

IMG STORE APPLICATION

API TESTING DOCUMENTATION

Following is the documentation of Api testing with all valid inputs

Link to these Postman apis -

<https://documenter.getpostman.com/view/19211140/2sAYdeKrGD>

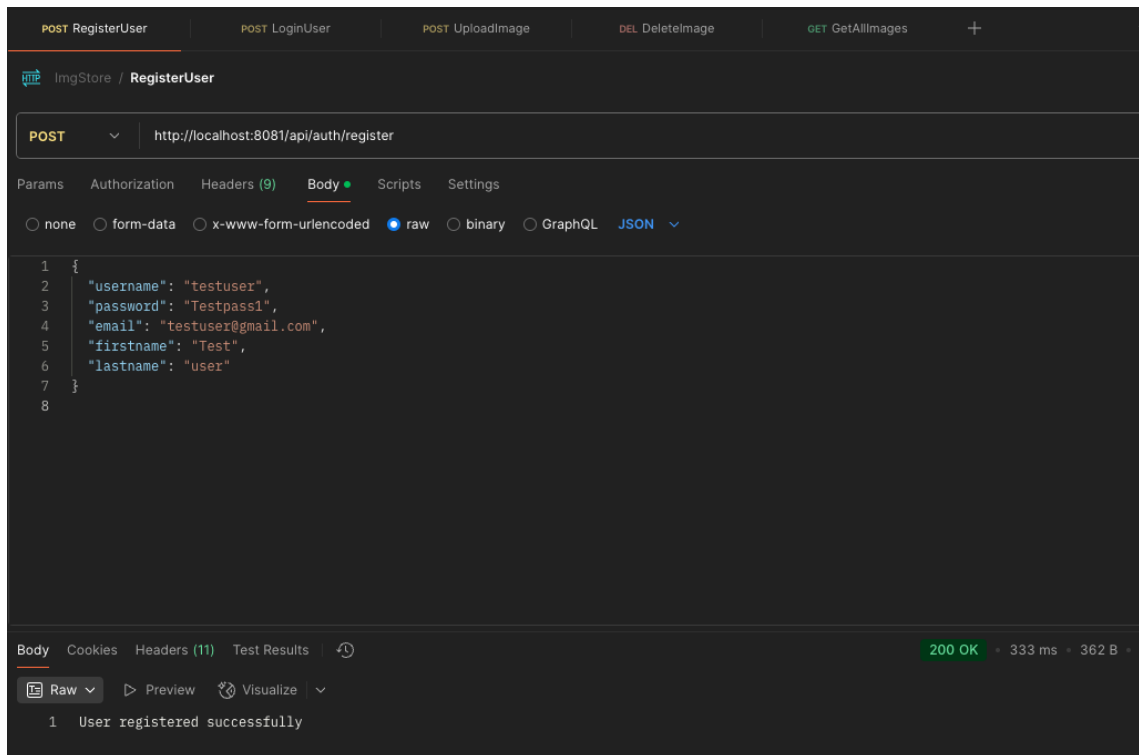
- Authentication Endpoints

- POST /api/auth/register
- POST /api/auth/login

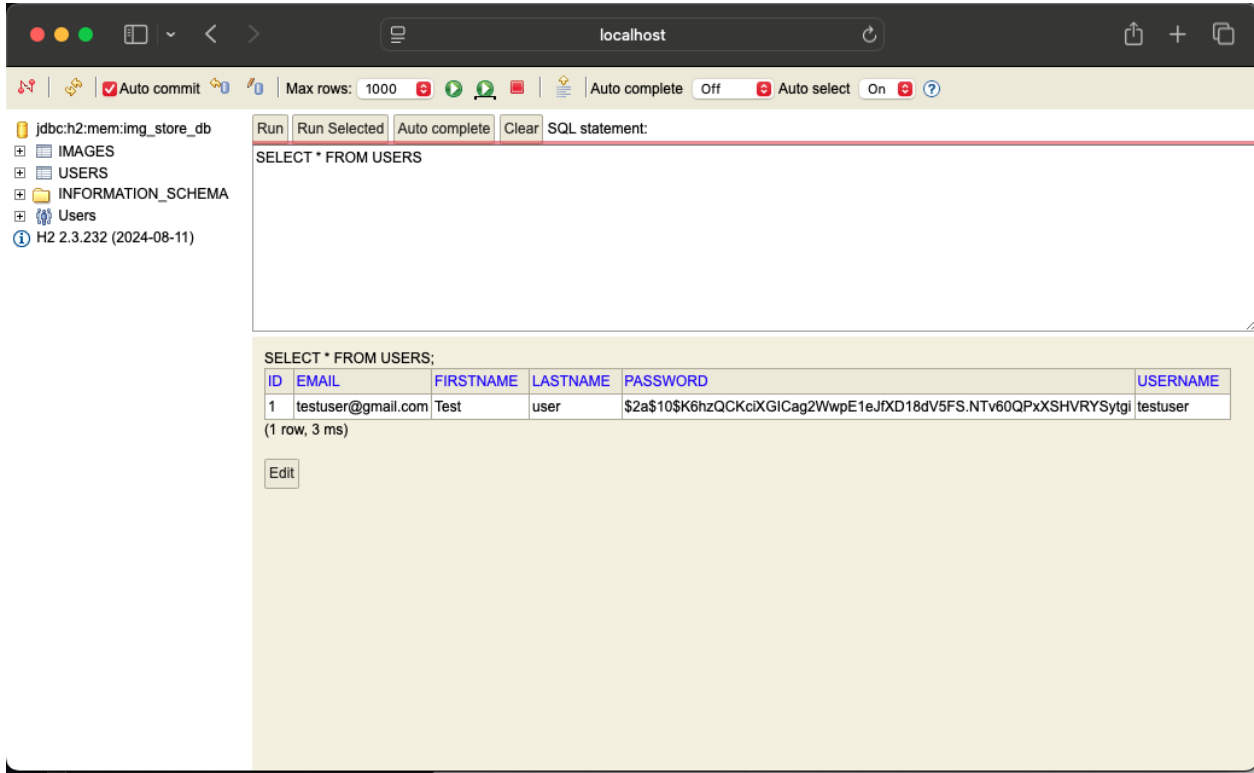
- Image Endpoints

- POST /api/images/upload
- GET /api/images/all
- GET /api/images/{id}
- DELETE /api/images/delete/{deleteHash}

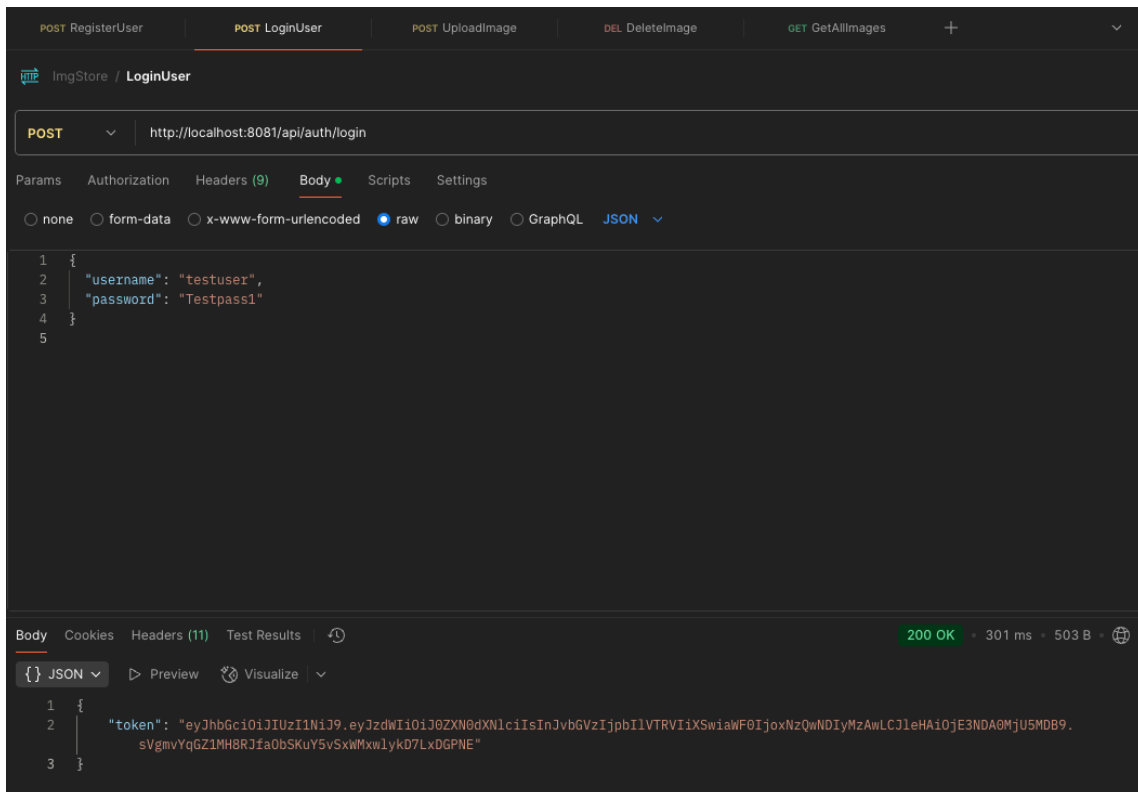
/register



Configured DB



/Login



/upload with kafka disabled

The screenshot shows a Kafka console consumer terminal window with the command `--topic image-uploads --bootstrap-server localhost:9092`. A red box highlights the message: "No message received when kafka is disabled". To the right, the `application.properties` file is open, showing `kafka.enabled=false` under the Kafka Configuration section.

Below the terminal, the Postman API client shows a successful `POST` request to `http://localhost:8081/api/images/upload` with a `200 OK` status. The response body is a JSON object:

```
{
  "imageLink": "https://i.imgur.com/9wkdFPH.gif",
  "message": "Image uploaded successfully"
}
```

/upload with kafka enabled

The screenshot shows the Kafka console consumer terminal window with the same command. A red box highlights the message: "When kafka is enabled in applications.properties, the consumer gets a message". To the right, the `application.properties` file is open, showing `kafka.enabled=true` under the Kafka Configuration section.

Below the terminal, the Postman API client shows a successful `POST` request to `http://localhost:8081/api/images/upload` with a `200 OK` status. The response body is a JSON object:

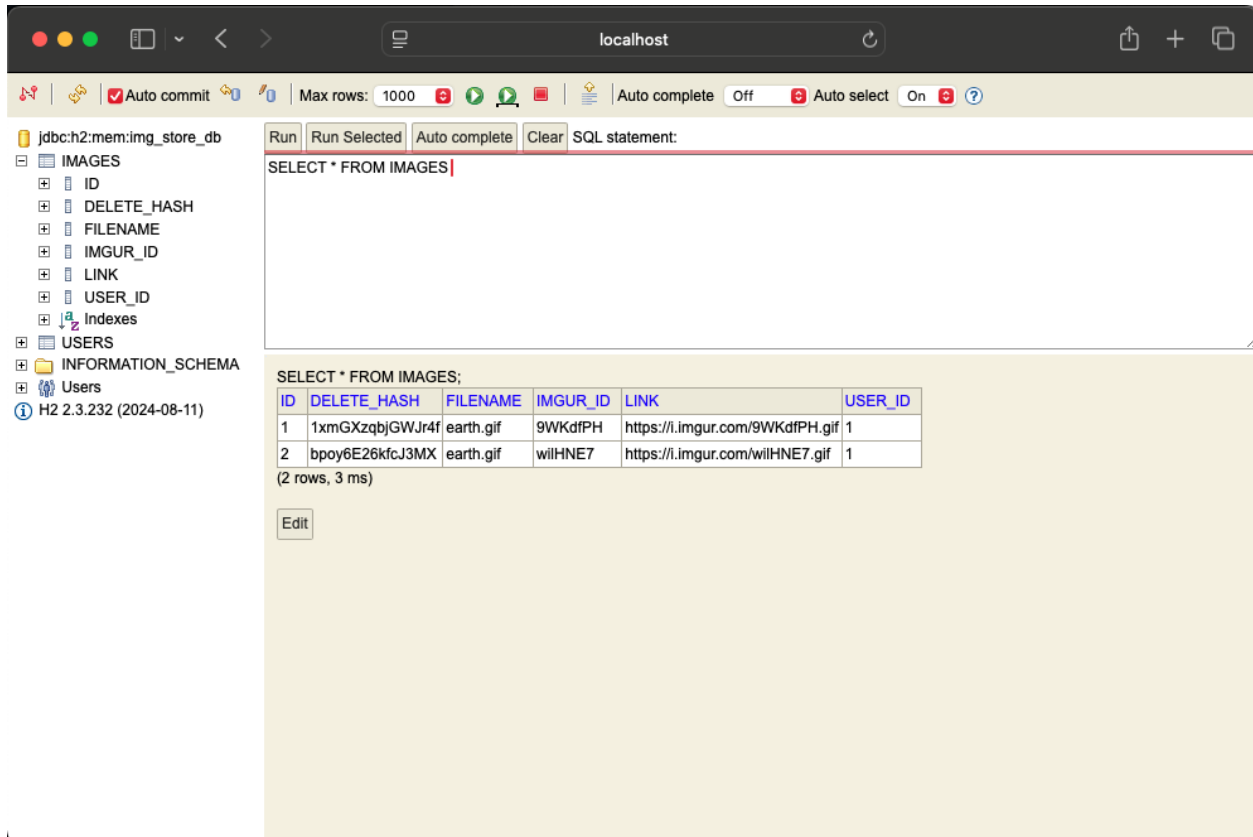
```
{
  "imageLink": "https://i.imgur.com/54kyj3.png",
  "message": "Image uploaded successfully"
}
```

At the bottom, a SQL query is shown in a terminal window:

```
SELECT * FROM IMAGES;
```

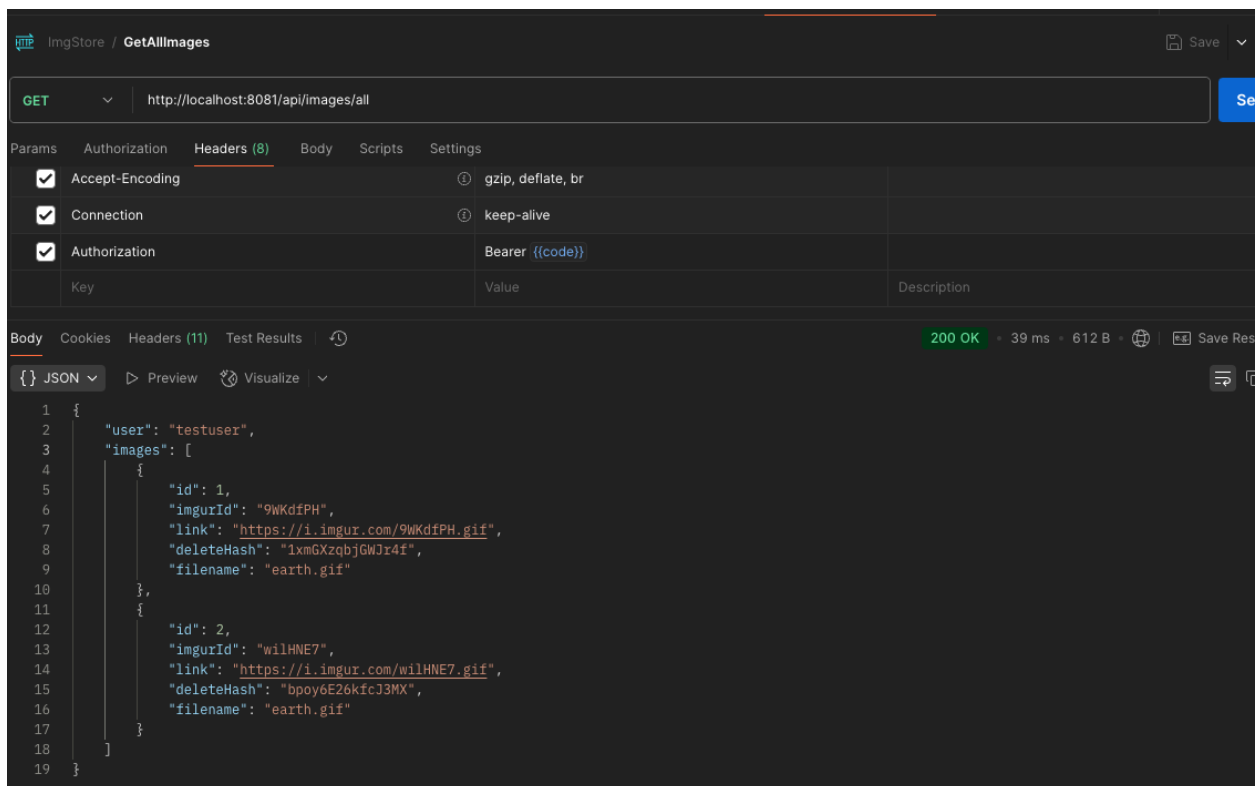
ID	DELETE_HASH	FILENAME	IMOUR_ID	LINK	USER_ID
1	JdV24KXVAlw8Rs	Screenshot 2025-02-24 at 1.43.21 PM.png	JNIGHy3	https://i.imgur.com/JNIGHy3.png	1
2	OmuCyHBPZpYYS	Screenshot 2025-02-24 at 1.43.21 PM.png	xvMRfEs	https://i.imgur.com/xvMRfEs.png	1
3	SIBAZZiUzfyqW	Screenshot 2025-02-24 at 1.43.21 PM.png	CC8kJ4L	https://i.imgur.com/CC8kJ4L.png	1
4	SpUlmY009UIdg	Screenshot 2025-02-24 at 1.43.21 PM.png	eVBn6UD	https://i.imgur.com/eVBn6UD.png	2
5	fWmkyQIwBUCLN	Screenshot 2025-02-24 at 1.43.21 PM.png	7TL0Ll	https://i.imgur.com/7TL0Ll.png	2
6	z524k3H1Y7evh	Screenshot 2025-02-24 at 1.43.21 PM.png	54kyj3	https://i.imgur.com/54kyj3.png	2

When we call api/images/all, we get all the images from the db



The screenshot shows a database client interface with a sidebar on the left displaying the database structure: jdbc:h2:mem:img_store_db, IMAGES (with fields ID, DELETE_HASH, FILENAME, IMGUR_ID, LINK, USER_ID, and Indexes), USERS, and INFORMATION_SCHEMA. The main area shows the SQL statement 'SELECT * FROM IMAGES;' and its results. The results are displayed in a table with 6 columns: ID, DELETE_HASH, FILENAME, IMGUR_ID, LINK, and USER_ID. There are 2 rows of data. Below the table, it indicates '(2 rows, 3 ms)' and an 'Edit' button.

ID	DELETE_HASH	FILENAME	IMGUR_ID	LINK	USER_ID
1	1xmGXzqbJGWJr4f	earth.gif	9WKdfPH	https://i.imgur.com/9WKdfPH.gif	1
2	bpoy6E26kfcJ3MX	earth.gif	wilHNE7	https://i.imgur.com/wilHNE7.gif	1



The screenshot shows a REST client interface for the endpoint 'http://localhost:8081/api/images/all'. The 'Headers' tab is selected, showing 'Accept-Encoding: gzip, deflate, br', 'Connection: keep-alive', and 'Authorization: Bearer {{code}}'. The 'Body' tab is also selected, showing the JSON response. The response status is '200 OK' with a response time of '39 ms' and a body size of '612 B'. The JSON response is as follows:

```
{
  "user": "testuser",
  "images": [
    {
      "id": 1,
      "imgurId": "9WKdfPH",
      "link": "https://i.imgur.com/9WKdfPH.gif",
      "deleteHash": "1xmGXzqbJGWJr4f",
      "filename": "earth.gif"
    },
    {
      "id": 2,
      "imgurId": "wilHNE7",
      "link": "https://i.imgur.com/wilHNE7.gif",
      "deleteHash": "bpoy6E26kfcJ3MX",
      "filename": "earth.gif"
    }
  ]
}
```

After deleting the image, it's removed from the db

The screenshot shows a Postman interface for a DELETE request. The URL is `http://localhost:8081/api/images/delete/{{deleteHash}}`. The Headers tab is active, showing a list of headers including Cache-Control, Postman-Token, Host, User-Agent, Accept, Accept-Encoding, Connection, and Authorization. The Authorization header is set to `Bearer {{code}}`. The Body tab is also active, showing a JSON response: `{ "message": "Image deleted successfully" }`. The status bar indicates a 200 OK response.

Key	Value	Description
Cache-Control		
Postman-Token		
Host	<calculated when request is sent>	
User-Agent	PostmanRuntime/7.43.0	
Accept	*/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
Authorization	Bearer {{code}}	

```
1 {
2   "message": "Image deleted successfully"
3 }
```

The screenshot shows a Postman interface for a GET request. The URL is `http://localhost:8081/api/images/all`. The Headers tab is active, showing a list of headers including Accept-Encoding, Connection, and Authorization. The Authorization header is set to `Bearer {{code}}`. The Body tab is also active, showing a JSON response: `{ "user": "testuser", "images": [{ "id": 2, "imgurId": "wilHNE7", "link": "https://i.imgur.com/wilHNE7.gif", "deleteHash": "bpoy6E26kfcJ3MX", "filename": "earth.gif" }] }`. The status bar indicates a 200 OK response with a response time of 18 ms.

Key	Value	Description
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
Authorization	Bearer {{code}}	

```
1 {
2   "user": "testuser",
3   "images": [
4     {
5       "id": 2,
6       "imgurId": "wilHNE7",
7       "link": "https://i.imgur.com/wilHNE7.gif",
8       "deleteHash": "bpoy6E26kfcJ3MX",
9       "filename": "earth.gif"
10    }
11  ]
12 }
```

Get my image id

The image shows a Postman interface for a GET request to `http://localhost:8081/api/images/{{id}}`. The request is configured with the following headers:

Key	Value
Postman-Token	
Host	
User-Agent	PostmanRuntime/7.43.0
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	Bearer {{code}}

The response body is displayed in JSON format:

```
1 {
2   "id": 2,
3   "deleteHash": "bpoy6E26kfcJ3MX",
4   "filename": "earth.gif",
5   "imgurId": "wilHNE7",
6   "link": "https://i.imgur.com/wilHNE7.gif"
7 }
```