

Java Assignment

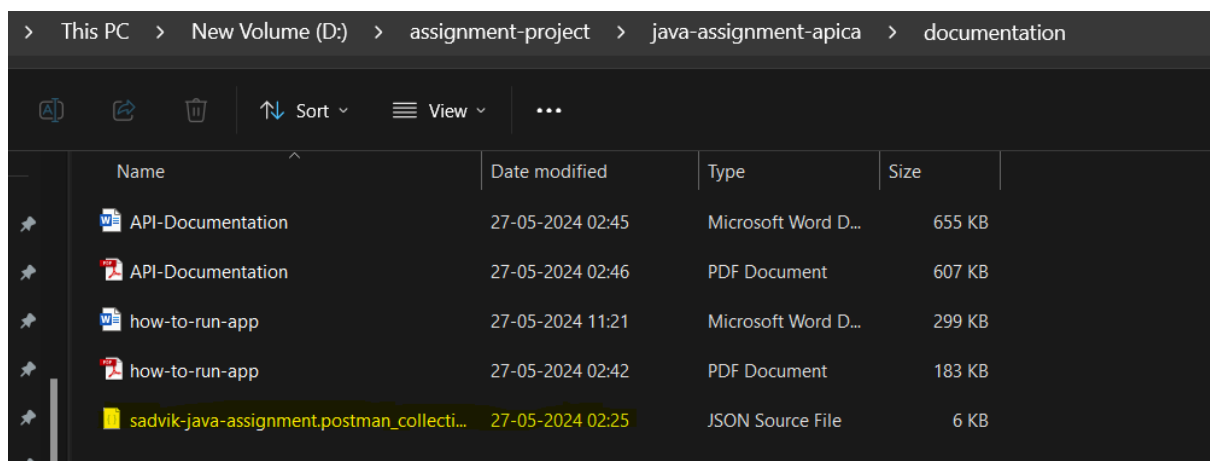
Two spring boot microservices are created to maintain user details and to log the user events.

The user-service application does the CRUD operations like register, get, update and delete user details and produce the event message to user-event kafka topic. And the other application journal-service consume message from user-event topic and write those messages into database, and Admin user can access those user event messages

Prerequisite:

Please import the postman collect from the below repo.

GitHub link :- <https://github.com/Sadvik-gowda-BD/java-assignment-apica>



1)user-service application

This application does the CRUD operation on user details and publish the message “user-event” kafka topic.

GitHub link for source code: - [GitHub - Sadvik-gowda-BD/user-service](https://github.com/Sadvik-gowda-BD/user-service)

Api endpoints:-

Here we are going to save only below fields:

Field	Mandatory/Optional
"userId"	
"firstName":"one",	Mandatory field
"middleName":"mname",	Optional field
"lastName":"lname",	Mandatory field

"emailId":"two@gmail.com",	Mandatory field and email validation check is applied. Email should be unique for all customers.
"password":"pwd",	Mandatory field
"role":"ADMIN",	Mandatory filed. (Expected value is only USER or ADMIN)

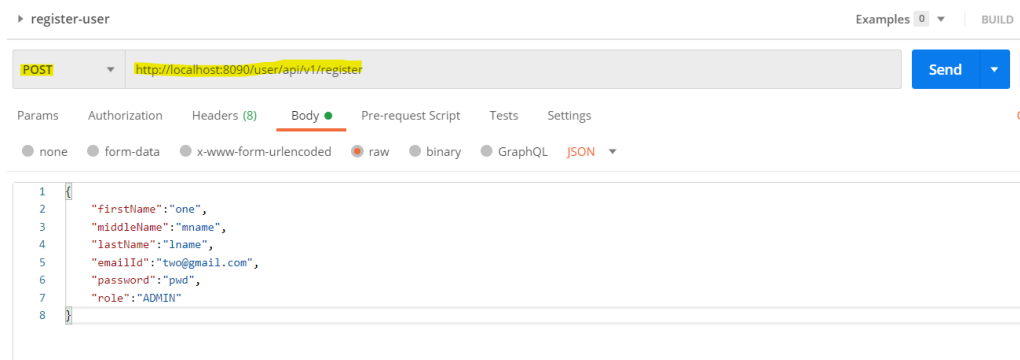
Note:- Please import and use postman collection to invoke API.

1)Register:

This API is used to register the user.

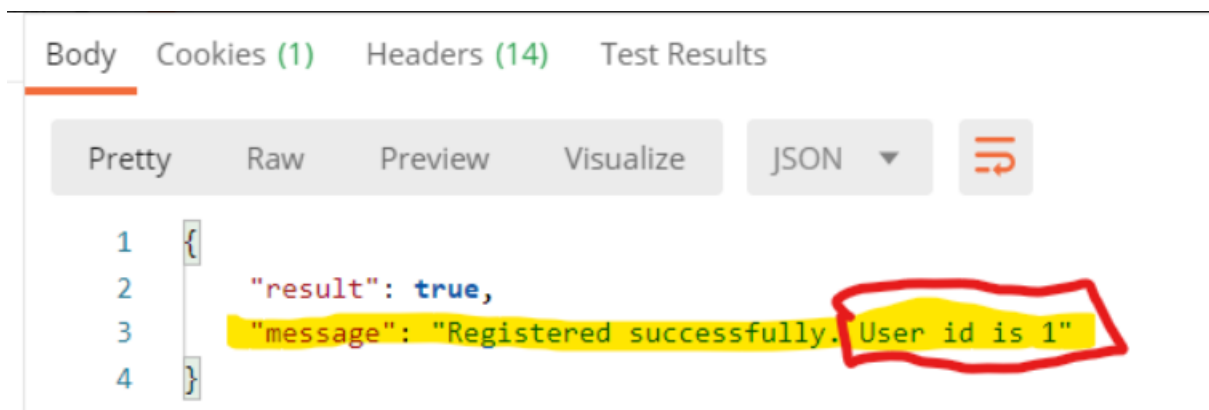
Endpoint: POST: <http://localhost:8090/user/api/v1/register>

Request: JSON object user details



Authorization: No auth

Response: user id, this user id will be used for authentication and identify the user. In below image 1 is the user id.



2) get-user-details-by-id

This API is used to get user details by user id (user id is returned after user registration). Only ADMIN can access this endpoint.

Endpoint: GET <http://localhost:8090/user/api/v1/details/3>

Request: User id in URL(path variable)

Here 3 is the user id.

get-user-by-id Examples 0 BUILD

GET http://localhost:8090/user/api/v1/details/3 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION	...
Key	Value	Description	

Authorization: Admin user id and password

GET http://localhost:8090/user/api/v1/details/3 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

TYPE Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment we recommend using variables. [Learn more about variables](#)

Username 4

Password pwd

☒ Show Password

Response: user details.

Body Cookies (1) Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "userId": 3,
3   "firstName": "three",
4   "middleName": "mname",
5   "lastName": "lname",
6   "emailId": "three@gmail.com",
7   "role": "USER"
8 }
```

3) get-current-user-details

This API is used to get the current login user details. Any user can access this endpoint with user id and password.

Endpoint: GET <http://localhost:8090/user/api/v1/details>

Request: not required any addition data to send because user id can be got from authentication.

▶ get-current-user-details Examples 0

GET <http://localhost:8090/user/api/v1/details>

Params Authorization ● Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Authorization: user id and password

▶ get-current-user-details Examples 0 BUILD

GET <http://localhost:8090/user/api/v1/details> Send

Params Authorization ● Headers (7) Body Pre-request Script Tests Settings

TYPE
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, recommend using variables. [Learn more about variables](#)

Username

Password

☒ Show Password

Response: user details

4) update-user

This API is used to update any user details. Only ADMIN user can access this API.

Endpoint: PUT: <http://localhost:8090/user/api/v1/update>

Request: JSON object with user details.

Note:- update will happen for the given user id.

PUT http://localhost:8090/user/api/v1/update Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "userId": 2,
3   "firstName": "fname",
4   "middleName": "test-update",
5   "lastName": "bd",
6   "emailId": "tesht@gmail.com",
7   "role": "ADMIN"
8 }

```

Authorization: Admin user id and password

get-current-user-details Examples 0 BUILD

GET http://localhost:8090/user/api/v1/details Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username 2

Password pwd

☒ Show Password

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, recommend using variables. [Learn more about variables](#)

Response: user details

5) get-all-user-details

This API is used to get all user details. Only ADMIN can access this endpoint.

Endpoint: GET : <http://localhost:8090/user/api/v1/details/all>

Request:- Not required any additional data.

Authorization:- Admin user id and password

GET http://localhost:8090/user/api/v1/details/all Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username 3

Password ...

☐ Show Password

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, recommend using variables. [Learn more about variables](#)

Response: List of all user details

6)delete-by-user-id

This API is used to delete the user by id. Only ADMIN can access this API.

Endpoint: DELETE: <http://localhost:8090/user/api/v1/delete/2>

Request: user id in URL(path variable)

Here 2 is the user id for deletion.

The screenshot shows the REST client interface for the endpoint `delete-user-by-id`. The HTTP method is set to `DELETE` and the URL is `http://localhost:8090/user/api/v1/delete/2`. The `Params` tab is selected, showing no query parameters. Other tabs like `Authorization`, `Headers`, `Body`, `Pre-request Script`, `Tests`, and `Settings` are visible.

Authorization: Admin user id and password

The screenshot shows the `Authorization` tab selected. The `TYPE` is set to `Basic Auth`. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment we recommend using variables. [Learn more about variables](#)". The `Username` field contains `3` and the `Password` field contains `pwd`. The `Show Password` checkbox is checked.

Response: Delete successful message

The screenshot shows the `Body` tab selected, displaying the response in JSON format. The response is:

```
{  "result": true,  "message": "Deleted user 2 successfully"}
```

. The `Headers` tab shows 14 headers, and the `Test Results` tab is also visible.

2)journal-service application

This application consume message from “**user-event**” kafka topic and save those details into database, and it has API to get all events.

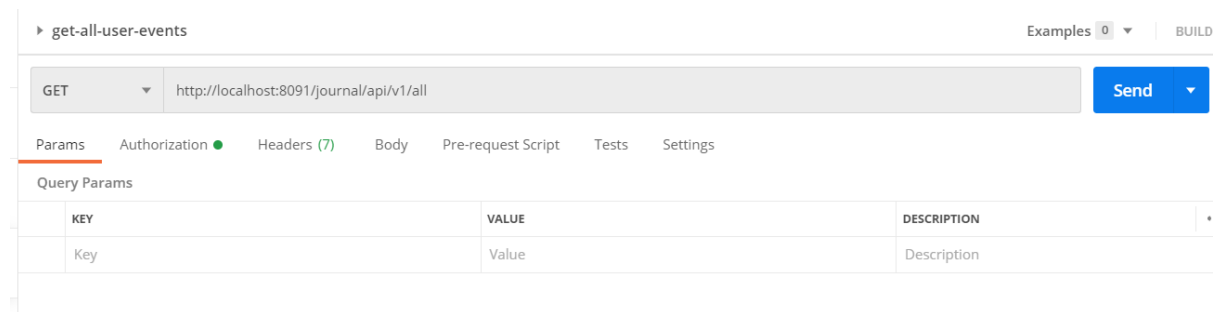
GitHub link for source code: - [GitHub - Sadvik-gowda-BD/journal-service](#)

Api endpoints:-

1)get-all-user-events

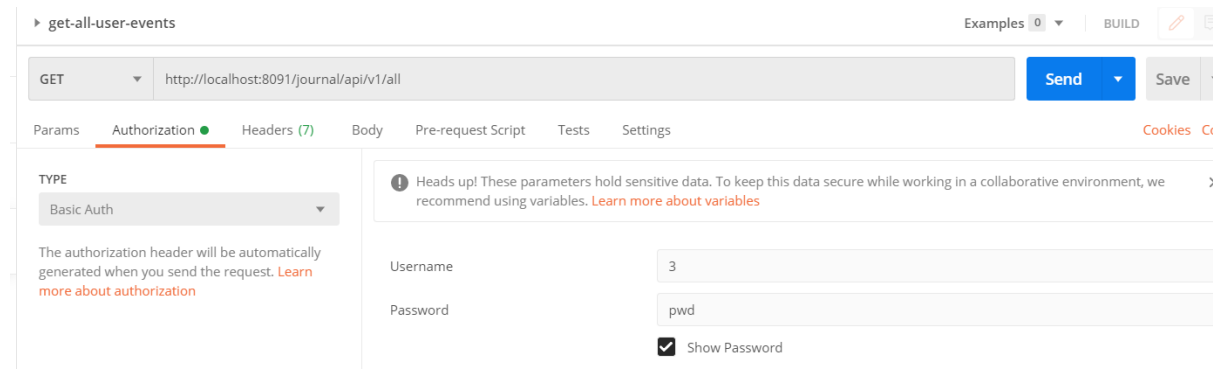
Endpoint: GET: <http://localhost:8091/journal/api/v1/all>

Request: Not require any additional details.



KEY	VALUE	DESCRIPTION
Key	Value	Description

Authorization: Admin user id and password



Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username: 3

Password: pwd

☒ Show Password

Response:All user events.

Sample response object.

```
{  
  "eventId": 9, // event id  
  "eventDescription": "Requested to update user data", // event description  
  "accessedBy": "3", //user id of the accessed by user  
  "accessedFor": "1", //user id of the accessed for user  
  "createDateTime": "2024-05-27T06:39:02.821434" //Event create date time  
}
```

GET http://localhost:8091/journal/api/v1/all

Params Authorization Headers (8) Body Pre-request Script Tests Settings

TYPE
Basic Auth

Username 3
Password pwd
☒ Show Password

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body Cookies (1) Headers (14) Test Results

Status: 200 OK Time: 91 r

Pretty Raw Preview Visualize JSON

```
52 {
53   "eventId": 8,
54   "eventDescription": "Registration for new user completed successfully",
55   "accessedBy": "NEW USER",
56   "accessedFor": "NEW USER",
57   "createDateTime": "2024-05-27T06:38:20.872860"
58 },
59 {
60   "eventId": 9,
61   "eventDescription": "Requested to update user data",
62   "accessedBy": "3",
63   "accessedFor": "1",
64   "createDateTime": "2024-05-27T06:39:02.821434"
65 },
66 {
67   "eventId": 10,
```

Contact number:

Please call to following mobile number for any doubts: +91 9741471445

Email-id: sadvikgowda3333@gmail.com