Author : Kelvin Au

BBPOS JWT Token Program design Spec

**Table of Content**

[Application tools used for the program 3](#_Toc132739507)

[Technical structure of the program 4](#_Toc132739508)

[JWT token Authentication mechanism 5](#_Toc132739509)

[Spring boot Program structure 7](#_Toc132739510)

[Integration test result 10](#_Toc132739511)

# Application tools used for the program

* Eclipse Version: 2023-03 (4.27.0)

For maven project setup on the spring boot program, maven and Git built in

* Git Hub

URL: <https://github.com/BBPOSTest/login>

Act as Spring boot program repository

* Docker Desktop for windows

Use to create container and thus the mongo DB as persistent storage for the spring boot program as demo purpose

*MongoDB created information:*

User Login account = kelvin

User Login Password = 123456

MongoDB database name = testDB

MongoDB used for Program demo = Users

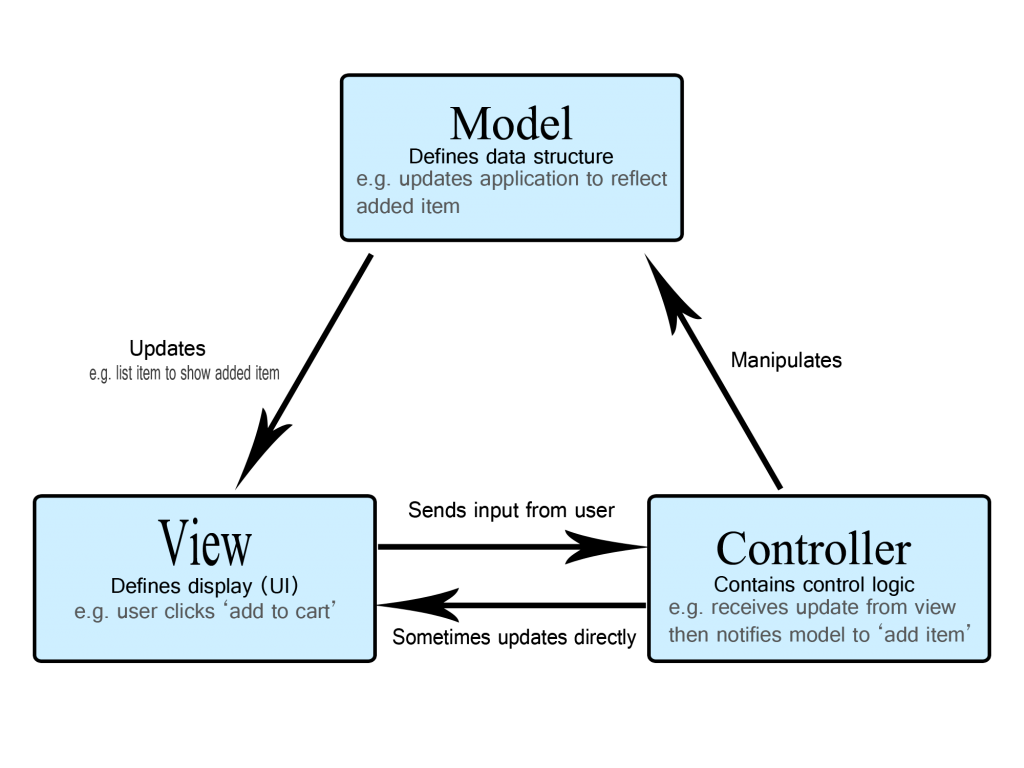
* NoSQL Manager for MongoDB

Use as DB client to query mongo DB creation from the docker for verification

* Postman windows 64-bit version

Use to test the HTTP request as integration test from the URL call to the servers and to mongo DB action. To prove the correctness of the spring boot program

# Technical structure of the program



The spring boot program is built according to the 3-Tier architecture with MVC model for the whole JWT authentication and assessment logic for the program flow.

When the spring boot receive the API request from the following HTTP URL:

|  |  |  |
| --- | --- | --- |
| Methods | Endpoints | Actions |
| POST | /api/auth/signup | Signup new account |
| POST | /api/auth/login | Login an account |
| GET | /api/test/all | Retrieve public content |
| GET | /api/test/user | Access user’s content |

The action will be received by the mapping controller class of the spring boot program. According to different action request during invoke, different business service injection bean will be injected for the action.

After that, A mongo DB repository bean will be injected to the different business service bean as a data object to process the action to the created mongo DB in the backend side. The actions to be done on the mongo DB including creation record and querying record etc.

# JWT token Authentication mechanism

Validate login, password, email

Generate token

Set refresh token to cookie

Client

HTTP API request

Server

POST /api/auth/login

Body: {login, password, email}

Return JWT token for client to use in next API call, set in the request header

JWT token authentication mechanism is shown in the above sequence diagram.

When a client login to the spring boot program via the API call “/api/auth/login”, the program will check with the mongo DB on the existence of the account according to email, user account, password input information.

After the verification succussed, the spring boot program will generate a JWT token with 30 minutes expiration time and then will return this token to the client to indicate the successfully login.

The client will then able to use this generated token to set in the HTTP request header “authentication” in another all for their business action.

Mongo database table: **User** is created in the following format

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Type | Uniqueness | Null / Not Null |
| id | String | Y | Not Null |
| Email | String | Y | Not Null |
| User Account | String | N | Not Null |
| User Password | String | N | Not Null |
| Status | Integer | N | Not Null |

# Spring boot Program structure

Following tables listed out the package and class structure of my design spring boot program:

|  |  |
| --- | --- |
| Package | Description |
| com.kelvin.application | The core package of the source. Classes of the spring boot start program; Junit test is placed under this package |
| com.kelvin.application.db | Package to place data object in the MVC model. It contains class to represent data object for Mongo database and the repository for the connection between mongo DB and spring boot program |
| com.kelvin.application.filter | The package contains Web Security filter to filter and handle the JWT token authentication logic. |
| com.kelvin.application.service | Business injection bean as a micro-service for the spring boot program to handle the input API request and process and access different Mongo DB function to fulfill the business requirement |
| com.kelvin.application.View | It is the view object in the MVC model, Controller class is created to handle different API URL mapping request and forward to the business injection bean to handle accordingly. |

|  |  |
| --- | --- |
| Package: com.kelvin.application | |
| Class | Description |
| Application | Annotated with @SpringBootApplication, a core class to start up the whole program, start the tomcat server. The entry point of the spring boot application. |
| ApplicationTests | Annotated with @SpringBootTest, it is the class used for performing Junit test of the spring boot application |

|  |  |
| --- | --- |
| Package: com.kelvin.application.db | |
| Class | Description |
| User | Class annotated with @Document annotation; it represents the User table in the Mongo DB which will use it as demo in this spring boot program |
| UserRepo | Interface annotated with @Repository annotation and extend the parent interface of MongoRepository. The interface for us to interaction between Mongo DB and the spring boot program |

|  |  |
| --- | --- |
| Package: com.kelvin.application.filter | |
| Class | Description |
| AuthorizationCheckFilter | A class extend from OncePerRequestFilter, act as a web security filter to handle the JWT token authentication for each input HTTP request. |
| SecuirtyConfigure | A configuration bean to register the above filter to the spring boot program in order to handle each input HTTP request. |

|  |  |
| --- | --- |
| Package: com.kelvin.application.service | |
| Class | Description |
| UserService | A Service bean to handle the whole business logic of the spring boot program. |

|  |  |
| --- | --- |
| Package: com.kelvin.application.Views | |
| Class | Description |
| LoginServiceController | A Controller class to handle the below post HTTP request action:   1. /api/auth/signup 2. /api/auth/login |
| LoginTestController | A Controller class to handle the below get HTTP request action:   1. /api/test/all 2. /api/test/user |

Following two configurations file in the spring boot program helps to configure the program setup:

1. pom.xml

The file to depend the dependency library required to setup the spring boot program. This file would be auto reference when the program startup

1. application.properties

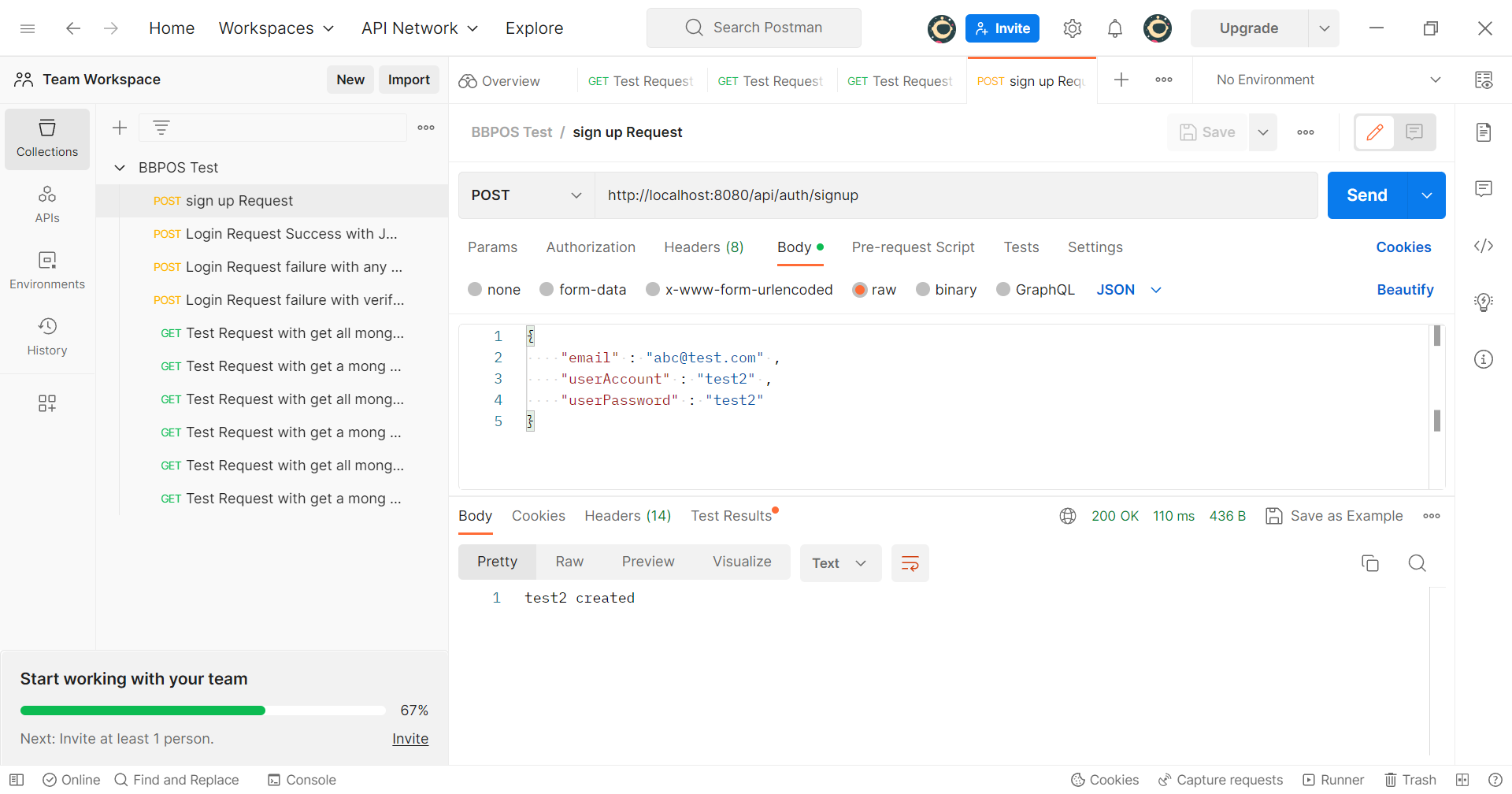
The Mongo DB connection string and setting are defined in this properties file

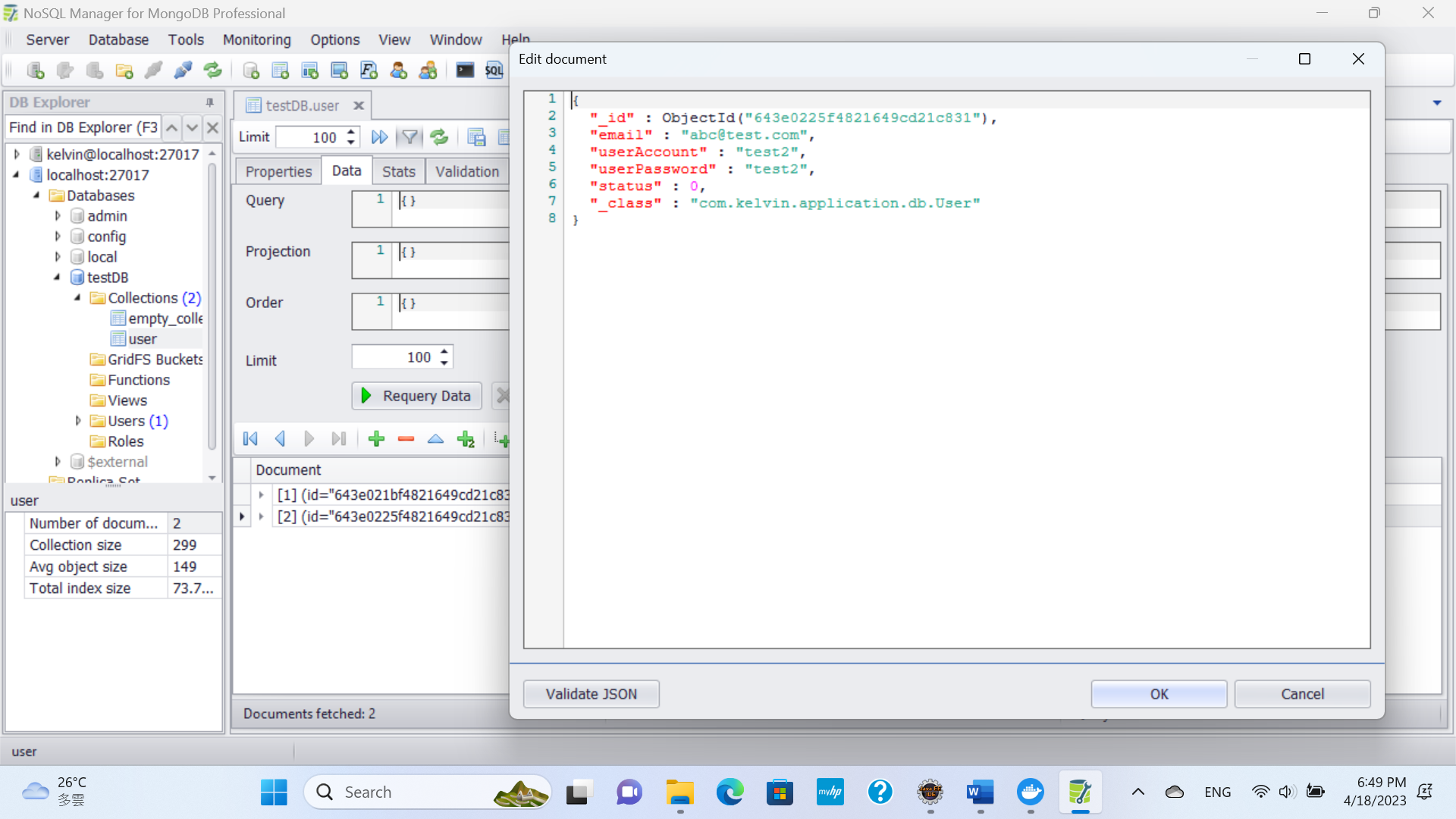
# Integration test result

Following is the integration test result for each scenario of HTTP request API via Postman to spring boot program. All the testing case results positive:

*Scenario 1:*

To create User Profile record via HTTP POST API request /api/auth/signup

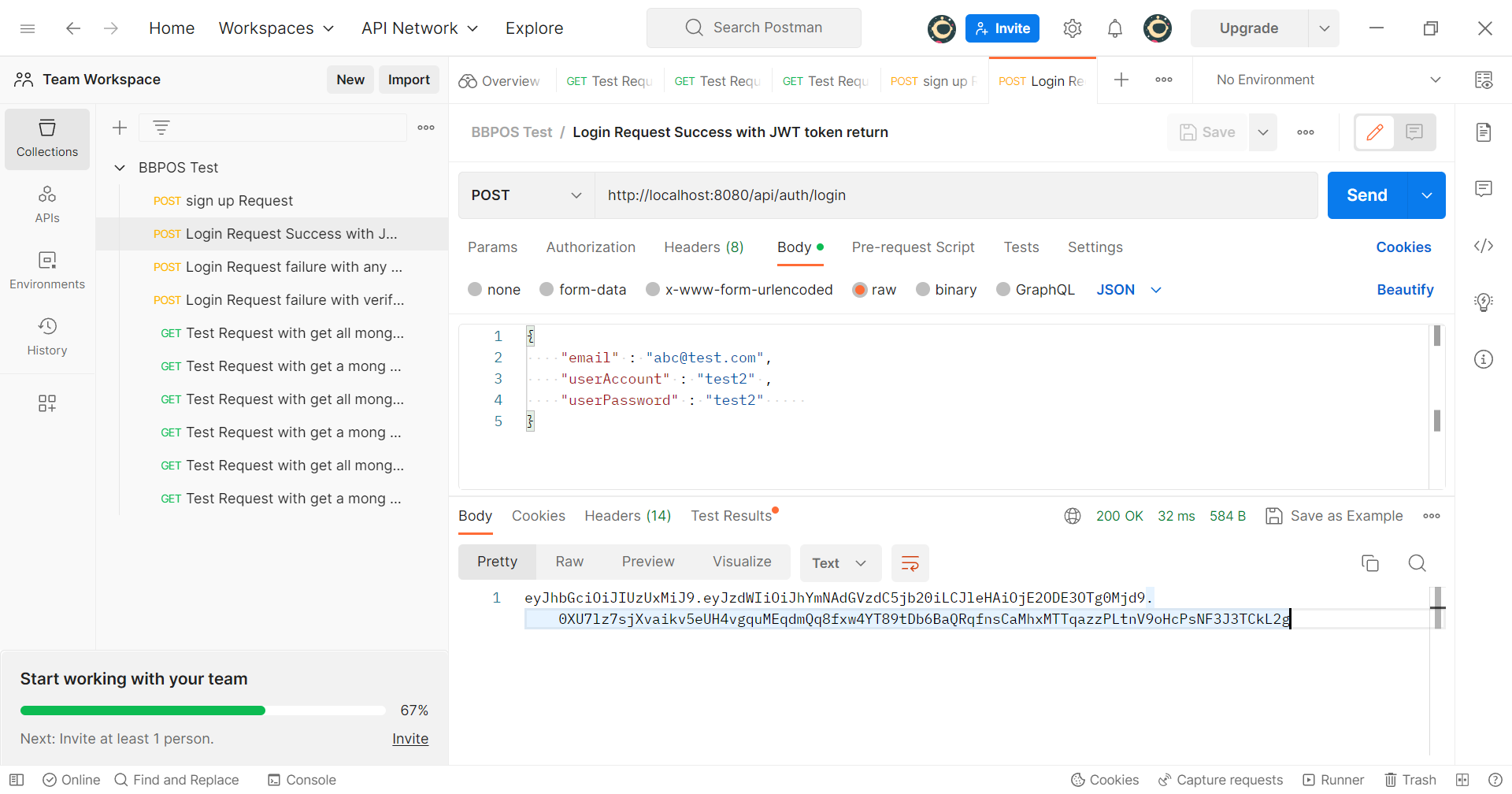




User Profile record is succussed created in Mongo DB.

*Scenario 2:*

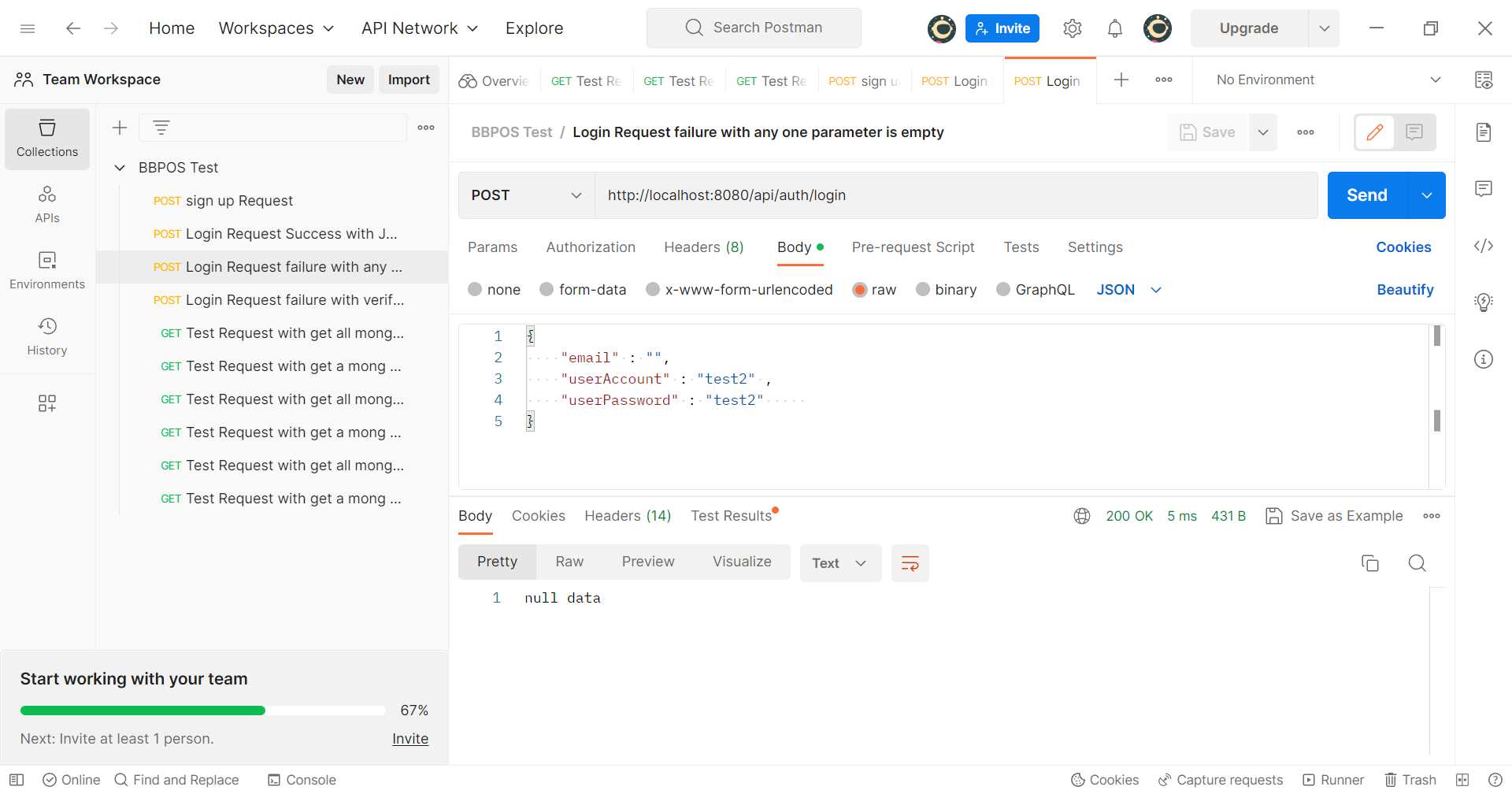
User with account “test2” is successfully login via HTTP POST API request /api/auth/login.



The account is successfully login with JWT token generated and return from API. This token can be used for further business action in another HTTP request. The token will be set to the “Authorization” header for further action. 30 minutes timeout expire condition is set to this generated JWT token.

*Scenario 3:*

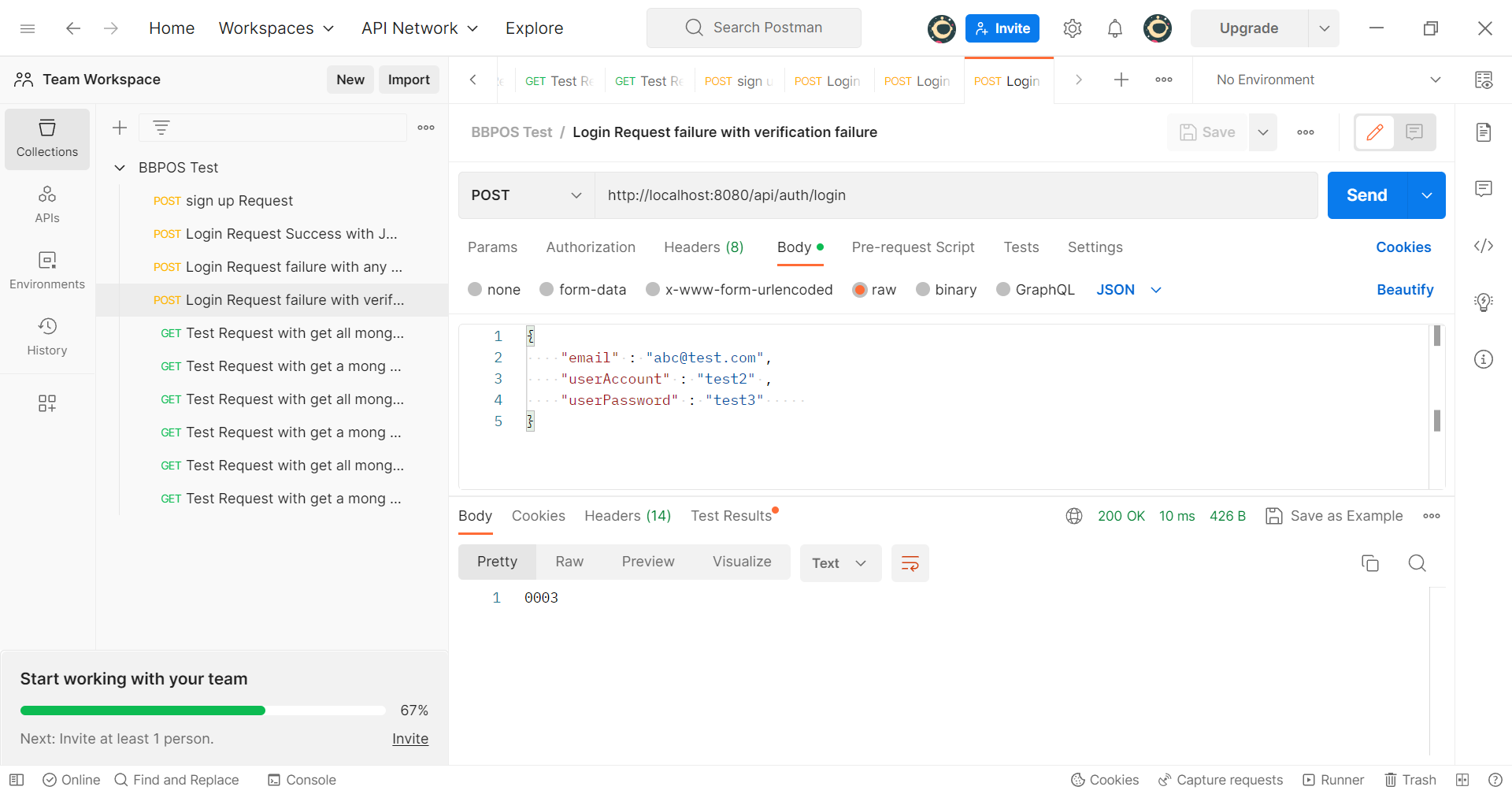
User account test2 but with missing / null email parameter input in the HTTP POST API request /api/auth/login.



The API will return with a message “null data” indicates missing input parameter to the call.

*Scenario 4:*

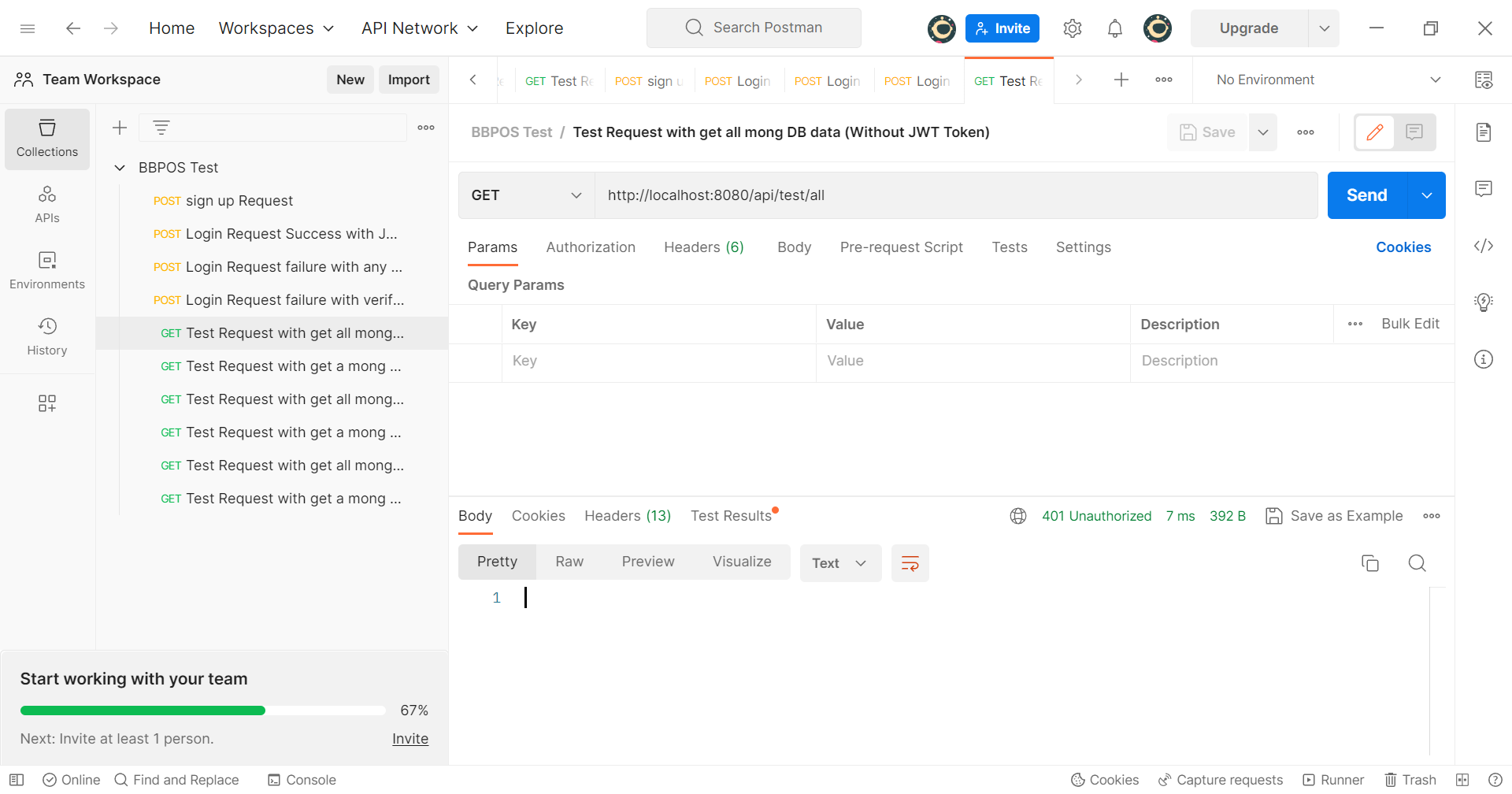
User account test2 but with invalid password input to the HTTP POST API request /api/auth/login.



The business bean of the authentication failure happens, an error code will be return from the call to indicate verification failure. The above capture screen return with “0003” indicating the input password is incorrect.

*Scenario 5:*

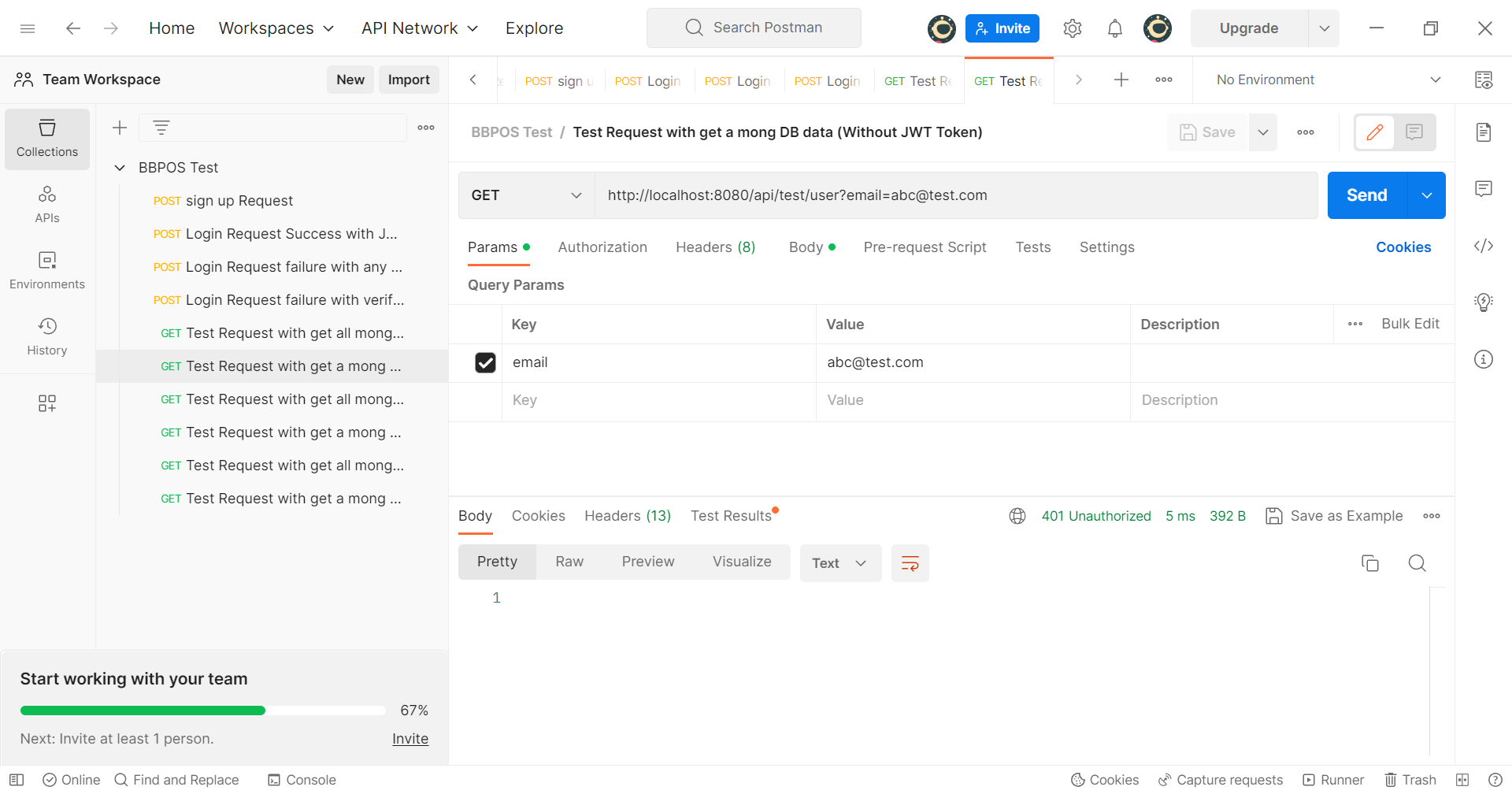
To trigger the HTTP GET API “/api/test/all” to get all the User profiles from Mongo DB without JWT Token inserted in the request authentication header.



Due to absent of JWT token, HTTP response code 401 indicates Unauthorized access is return.

*Scenario 6:*

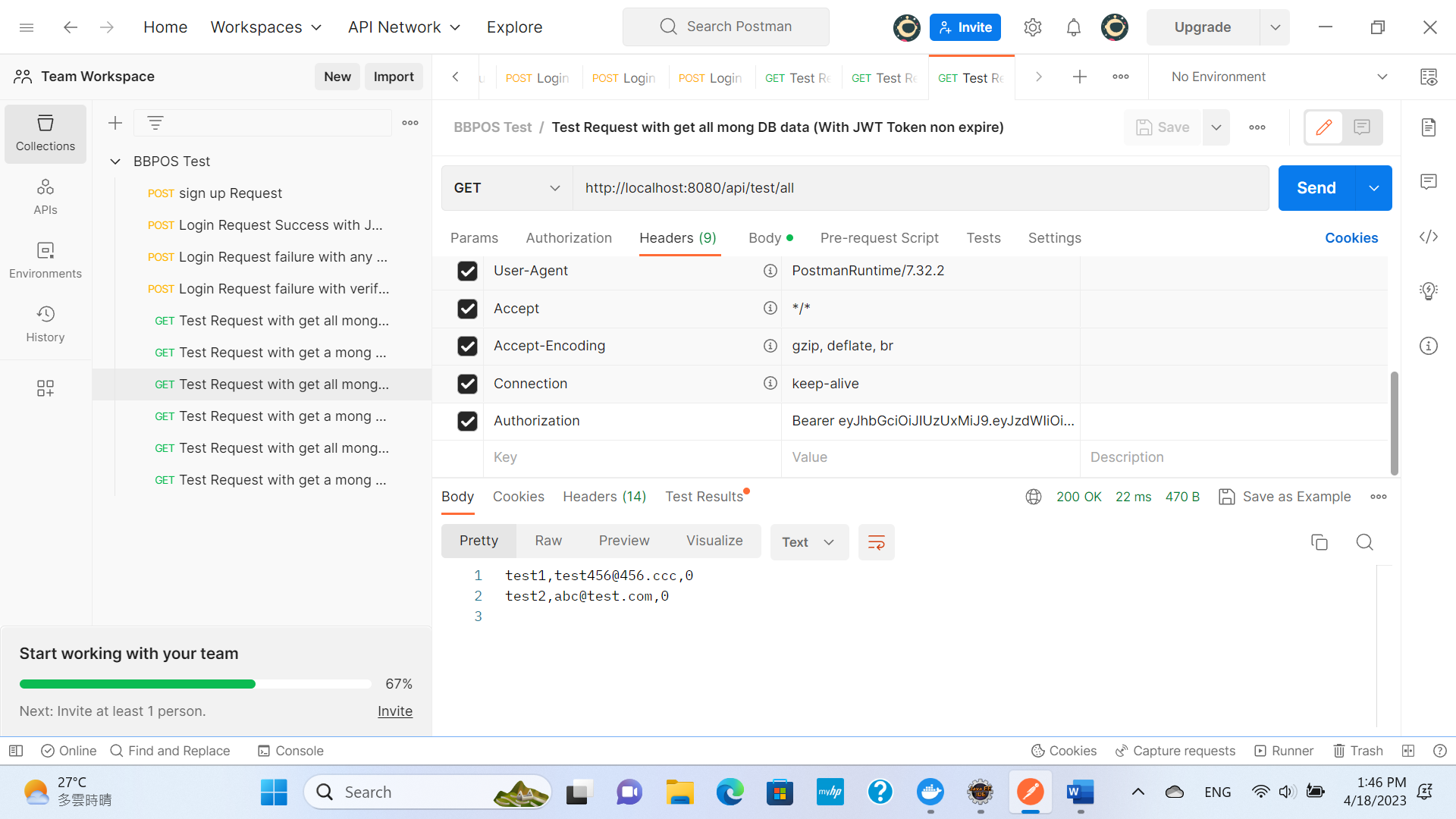
To trigger the HTTP GET API “/api/test/user” to get a particular user profile from Mongo DB according to email request parameter without JWT Token inserted in the request authentication header.

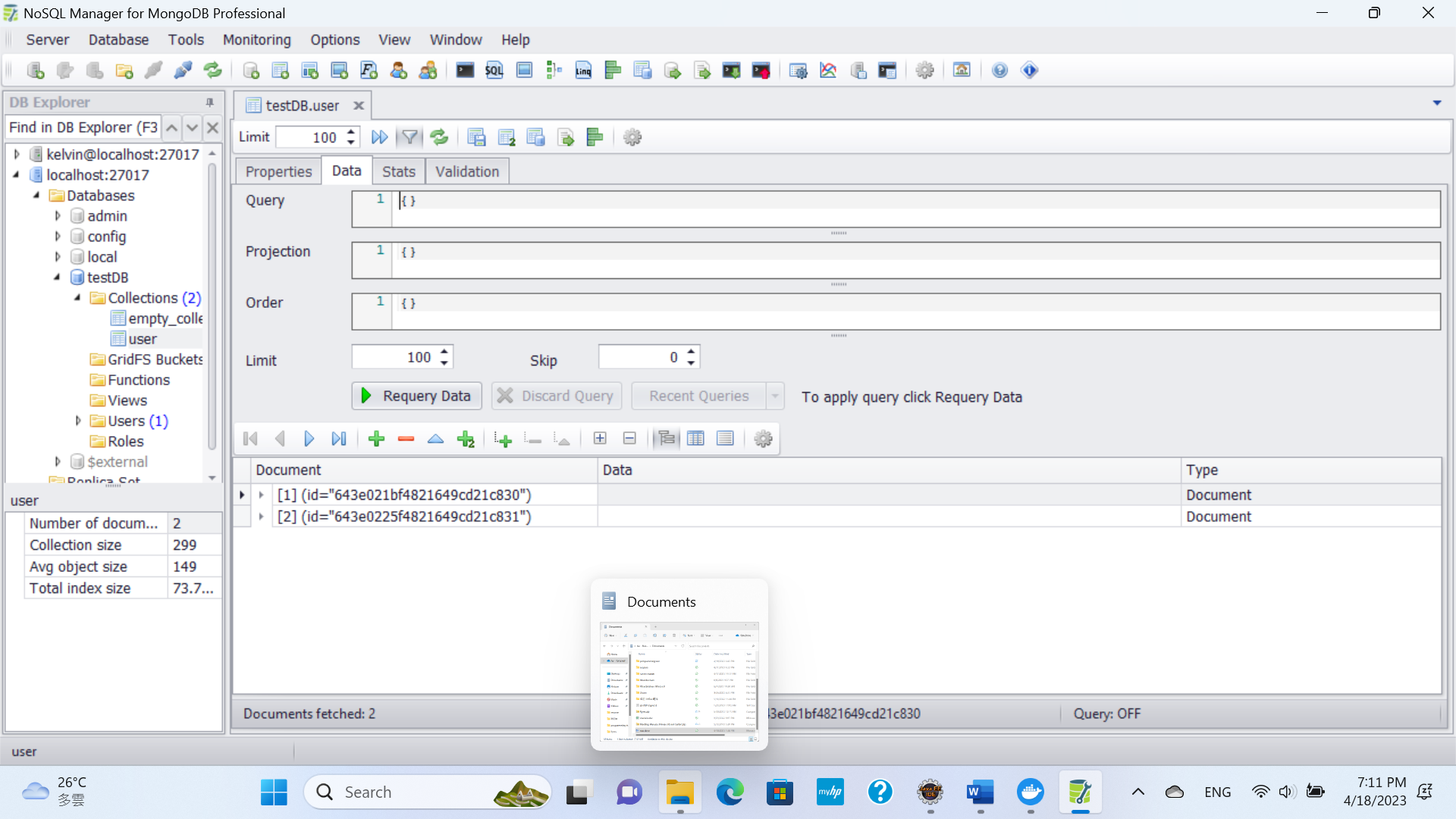


Due to absent of JWT token, HTTP response code 401 indicates Unauthorized access is return.

*Scenario 7:*

To trigger the HTTP GET API “/api/test/all” to get all the User profiles from Mongo DB with valid and non-expire JWT Token inserted in the request authentication header.

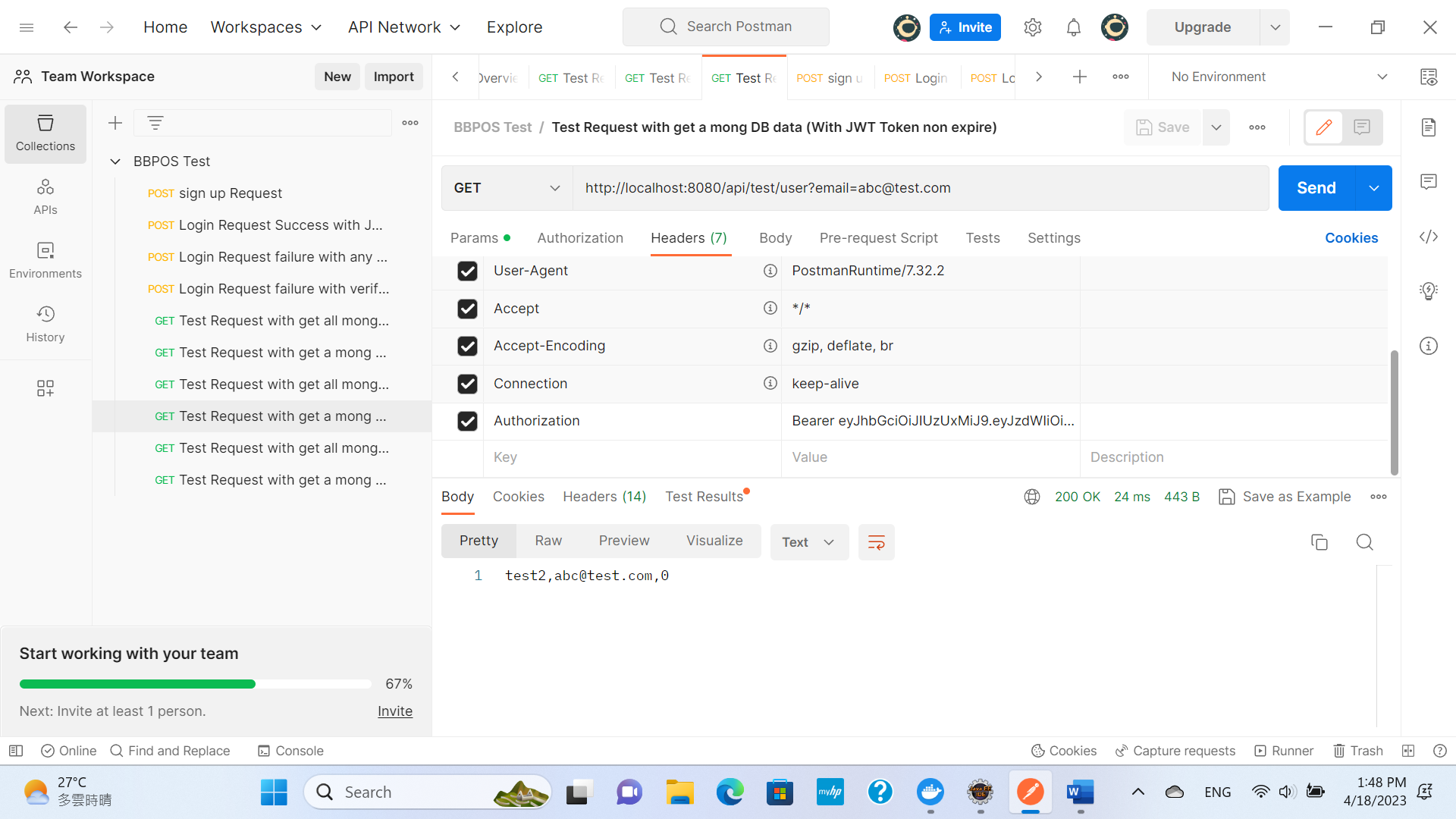




Total 2 records created in the Mongo DB and according to the capture screen of Postman, all user profiles records are returned via the API call.

*Scenario 8:*

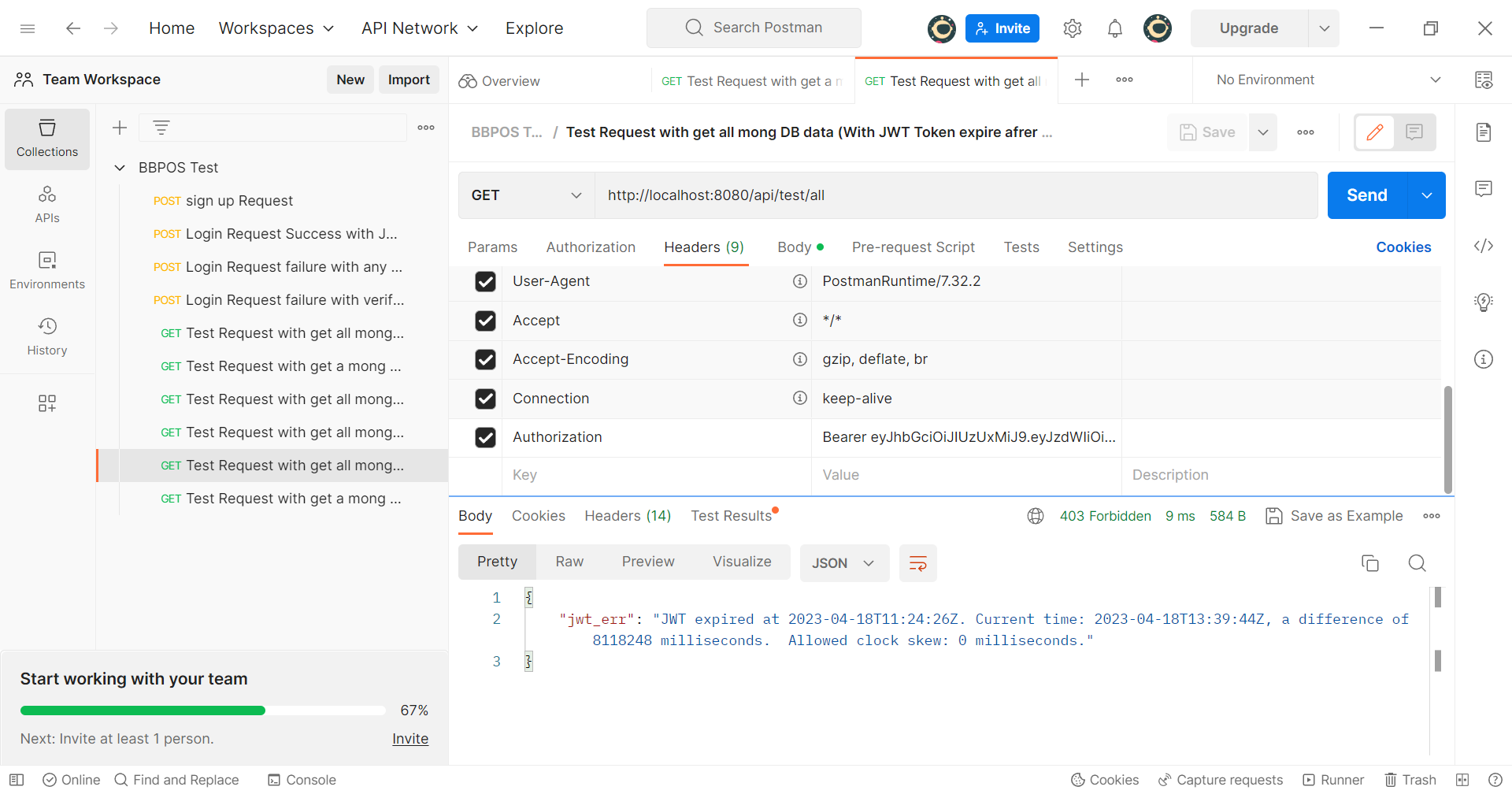
To trigger the HTTP GET API “/api/test/user” to get a particular user profile from Mongo DB according to email request parameter with valid and non-expire JWT Token inserted in the request authentication header.



A particular user profile according to email [abc@test.com](mailto:abc@test.com) return from the HTTP API called

*Scenario 9:*

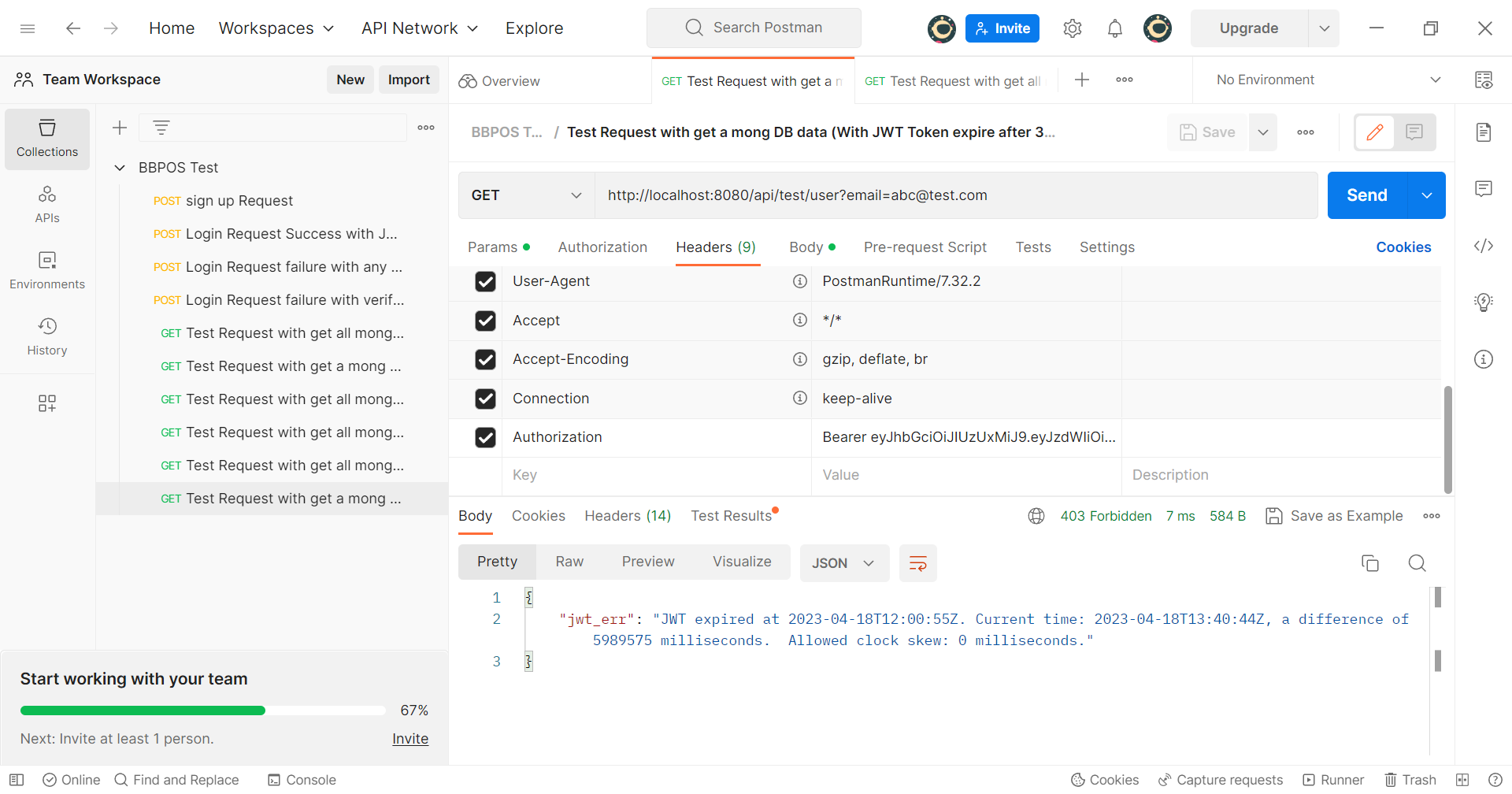
To trigger the HTTP GET API “/api/test/all” to get all the User profiles from Mongo DB with an expired JWT Token inserted in the request authentication header.



The HTTP response code 403 return to indicate the JWT token is expire and spring boot program forbidden to further action.

*Scenario 10:*

To trigger the HTTP GET API “/api/test/user” to get a particular user profile from Mongo DB according to email request parameter with an expired JWT Token inserted in the request authentication header.



The HTTP response code 403 return to indicate the JWT token is expire and spring boot program forbidden to further action.