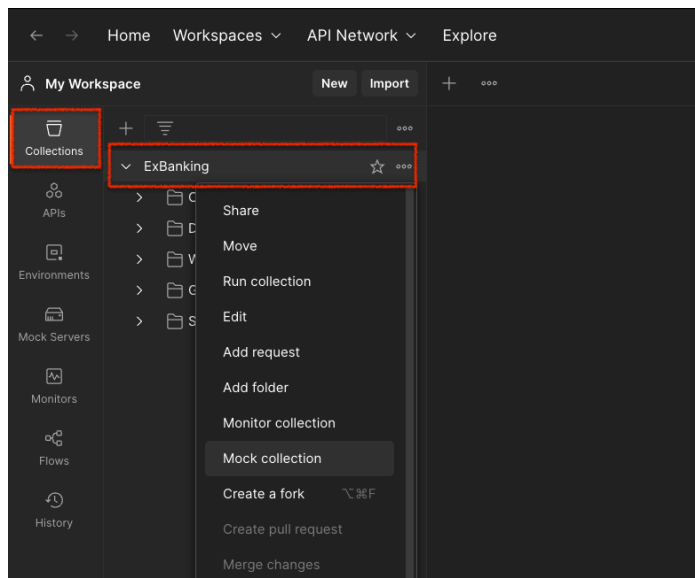
1. Send get_balance request with valid first name, last name, account number and identification number
2. Send get_balance request with invalid account number

Send

1. Send request with valid first name, last name, remarks, receiving account number, sending account, identification number, amount
2. Send request with using same account number for sending and receiving account numbers
3. Send request with minus amount

Configuration of Mock service

- Import shared postman collection and environment variables file.
 - The shared collection contains a mock service endpoint.
 - Follow the below steps to create a new mock service endpoint.
1. Right-click on the collection from the top right corner “more” icon and select “Mock collection”.



2. Insert “Mock Server Name” and select the collection. Create Mock Server button.

My Workspace

New Import

Create mock server

No Environment

Collections

- ExBanking
 - Create User
 - Deposit
 - Withdraw
 - Get Balance
 - Send

APIs

Environments

Mock Servers

Monitors

Flows

History

Create a mock server

Select collection to mock 2. Configuration

Mock Server Name

ExBankingMockService

Collection

ExBanking

Environment

ExBanking App

☒ Save the mock server URL as an new environment variable

Simulate a fixed network delay

No delay selected

☐ Make mock server private

Back

Create Mock Server

3. Copy and paste the values to environment variables,

- url = full URL value
- gateway-host = only the domain name

ExBankingMockService

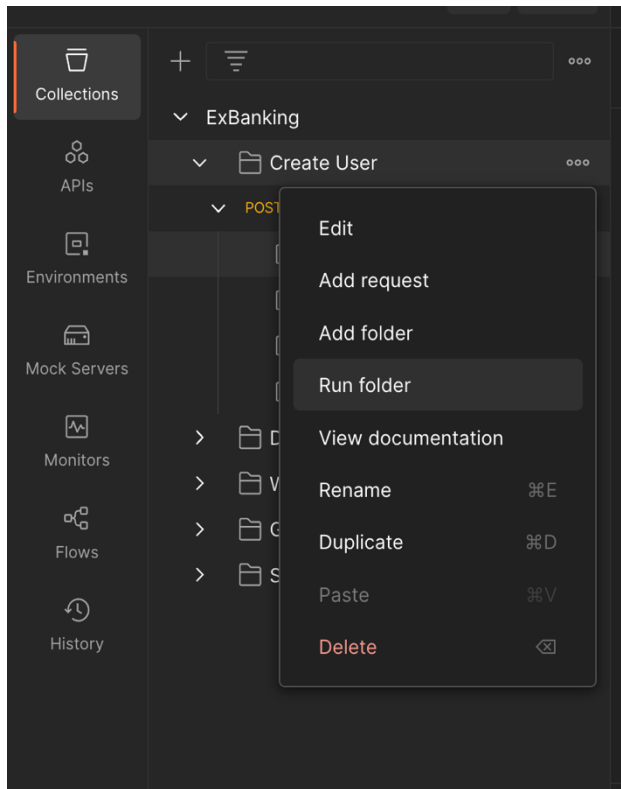
Filter variables

	VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE		Persist All	Reset All
<input checked="" type="checkbox"/>	url	default	https://ac8f51c5-fb41-4bde-92cb-e...	https://ac8f51c5-fb41-4bde-92cb-eda8f4332ac...			
<input checked="" type="checkbox"/>	protocol	default	https://	https://			
<input checked="" type="checkbox"/>	gateway-host	default	ac8f51c5-fb41-4bde-92cb-eda8f43...	ac8f51c5-fb41-4bde-92cb-eda8f4332ac4.mock...			
<input checked="" type="checkbox"/>	gateway-url	default	/rest/ex_banking/v1/	/rest/ex_banking/v1/			
<input checked="" type="checkbox"/>	url-resource	default					
<input checked="" type="checkbox"/>	url-method	default					
<input checked="" type="checkbox"/>	first_name	default					
<input checked="" type="checkbox"/>	last_name	default					
<input checked="" type="checkbox"/>	date_of_birth	default					
<input checked="" type="checkbox"/>	nationality	default					
<input checked="" type="checkbox"/>	username	default					
<input checked="" type="checkbox"/>	password	default					

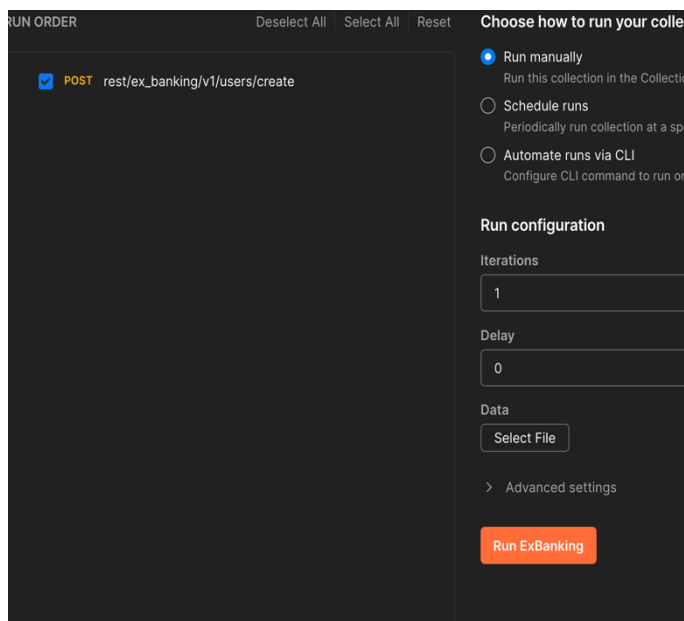
Execute the Mock Service

To execute the mock service use file from the test data folder.

1. Right-click on the folder inside the collection from the top right corner “more” icon and select “Run Folder”.



2. Select the correct JSON file from the folder and run the tests.



- Following results can be seen in the postman; similarly, you can execute for other test files.

Source Runner	Environment	Iterations	Duration	All tests	Avg. Resp. Time
ExBanking App	ExBanking App	4	3s 175ms	4	711 ms

Iteration	Test Name	Status	Message	Response Code	Duration	Size
Iteration 1	POST rest/ex_banking/v1/users/...	Pass	Send create user request with valid first name, last name, age, nationality, username, password, identity number and email.	201 Created	1368 ms	403 B
Iteration 2	POST rest/ex_banking/v1/u...	Fail	Send create user request with invalid email.	400 Bad Request	693 ms	536 B
Iteration 3	POST rest/ex_banking/v1/u...	Fail	Create user request without username.	400 Bad Request	391 ms	534 B
Iteration 4	POST rest/ex_banking/v1/u...	Fail	Create user request with already registered username.	400 Bad Request	390 ms	550 B

Process of Non-functional Test Case Automation

As explained in the Used Tools and Languages, for the load test use the 'K6' performance tool. Exported Postman API collection has been integrated with the K6 and runs the performance testing script. Here attached the screenshot of the results after executing the performance test.

Selected Test Case as the Non-functional Test case

- Send request with valid first name, last name, remarks, receiving account number, sending account, identification number, amount

```
admin@Admins-MacBook-Pro Final % k6 run k6-script.js

      A R K6
    .io

execution: local
script: k6-script.js
output: -

scenarios: (100.00%) 1 scenario, 100 max VUs, 40s max duration (incl. graceful stop):
  * default: 100 looping VUs for 10s (gracefulStop: 30s)

running (10.3s), 000/100 VUs, 3732 complete and 0 interrupted iterations
default ✓ [=====] 100 VUs 10s

data_received.....: 1.0 MB 98 kB/s
data_sent.....: 1.6 MB 157 kB/s
http_req_blocked.....: avg=19.91ms min=0s med=3µs max=772.51ms p(90)=5µs p(95)=8µs
http_req_connecting.....: avg=7.46ms min=0s med=0s max=306.05ms p(90)=0s p(95)=0s
http_req_duration.....: avg=250.81ms min=212.88ms med=249.85ms max=325.3ms p(90)=268.93ms p(95)=274.2ms
http_req_failed.....: 100.00% ✓ 3732 ✗ 0
http_req_receiving.....: avg=51.72µs min=7µs med=45µs max=2.31ms p(90)=75µs p(95)=91µs
http_req_sending.....: avg=21.2µs min=2µs med=18µs max=484µs p(90)=30µs p(95)=41.44µs
http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=250.73ms min=212.82ms med=249.78ms max=325.19ms p(90)=268.86ms p(95)=274.12ms
http_reqs.....: 3732 361.782331/s
iteration_duration.....: avg=270.87ms min=213.04ms med=250.09ms max=1.09s p(90)=269.94ms p(95)=277.11ms
iterations.....: 3732 361.782331/s
vus.....: 100 min=100 max=100
vus_max.....: 100 min=100 max=100

admin@Admins-MacBook-Pro Final %
```